A SET OF EXAMPLES OF GLOBAL AND DISCRETE OPTIMIZATION 2
Home Work for Graduate Students

JONAS MOCKUS
Institute of Mathematics and Informatics,
Kaunas Technological University
Lithuania

Second edition,
in preparation to be published by

Kluwer Academic Publishers
Boston/Dordrecht/London

Contents

1. Preface

This book shows how the Bayesian Approach (BA) improves well-known heuristics by randomizing and optimizing their parameters. That is the Bayesian Heuristic Approach (BHA).

The ten in-depth examples are designed to teach Operations Research using Internet. Each example is a simple representation of some important family of real-life problems.

The accompanying software can be run by remote Internet users. The supporting web-sites include software for Java, C++, and other languages.

A theoretical setting is described in which one can discuss a Bayesian adaptive choice of heuristics for discrete and global optimization problems. The techniques are evaluated in the spirit of the average rather than the worst case analysis. In this context, "heuristics" are understood to be an expert opinion defining how to solve a family of problems of discrete or global optimization. The term "Bayesian Heuristic Approach" means that one defines a set of heuristics and fixes some prior distribution on the results obtained. By applying BHA one is looking for the heuristic that reduces the average deviation from the global optimum.

The theoretical discussions serve as an introduction to examples that are the main part of the book. All the examples are interconnected. Different examples illustrate different points of the general subject. However, one can consider each example separately, too.

In part I, About the Bayesian Heuristic Approach, some well-known results are discussed to help understand what the Bayesian Heuristic Approach is about. First the general ideas of Bayesian related approaches are outlined. This is for readers who want to understand the place of BHA in the general optimization theory and applications. Then it is shown how BHA works by considering a simple example of the knapsack problem. This is for those who are interested in optimization algorithms and would like to skip the underlying theory. In Part II, Software for Global Optimization, different versions of software implementing the Bayesian and other approaches to global optimization are considered. A set of portable Fortran 77 programs, reflecting the early developments of global and local optimization software, is discussed. An interactive Turbo C version, designed for DOS environment, and an interactive C++ version, designed for Unix environment, are described. Three Java versions of global optimization software are presented. The Java software is run by standard Internet browsers.

In part III, Examples of Models, examples of global and discrete optimization are described. The choice of examples aims to help teaching. The teaching topics include: Operations Research, Theory of Games and Markets, Decision Theory, Utility Theory, Queuing Theory, Scheduling Theory, Stochastic Optimization, and Discrete Optimization.

Following mathematical models are considered:

- competition models with fixed resource prices, the "Nash equilibrium,"
- competition models with free resource prices, the "Walras equilibrium,"
- models of dynamic competition,
- bimatrix game models, the "Inspector problem,"
- differential game models, the 'Star Wars' problem,
- investment models, the "Portfolio problem,"
- auto regressive moving average models (ARMA), the problem of exchange rate prediction,
- queuing models for prediction and optimization, the "Call Center problem",
- discrete optimization models, the scheduling problem,
- sequential statistical decision models, the "Bride problem."

The first nine models are solved using a set of algorithms of global and stochastic optimization. The last model is an example of stochastic dynamic programming. All the models are formulated in simple terms to serve as "classroom" examples. However, each of these models can be considered as simple representations of important families of real-life problems. Therefore, the models and the solution algorithms may be of interest for application experts, too.

This book is to describe underlying ideas of algorithms and models in mathematical terms supplemented by illustrative examples and explanations. The information is presented in two forms: hard copy and web-sites.

An advantage of the hard copy is that one does not need a computer to read it. The disadvantage is the static presentation. A hard copy cannot be updated. Therefore, new results must wait until the next edition.

The aim of supporting web-sites is to supplement the algorithms and models by software systems for different computing environments, using different languages and operating systems. An advantage of web-sites is the dynamic presentation. Thus, the results can be easily updated and made accessible free of charge for a wide range of readers. An additional advantage is the possibility to run the software directly by Internet. That is convenient teaching the programming using Internet.

The author considers the hard copy as a static introduction to the dynamic web-sites. There are two of them; they differ by volume and scope.
$http://mockus.org/optimum$
$http://optimum.mii.lt/\tilde{}jonas$
$http://vaidila.vdu.lt/\tilde{}iijomo$

The software presented on these sites is intended for "active" readers, who would like to apply the results to their own problems immediately.

Contact e-mail addresses:
$jonas@optimum.mii.lt$
$jonas@mockus.org$

Finally, I want to express my thanks to colleagues and students who directly or indirectly contributed to this book.

J. Mockus
January 2000

I

ABOUT THE BAYESIAN APPROACH

Chapter 1

GENERAL IDEAS OF BAYESIAN HEURISTIC APPROACH

1. Outline

The traditional numerical analysis considers optimization algorithms that guarantee some accuracy for all functions to be optimized. This includes the exact algorithms. That is the worst case analysis. To limit maximal errors one needs computational efforts that often increase exponentially with the size of the problem. This is the main disadvantage of the worst case analysis.

The alternative is the average case analysis. Here an average error is not limited but is made as small as possible. The average is taken over a set of functions to be optimized. The average case analysis is the Bayesian Approach (BA) (Diaconis, 1988, Mockus, 1989a).

There are several ways to apply BA in optimization. The first one, the Direct Bayesian Approach (DBA), is defined by fixing a prior distribution P on a set of functions $f(x)$ and by minimizing the Bayesian risk function (DeGroot, 1970, Mockus, 1989a). The risk function $R(x)$ is the expected deviation from the global minimum at a fixed point x. The distribution P is considered as a stochastic model of $f(x),\ x \in R^m$, where $f(x)$ might be a deterministic or a stochastic function. In the Gaussian case, assuming (Mockus, 1989a) that the $(n+1)$th observation is the last one

$$R(x) = \frac{1}{\sqrt{2}\pi s_n(x)} \times \int_{-\infty}^{+\infty} \min(c_n, z) e^{-\frac{1}{2}\left(\frac{y-m_n(x)}{s_n(x)}\right)^2} dz. \tag{1.1}$$

Here $c_n = \min_i z_i - \epsilon,\ z_i = f(x_i).\ m_n(x)$ is a conditional expectation with respect to observed values $z_i,\ i = 1, ..., n.\ s_n^2(x)$ is a conditional variance, and $\epsilon > 0$ is a correction parameter. The minimum of the risk function $R(x)$ is

obtained (Mockus, 1989a) at the point

$$x_{n+1} = \arg\max_x \frac{s_n(x)}{m_n(x) - c_n}. \tag{1.2}$$

The objective of DBA, used mainly in continuous cases, is to provide as small average error as possible while keeping convergence conditions.

Another way, the Bayesian Heuristic Approach (BHA), means fixing a prior distribution P on a set of auxiliary functions $f_K(x)$. These functions define the best values obtained by using K times some heuristic $h(x)$. It is assumed that heuristics $h(x)$ depend on some continuous parameters $x \in R^m$. The heuristic helps to optimize an original function $C(y)$ of variables $y \in R^n$, where $m < n$ (Mockus et al., 1997). As usual, components of y are discrete variables. Heuristics are based on expert opinions about the decision priorities. Now both DBA and BHA will be considered in detail.

2. Direct Bayesian Approach (DBA)

The Wiener process is a common (Kushner, 1964a, Saltenis, 1971, Torn and Zilinskas, 1989) stochastic model in the one-dimensional case $m = 1$.
Figure 1.1 shows the conditional expectation, the conditional standard, and the risk function with respect to available evaluations. The Wiener model implies continuity of almost all sample functions $f(x)$. The model assumes that increments $f(x_4) - f(x_3)$ and $f(x_2) - f(x_1)$, $x_1 < x_2 < x_3 < x_4$, are stochastically independent. Here $f(x)$ is Gaussian $(0, \sigma x)$ at any fixed $x > 0$. Note, that the Wiener process originally provided a mathematical model of a particle in the Brownian motion.

The Wiener model is extended to multidimensional case, too (Mockus, 1989a). However, simple approximate stochastic models are preferable if $m > 1$. Approximate models are designed by replacing the traditional Kolmogorov consistency conditions. These conditions require the inversion of matrices of nth order for computing the conditional expectation $m_n(x)$ and variance $s_n^2(x)$ [1].

Replacing the regular consistency conditions by:
- continuity of the risk function $R(x)$
- convergence of x_n to the global minimum
- simplicity of expressions of $m_n(x)$ and $s_n(x)$

the following simple expression of $R(x)$ is obtained using the results of (Mockus, 1989a)

$$R(x) = \min_{1 \le i \le n} z_i - \min_{1 \le i \le n} \frac{\|x - x_i\|^2}{z_i - c_n}.$$

[1] The favorable exceptions are the Markovian processes including the Wiener one. Extending the Wiener process to $m > 1$ the Markovian property disappears.

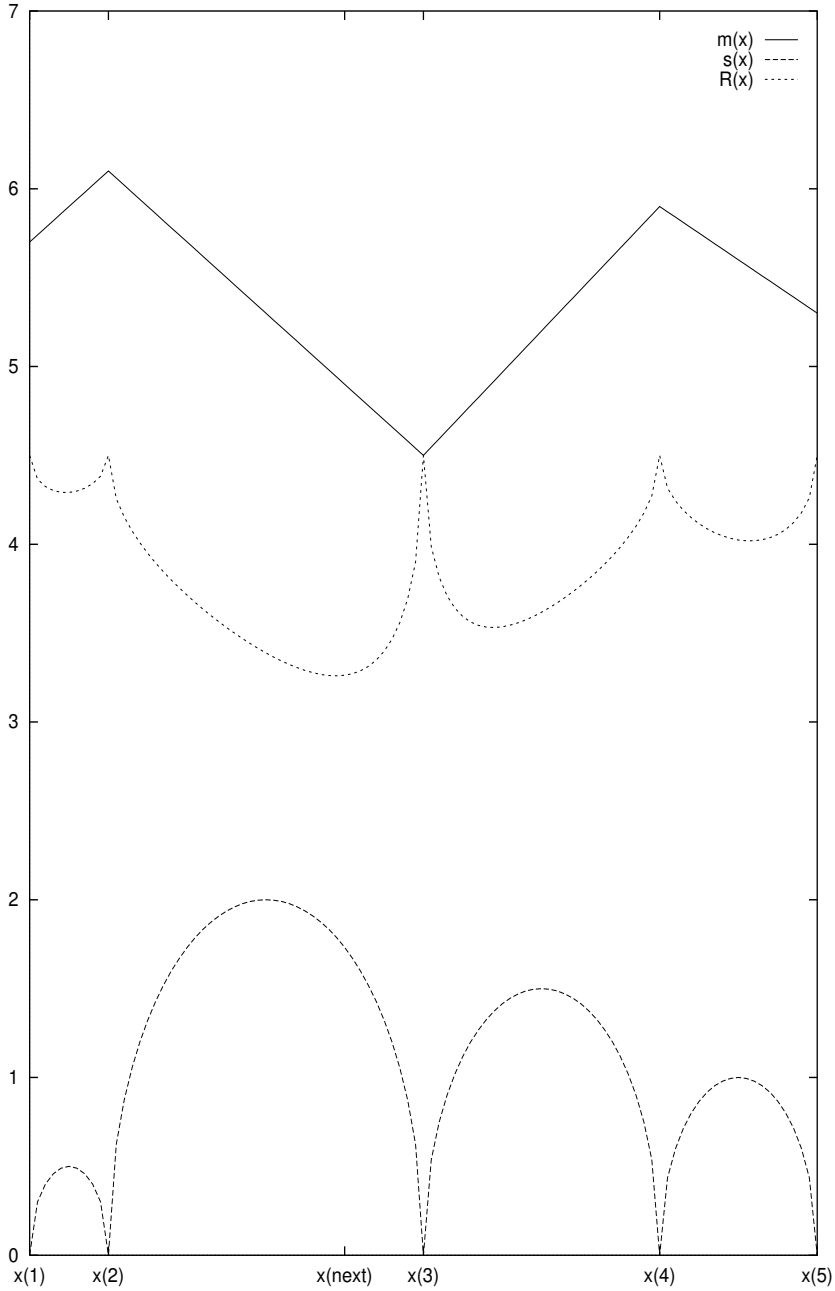


Figure 1.1. The Wiener model. The conditional expectation $m(x)$, the conditional standard $s(x)$, and the risk function $R(x)$ with respect to observed values $x(1), y(1), x(2), y(2), ...$

The aim of DBA is to reduce the expected deviation. In addition, DBA has some good asymptotic properties, too. It is shown (Mockus, 1989a) that

$$d^*/d_a = \left(\frac{f_a - f^* + \epsilon}{\epsilon}\right)^{1/2}, \; n \to \infty.$$

Here d^* is a density of points x_i around the global optimum. d_a is an average density of x_i in the feasible area. f_a is an average value of $f(x)$ in this area. f^* is an average value of $f(x)$ around the global minimum. ϵ is the correction parameter in expression (1.1). That means that DBA provides convergence to the global minimum for any continuous $f(x)$ and a greater density of observations x_i around the global optimum, if n is large.

Note, that the correction parameter ϵ has a similar influence as the temperature in simulated annealing. However, that is a superficial similarity. The good asymptotic behavior is just some "by-product" of DBA. The reason is that Bayesian decisions are applied for small size samples where asymptotic properties are not noticeable. To choose the next point x_{n+1} by DBA, one minimizes the expected deviation $R(x)$ from the global optimum (see Figure 1). The minimization of $R(x)$ is a complicated auxiliary optimization problem. That means that DBA is useful for expensive, in terms of computing times, functions of a few ($m < 20$) continuous variables. This happens in wide variety of problems.

Some examples are described in (Mockus, 1989a). These include maximization of the yield of differential amplifiers, optimization of mechanical systems of a shock absorber, optimization of composite laminates, evaluation of parameters of immunological models and nonlinear time series, planning of extremal experiments on thermostable polymeric compositions. A large set of test problems in global optimization is considered in (Floudas et al., 1999).

3. Bayesian Heuristic Approach (BHA)

The Bayesian Heuristic Approach (BHA) is for optimization of simple objective functions with large variable numbers. That is the case in many discrete optimization problems. As usual, these problems are solved using heuristics based on an expert opinion. Heuristics often involve randomization[2] procedures that depend on some empirically defined parameters.

The examples of such parameters are:
- an initial temperature of the simulated annealing,
- probabilities of different randomization algorithms in their "mixture."

In these problems DBA is a convenient tool for optimization of the continuous parameters of various heuristic techniques and their mixtures. That is the Bayesian Heuristic Approach (BHA) (Mockus et al., 1997).

[2]The randomization means that a decision depends on some random variable included into an algorithm deliberately.

Applying DBA, the expert knowledge is involved by defining a prior distribution. In BHA one exploits expert knowledge "twice". First time one does that by defining the heuristics. Second time the expert knowledge is involved by defining a prior distribution, needed to optimize parameters of heuristics using DBA.

4. Illustrative Examples

The example of a knapsack problem is convenient to illustrate basic principles of BHA in the discrete optimization. Given a set of objects $j = 1, ..., n$ with values c_j and weights g_j, find the most valuable collection of a limited weight g

$$\max_y C(y),\ C(y) = \sum_{j=1}^{n} c_j y_j,\ \sum_{j=1}^{n} g_j y_j \le g.$$

Here the objective function $C(y)$ depends on n Boolean variables $y = (y_1, ..., y_n)$, where $y_j = 1$, if an object j is in the collection, and $y_j = 0$, otherwise. The well known greedy heuristic $h_j = c_j/g_j$ is the specific value of an object j. The greedy heuristic algorithm: "take the greatest feasible h_j", is very fast but it may get stuck in some non optimal decision. One may force the heuristic algorithm out of there by taking a decision j with some probability $r_j = \rho_x(h_j)$[3]

Here DBA is applied to optimize parameters x of the auxiliary function $f_K(x)$. This function defines the best results obtained applying K times a randomized heuristic algorithm $\rho_x(h_j)$. That is the most expensive operation of BHA. Therefore, the parallel computing of $f_K(x)$ should be used, if possible[4]. Optimization of x adapts the heuristic algorithm $\rho_x(h_j)$ to a given problem. The parameter x may be useful to solve related problems (Mockus et al., 1997). That helps using the "on-line" optimization.

We illustrate the parameterization of $\rho_x(h_j)$ by using three randomization functions $r_i^l = h_i^l / \sum_j h_j^l,\ l = 0, 1, \infty$. Here the superscript $l = 0$ denotes the uniformly distributed component. $l = 1$ defines the linear component of randomization. The superscript ∞ denotes the pure heuristics with no randomization. That means $r_i^\infty = 1$, if $h_i = \max_j h_j$ and $r_i^\infty = 0$, otherwise. The parameter $x = (x_0, x_1, x_\infty)$ defines probabilities of using the randomizations $l = 0, 1, \infty$, correspondingly. That can be understand as some "lottery of algorithms" where x_l is a probability to "win" the algorithm l. The detail description of the knapsack example is in the next chapter.

[3]Meaning that a randomization is applied. For example:
- generate an uniformly distributed random number $\xi \in [0, 1]$
- take a decision j if $\xi < r_j$, where $\rho_x(h_j)$ is an increasing function of h_j and $x = (x_1, ..., x_m)$ is a parameter vector.

[4]In this case the solution time is inversely proportional to the number of parallel processors.

5. Heuristics

The main objective of BHA is to improve any given heuristic by defining the best parameters of some heuristic and/or the best "mixtures" of different heuristics. Heuristic decision rules, mixed and adapted by BHA, often outperform[5] the best individual heuristics, as judged by examples. In addition, BHA provides almost sure convergence. However, the results of BHA depend on the quality of specific heuristics. The quality of heuristics reflects the expert knowledge. That means, BHA should be considered as a tool for enhancing the heuristics but not for replacing them.

Many well-known optimization algorithms, such as the Genetic Algorithms (GA) (Goldberg, 1989), GRASP (Mavridou et al., 1998), and the Tabu Search (TS) (Glover, 1994), may be considered as some general heuristics that can be improved using BHA. There are specific heuristics tailored to fit particular problems. For example, the Gupta heuristic was the best one while applying BHA to the flow-shop problem (Mockus et al., 1997).

The Genetic Algorithms (Goldberg, 1989) is an important "source" of interesting and useful stochastic search heuristics. It is well known (Androulakis and Venkatasubramanian, 1991) that results of the genetic algorithms depend on mutation and crossover parameters. The Bayesian Heuristic Approach could be used in optimizing those parameters.

In the GRASP system (Mavridou et al., 1998), a heuristic is repeated many times. During each iteration, a greedy randomized solution is constructed and the neighborhood around that solution is searched for the local optimum. The "greedy" component adds one element at a time, until a solution is constructed. A possible application of BHA in GRASP is in optimizing a random selection of a candidate to be in the solution. BHA might be useful as a local component, too, by randomizing the local decisions and optimizing the corresponding parameters.

In the Tabu search, two important parameters may be optimized using BHA. Those are:

- best combinations of short and long term memory,
- best balances of intensification and diversification strategies.

The examples show that the Bayesian Heuristics Approach is useful applying many well-known stochastic and heuristic algorithms of discrete optimization. The convergence conditions, if needed, are provided by tuning the BHA (Mockus et al., 1997).

[5] In terms of average errors.

Chapter 2

EXPLAINING BAYESIAN HEURISTIC APPROACH BY EXAMPLE OF KNAPSACK PROBLEM

In the previous chapter we showed how BHA works while searching for the solution of the knapsack problem. Now we shall use this problem to illustrate similarities and differences of various approaches including BHA.

The reason for such special attention is that the knapsack problem is a simple example of important family of NP-complete problems[1]. Note, that this example is good mainly for illustration of BHA and alternative approaches. The reason is that the simple greedy heuristic of the knapsack problem is very efficient. Therefore, only marginal improvements can be made by more sophisticated methods, including BHA. In scheduling problems, such as the flow-shop and batch scheduling (Mockus et al., 1997), BHA works more efficiently. However, the scheduling problems are more difficult for explanations. Thus, we start the explanations by the knapsack problem.

The knapsack problem is to maximize the total value of a collection of objects when the total weight g of these objects is limited. Denote values of objects i by c_i and their weights by g_i.

$$\max_y \sum_{i=1}^{n} c_i y_i \tag{2.1}$$

$$\sum_{i=1}^{n} g_i y_i \le g \tag{2.2}$$

$$y_i \in \{0, 1\}.$$

Here the objective $C(y) = \sum_{i=1}^{n} c_i y_i$ depends on n Boolean variables $y_i,\ i = 1, ..., n$.

[1]No polynomial time algorithm has been found for solving the NP-complete problems. No proof is known that these problems cannot be solved in polynomial time.

1. Exact Algorithms

1.1 Exhaustive Search

The decision m means that a collection of objects $I(m) = \{i : y_i(m) = 1\}$ is selected. The value of the collection is $c(m) = C(y(m))$. This collection is described by the decision vector $y = y(m) = (y_i(m),\ i = 1, ..., n)$. The simplest exact algorithm is to compare all the collections:

1 select a current collection $I(m^N),\ N = 1, ...K$,

2 record the best current collection $I^*(m^N)$,

3 stop when all the collections are considered $N = K = 2^n$.

The best current collection $I^*(m^N)$ denotes the most valuable collection satisfying the weight limit that is obtained during N iterations. The best current collection is updated during the recording operation in step two. It is replaced by better current collection. The exhaustive search of all the decisions needs $K = 2^n$ of iterations. This means $T = C\ 2^n$ of time, where the constant C is the observation time (the CPU time needed to evaluate sum (2.1) and to test inequality (2.2)).

1.2 Branch & Bound (B&B)

The efficiency of exact algorithms can be improved by the Branch & Bound (B&B) techniques:

1 define the decision tree,

2 define the upper limits of the branches,

3 cut the branch, if the upper limit is less then the best current collection,

4 stop, if there are no more branches to consider.

For example,

1 branch 0: $y_1 = 0$, branch 1: $y_1 = 1$,

2 the upper limit of branch 1:

$$C^1 = c_1 + \max_y \sum_{i=2}^{n} c_i y_i$$

$$g_1 + \sum_{i=2}^{n} g_i y_i \le g$$

the upper limit of branch 0:

$$C^0 = \max_y \sum_{i=2}^{n} c_i y_i$$
$$\sum_{i=2}^{n} g_i y_i \le g$$

3 cut branch 0, if $C^0 \le c_1$,

4 stop, if there are no more branches to consider.

In this algorithm, the number of iterations to obtain the exact solution is $K \le 2^n$ and the time is $T \le C\, 2^n$. Usually, the time T is much less then the exact upper limit $C\ 2^n$. However, this limit may be approached. For example, if all the prices c_i and weights g_i are almost equal. In such a case no branches will be cut. That is a reason why approximate algorithms are preferred, if n is large.

2. Approximate Algorithms

2.1 Monte Carlo

The simplest approximate algorithm is Monte-Carlo (MC):

1 select a collection $I(m^N),\ N = 1, ..., K$ with probability [2]

$$r(m) = \frac{1}{2^n}, \tag{2.3}$$

2 record the best current collection $I^*(m^N)$,

3 stop, if the number of iterations $N = K$.

Recording just the best current collection, one samples with replacement. Otherwise, one should keep in the memory up to 2^n of previous samples. In cases with replacement, one does not know when the optimum is reached. Therefore, MC stops, if the time allocated for the optimization is exhausted. This algorithm converges to an exact solution with probability one, if the number of iterations $K \to \infty$. However, the convergence is very slow, nearly logarithmic. Note, that the time $T = Cn^m$ is not a limit for MC with replacement.

[2] That can be understood as a lottery with the probability $r(m)$ to win a collection m. This lottery can be carried out by generating a random number ξ uniformly distributed in $[0, 1]$ and by selecting a collection m if $\xi < r(m)$.

2.2 Greedy Heuristics

Define the heuristic h_i as the specific value of the object i

$$h_i = \frac{c_i}{g_i}. \tag{2.4}$$

This "greedy" heuristic is the well known and widely used in the knapsack problem. The Greedy Heuristic Algorithms prefer a feasible[3] object with the highest heuristic h_i:

1 select the object i with the best heuristic h_i in the starting list[4],

2 move the object from the starting list into the current one,

3 test feasibility of the collection which includes this object,

4 if the collection is infeasible, delete the object from the starting list,

5 test, whether the starting list is empty,

6 if the staring list is not empty, go to the step one,

7 if the starting list is empty, stop and record the current list as the best collection $I^*(m^N)$, where $N \leq n$ is number of the last iteration.

The greedy heuristic algorithm is fast, $T < C\ n^2$. The inequality is there because objects are gradually removed from the starting list. The algorithm may stick in some non optimal decision. The greedy heuristic h_i is obviously useless, if the specific values are equal $h_i = h,\ i = 1, ..., n$. In such cases the outcome depends on the numeration of objects.

2.3 Randomized Heuristics

Linear Randomization

Heuristic algorithms are shaked out of non optimal decisions by choosing an object i with some probability r_i that is proportional to the specific value h_i,

$$r_i = \frac{h_i}{\sum_{j=1}^{n} h_j}. \tag{2.5}$$

This algorithm is similar to that of Greedy Heuristic:

1 select an object i from the starting list at random with the probability r_i,

2 move the object from the starting list into the current one,

[3] Satisfying inequality (2.2).

[4] Initially the starting list includes all the objects

3 test feasibility of the collection including this object,

4 if the collection is infeasible, delete the object from the starting list,

5 test, if the starting list is empty,

6 if the staring list is not empty, go to step 1,

7 if the starting list is empty, stop and record the current list as a current collection $I(m^N),\ N = 1, ..., K$,

8 record the best current collection $I^*(m^N)$,

9 start new iteration $N + 1$ by restoring starting list to the initial state (as a set of all the objects) and return to step 1,

10 stop after K iterations and record the best current collection $I^*(m^K)$.

This algorithm is better than the greedy one, in the sense that it converges with probability one when $K \to \infty$. The algorithm is expected to be better than Monte-Carlo, too, since it includes an expert knowledge by relating the decision probabilities to heuristics.

Mixed Randomization

We may choose the preferred heuristic algorithm by considering all of them separately. Here the quality estimate of each algorithm is the best result obtained after K iterations. That is a traditional way.

One solves the same problem in a more general setup by considering a "mixture" of different heuristic algorithms. For example,

- denote the Monte Carlo by the index $l = 0$,
- denote the Linear Randomization by the index $l = 1$,
- denote the Greedy Heuristics by the index $l = 2$.

Then the Mixed Randomization uses the algorithm denoted by the index l with some probability $x(l)$ that is defined by probability distribution $x = (x(l),\ l = 0, 1, 2)$:

1 select an algorithm l at random with probability $x(l)$,

2 record the best current collection $I^*(m^N),\ N = 1, ..., K$,

3 start the new iteration $N + 1$ by returning to step 1,

4 stop after K iterations and record the best current collection $I^*(m^K)$, denote its value by $f_K(x) = c(m^K)$.

Here $f_K(x)$ is the best result obtained using K times the Mixed Randomization with the probability distribution x[5].

That is a way to extend a family of possible decisions from the discrete set $x(l) \in \{0,1\},\ l = 0,1,2$ to the continuous one. That is important because the best results we often obtain using a mixture of algorithms (Mockus et al., 1997).

Here we mixed three approximate algorithms: the Monte Carlo, the Linear Randomization and the Greedy Heuristics. Following expressions define a larger family of different approximate algorithms, including these three

$$r_i^l = \frac{h_i^l}{\sum_{j=1}^n h_j^l},\ l = 0,1,2,...,L, \tag{2.6}$$

and

$$r_i^\infty(m) \ = \ \begin{cases} 1, & \text{if } h_i = \max_j h_j, \\ 0, & \text{otherwise.} \end{cases} \tag{2.7}$$

Here the superscript $l = 0$ denotes the Monte-Carlo randomization. The superscripts $l = 1,\ l = 2,....,l = L$ define a family of "polynomial" randomizations. The superscript ∞ denotes the greedy heuristics with no randomization. The Mixed Randomization means using randomization functions $r_i^l,\ l = 0,1,2,...,L,\infty$ with probabilities $x(l)$ and recording the best result $f_K(x)$ obtained during K iterations.

Optimization of Mixture

The optimization of the mixture x is difficult because $f_K(x)$ is a stochastic function and the multimodal one, as usual. A natural way to consider such problems is by regarding functions $f_K(x)$ as samples of a stochastic function defined by some prior distribution. The next observation is obtained by minimizing the expected deviation from the exact solution. This technique is called the Bayesian Heuristic Approach (BHA) (Mockus et al., 1997).

2.4 Permutation

If we build a solution from "scratch," then we may apply so-called greedy heuristics (Helman et al., 1993). Building from scratch is convenient, if no initial expert decision is known. Otherwise, building from scratch, one ignores the expert information included into that initial decision. As usual, the permutation algorithms are used to improve initial decisions.

In the knapsack problem, the complete collection I is a set of all n objects. The initial collection is a set of objects $I(m^0) = \{i : y_i(m^0) = 1\} \in I$

[5]The algorithm is implemented in C++ and Java in the files $fi.C$ and $Knapsack.java$ correspondingly (see section 4 for details).

satisfying the inequality $\sum_{i \in I(m^0)} g_i y_i(m^0) \leq g$. The value of this collection is $c(m^0)$. A complement of the initial set $I(m^0)$ is denoted $I^0(m^0)$. We denote permuted collections as $I(m^j),\ j = 1, ..., J$ and values of these collections as $c(m^j)$. We define the direct permutation heuristics as

$$h_j = h(m^j) = c(m^j) - c(m^0). \tag{2.8}$$

To avoid confusion, the longer symbols $h(m^j)$ and $hn(m^j)$ are used sometimes instead of the short ones h_j and hn_j. We define the normalized permutation heuristics as

$$hn_j = hn(m^j) = \frac{h_j - A_j}{A^j - A_j}, \tag{2.9}$$

where $A_j = \min_{k=1}^{J} h_k$ and $A^j = \max_{k=1}^{J} h_k$. Normalized heuristics (2.9) can be easily converted into probabilities by expressions such as (2.6). Direct heuristics (2.8) are convenient in some well-known algorithms, such as Simulated Annealing. Here is an example of permutation algorithm using linear randomization:

1 make J feasible changes $m_j^0,\ j = 1, ..., J$ of the initial collection $I(m^0)$, by replacing randomly selected objects using this algorithm

   (a) select at random with equal probabilities an object j_1 from initial collection $I(m^0)$,

   (b) delete this object,

   (c) select at random with equal probabilities an object j_0 from complement $I^0(m^0) = \{i : y_j(m^0) = 0\}$ of the initial collection, skip this step if the complement is empty,

   (d) put the object j_0 into the initial collection,

   (e) test the weight limit,
   return to step one of the replacement algorithm, if the updated collection is to heavy,

2 define normalized permutation heuristics hn_j for all the changes $j = 1, ..., J$,

3 define probabilities r_j by expression (2.13),

4 select current collection $I(m^N),\ N = 1, ..., K$ at random with probability r_j,

5 record the best current collection $I^*(m^N)$,

6 stop after K iterations and record the best current collection $I^*(m^K)$.

Simulated Annealing

The Simulated Annealing (SA) is a popular global optimization method. Features of SA, considering it as a permutation algorithm, are:
- only one permutation is made, $J = 1$,
- the direct heuristic is used

$$h_j = c(m^j) - c(m^0), \tag{2.10}$$

- if $h_j \geq 0$, the new collection $I(m^j)$is selected with probability one,
- if $h_j < 0$, the new collection is selected with probability r_j that declines exponentially,

$$r_j = \begin{cases} e^{\frac{h_j}{x/\ln(1+N)}}, & \text{if } h_j < 0, \\ 1, & \text{otherwise.} \end{cases} \tag{2.11}$$

Here N is the iteration number and x is the "initial temperature." The difference of this formulation from traditional simulating annealing algorithms is that we optimize the parameter x for some fixed number of iterations $N = K$. We disregard the asymptotic behavior. The asymptotic properties are beyond the scope of the Bayesian Heuristics Approach.

Genetic Algorithms

General Idea. Genetic Algorithms (GA) present a general and visual way of permutation and randomization. GA is a methodology for searching a solution space in the manner similar to the natural selection procedure in biological evolution (Holand, 1975). Each candidate solution is represented by a string of symbols. The set of solutions at some state m is called the population of the mth generation. The population evolves for a prescribed number M of generations. The basic structure processed by GA is the string. Strings are composed of a sequence of characters.

A simple GA consists of
- one reproductive plan, defined as the fitness proportional reproduction
- two genetic operators, defined as crossover and mutation.

The probability of selection is defined in proportion to the fitness of individuals

During the crossover operation one splits two selected strings at some random position s. Then two new strings are created, by exchanging all the characters up to the split point s. During the mutation operation one alters some string characters at random. A mutation is of l-th order, if one changes l elements during one mutation operation. Both the crossover and the mutation are feasible, if they satisfy all constraints.

It follows from this definition that GA may be considered as a special case of the Permutation and Randomization. Thus, one may include GA into the framework of Bayesian Heuristic Approach. We illustrate that by using the Knapsack problem as an example.

Explaining Genetic Algorithm by Knapsack Example. In the Knapsack example the string of the collection $I(m)$ is represented as the Boolean vector

$$y(m) = (y_i(m),\ i = 1, ..., n). \tag{2.12}$$

The fitness of feasible[6] collection m^j is $c(m^j)$. The probability to select a string m^j is

$$r_j = \frac{hn_j}{\sum_{k=1} hn_k}, \tag{2.13}$$

where hn_j is the normalized permutation heuristics

$$hn_j = hn(m^j) = \frac{c(m^j) - A_j}{A^j - A_j}. \tag{2.14}$$

Here $A_j = \min_{k=1}^{J}\ c(m^k)$ and $A^j = \max_{k=1}^{J}\ c(m^k)$.

During the crossover operation one splits two selected strings at some random position $s,\ 1 < s < n$. Then two new strings are created, by exchanging all the characters up to the split point s.

During the mutation operation one produces J mutants by inverting randomly selected components y_j of the Boolean vector y. A mutation is of l-th order, if one changes $l,\ 1 \leq l \leq n$ components during one mutation operation. A mutant is fertile, if it satisfies all the constraints.

Denote by $I(m^N)$ a current collection at Nth iteration. Denote by $I^*(m^N)$ the best current collection obtained during N iterations.

Here is an example of a simple GA algorithm written in BHA terms. The algorithm is similar to that of Permutation.

1 produce a number J of fertile mutants by replacing randomly selected objects using feasible changes $m_j^0,\ j = 1, ..., J$ of the initial collection $I(m^0)$,

2 define normalized permutation heuristics $hn_j,\ j = 1, ..., J$ for all these mutants using expression (2.14),

3 define probabilities r_j by expression (2.13),

4 select two mutants $I_j(m^+)$ and $I_j(m^-)$ at random with probabilities r_j[7],

5 select a split point s at random with probability $1/n$

6 inverse the components $y_i(m^+),\ i \leq s$ and $y_i(m^-),\ i \leq s$ of these two mutants,

[6]Assume for simplicity that all the non feasible collections are eliminated automatically.

[7]If by chance the same mutant $m = m^+ = m^-$ will be selected then repeat the selection procedure.

7 update normalized permutation heuristics hn_j reflecting cross-over results

8 update probabilities r_j using (2.13)

9 select a current collection at random with the probability r_j,

10 record the best current collection $I^*(m^1)$

11 go to step 4 with the probability $x(0)$

12 produce J fertile mutants by feasible changes of the current collection,

13 define normalized permutation heuristics $hn_j,\ j = 1, ..., J$,

14 define probabilities r_j by expression (2.13),

15 select a current collection at random with the probability r_j,

16 record the best current collection $I^*(m^2)$ and return to step 11,

17 stop after K iterations and record the best current collection $I^*(m^K)$.

It is tacitly assumed that some segments of strings define specific "traits." In such cases, one may "enforce" good traits by uniting "good" segments. One may expose bad traits by uniting "bad" segments, too. Then, reproduction plans tend to favor the good traits and to reject the bad ones.

Thus, the crossover operation "improves" the population, if the "traits" assumption is true. If not, then the crossover may be considered merely as a sort of mutation. That may help to jump the area dividing separate "local" minima. However, the similar jump may be accomplished by high order mutations, too.

As usual (Androulakis and Venkatasubramanian, 1991), mutations are considered as more "radical" operations as compared with crossovers. That is correct, if one changes many elements of the "genetic" sequence during one mutation. This happens, if the mutation order l is close to the string length n.

The results of some real life examples of network optimization (Mockus, 1967) and parameter grouping (Dzemyda and Senkiene, 1990) show that low order mutations, when merely a few elements of the decision vector $r(k)$ are changed, work better. In such cases, the mutation may be considered as a less radical operation because fewer components of the vector y are changed, as compared with the crossover operation.

We may improve efficiency of the simple GA algorithm by using the crossover rate $x(0)$ (see step eleven of the algorithm) as an optimization variable in the framework of BHA. This was applied solving the Batch Scheduling problems (Mockus et al., 1997). If the crossover rate is considered as a time function (Androulakis and Venkatasubramanian, 1991), the parameters of this function can be optimized using BHA.

3. Software Examples of Knapsack Problem

There are two software versions: one in C++ and one is Java. In the C++ version, prices c_i and weights g_i are generated randomly. In the Java case, $c_i,\ g_i$ reflects a real life example (Malisauskas, 1998). The optimal probability $x(\infty)$ of the greedy heuristic $h_i = c_i/g_i$ is near to one, but not exactly one. Some degree of randomization helps to escape non optimal decisions. The randomized procedures $x(l), r_i^{(l)}$ are defined by (2.6) and (2.7). Both the software versions consider just three components.

The $C++$ version uses the Monte Carlo, the Linear and the Quadratic randomizations. Corresponding optimization parameters are
$x = (x(0), x(1), x(2))$. The Monte Carlo and Linear randomizations and the Pure Greedy heuristic are applied in the Java version. Corresponding optimization parameters are
$x = (x(0), x(1), x(\infty))$.

We are looking for such probability distribution x that provides the maximal $f_K(x)$ after K iterations. Bayesian methods (Mockus, 1989a, Mockus and Mockus, 1990, Mockus et al., 1997) are used for that.

3.1 C++

The aim of the GMC version is to estimate average error of BHA. A set of knapsack problems with random prices c_i and weights g_i is considered. The results of BHA and the exact deterministic B&B algorithms are in Table 2.1.

Average results were obtained by repeating the optimization procedures $K = 100$ times at fixed parameters $c_i,\ g_i$ and probability distribution x. In Table

Table 2.1. Comparison of the Bayesian method and the exact one.

$K_B = 100$, and $K = 1$							
N_O	K_E	f_B	f_E	$\delta_B\%$	$x_B(0)$	$x_B(1)$	$x_B(2)$
50	313	9.56057	9.62347	0.654	0.0114	0.0280	0.9605
100	526	13.0703	13.1241	0.411	0.0316	0.0412	0.9271
150	771	16.6301	16.6301	0.000	0.0150	0.1945	0.7904
200	875	37.4665	37.4859	0.050	0.0315	0.0530	0.9437
250	568	53.7781	53.9000	0.226	0.0091	0.0511	0.9397
300	1073	28.3144	28.6106	1.034	0.0113	0.0835	0.9050
350	1416	30.4016	31.7527	4.254	0.0064	0.0646	0.9288
400	2876	32.1632	33.3192	3.469	0.0202	0.0452	0.9344
450	1038	105.467	105.578	0.105	0.0101	0.0149	0.9748
500	2132	39.3583	42.1047	6.521	0.0078	0.1556	0.8365

2.1 N_0 is the number of objects. K is the number of repetitions. K_B is the

total number of observations by the Bayesian method. K_E is the number of nodes considered by the exact method. f_B is the best result of the Bayesian method. f_E is the exact optimum. $\delta_B\%$ is the mean percentage error of the Bayesian algorithm. $x_B(n), n = 0, 1, 2$, are optimized probabilities of different randomizations obtained using a Bayesian method. If the deviation of some solution does not exceed 5%, we call this a 5% solution. One can expect that the 5% solution satisfies the applications, where the level of data uncertainty is not less than 5%.

Table 2.1 shows that we need to consider from 313 to 2876 nodes to obtain the exact solution. Only 100 observations are needed to obtain the 5% solution by the Bayesian method. The deviation exceeds 1% only for three cases in ten. The average deviation is 1.67%.

Assume that roughly the same computing time is necessary for one node and for one observation. Then the Bayesian 5% solution is about three times "cheaper" as compared to the exact one, if the number of objects is fifty. If this number is 400, then the Bayesian 5% solution is almost thirty times cheaper. Other examples, in particular ones applying BHA to a family of scheduling problems (Mockus et al., 1997) show higher efficiency of BHA.

3.2 Java

Here w e optimize a "mixture" x of the Monte Carlo randomization, the linear randomization, and the pure greedy heuristic. The aim is to show how BHA works while solving a real life knapsack problem. The example illustrates how to apply the Java software system for global optimization called as GMJ1. Therefore, several figures are included. They illustrate the input and output of GMJ1 graphical interface.

Data File

The data represents the weights, the values, the numbers, and the names of inventory items of the "Norveda" shop that sells "Hitachi" electrical tools. Table 2.2 shows a fragment of data file 'norveda.txt'. Here fields, separated by spaces, denote these parameters of objects:

- weights g_i (double),
- prices c_i (double),
- numbers (integer),
- names (string).

Implementing Task

The algorithm is implemented as a $Task$ of the GMJ1 system. Figure 2.1 shows a fragment of the file 'Knapsack.java'

Weight	Value	Number	Name
3.8	2830	1	CNF35U
10.5	4170	2	CNF65U
11.5	3850	1	CM12Y
1.8	1500	2	CE16
1.7	1500	2	CN16
17.0	2100	4	CC14
20.9	2890	2	J9312N
0.9	330	8	D6SH
1.7	1170	10	D10YA
1.3	630	5	D10VC

Table 2.2. A fragment of data file 'norveda.txt.'

```
public class Knapsack implements Task     { private double
norm_point[];          private int    number_of_objects;
public  double total_weight=10;        private int
object_taken[];        private float  object_cost[];
private float  object_weight[];        private double
object_ratio[];        private Random rnd = new Random();
public  String data_url = new String("http://norveda.txt");
private String data_url_old = new String();
```

Figure 2.1. A fragment of the knapsack program.

Running GMJ1

Figure 2.2 (top) shows the input page.
On $Property$ fields,

$TotalWeight$ is the weight limit, from 10 to 10.000,

$URL\ of\ data\ file$ is the URL address.

This data file is on the same server as the applet. If the data is not correct, the corresponding field turns red. In the black-and-white figure 2.2, red is shown as the dark shadow. Therefore, the incorrect URL address is not legible.

On the $Dimension$ fields,

Min is the minimal value of $x(l)$,

Max is the maximal value of $x(l)$,

$Default$ is the default value of $x(l)$.

The values $x(l)$ show the proportions of each method. The mixture of three methods of picking objects is considered: the Monte Carlo, the Linearly Randomized Heuristics and the pure Greedy Heuristics. Probabilities x_l of methods $l,\ l = 1, 2, ...$ are related to the proportions $x(l)$ this way $x_l = x(l)/\sum_j x(j)$. Figure 2.2 (bottom) shows the output page that opens when the computing is

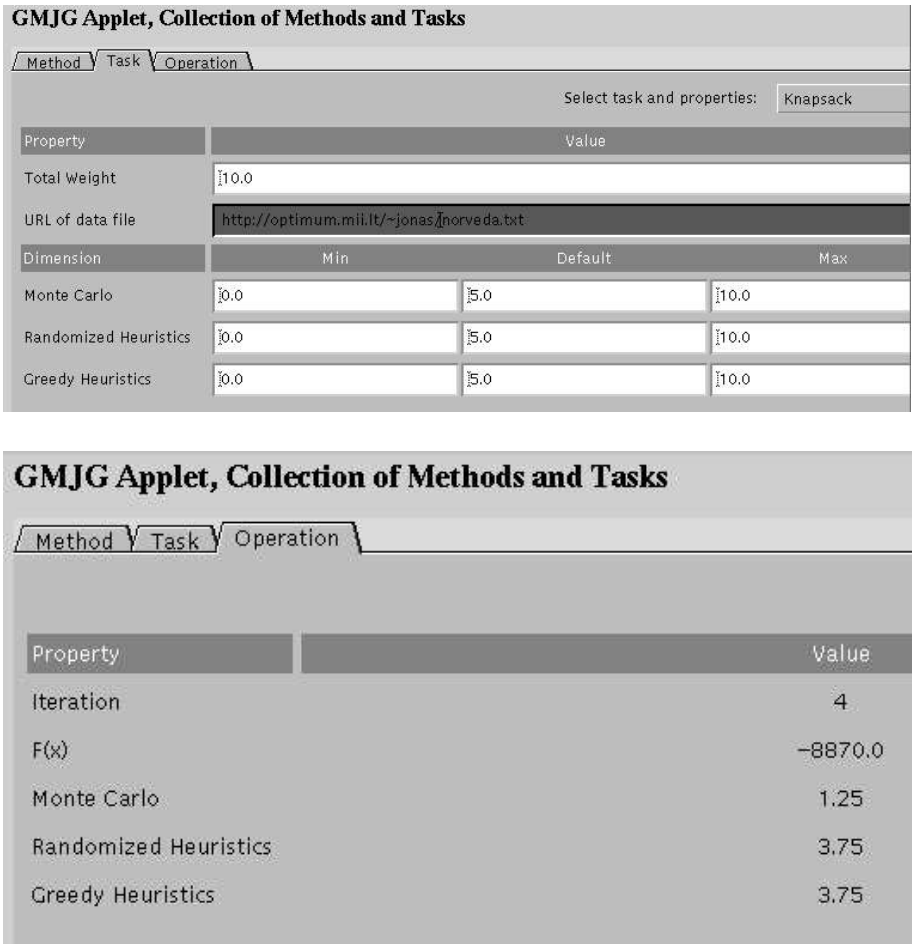


Figure 2.2. Input page (top) and output page (bottom).

completed. Here $Iteration$ means the iteration number where the best value was reached. $F(x)$ defines the best value, "Monte Carlo." "Randomized Heuristics," and "Greedy Heuristics" show the optimal part of each of the three methods in the mixture.

Figure 2.3 (top) shows how the best value of $f_K(x)$ changes subject to the iteration number.

Figure 2.3 (bottom) shows how $f_K(x)$ changes subject to proportions of the Monte Carlo randomization.

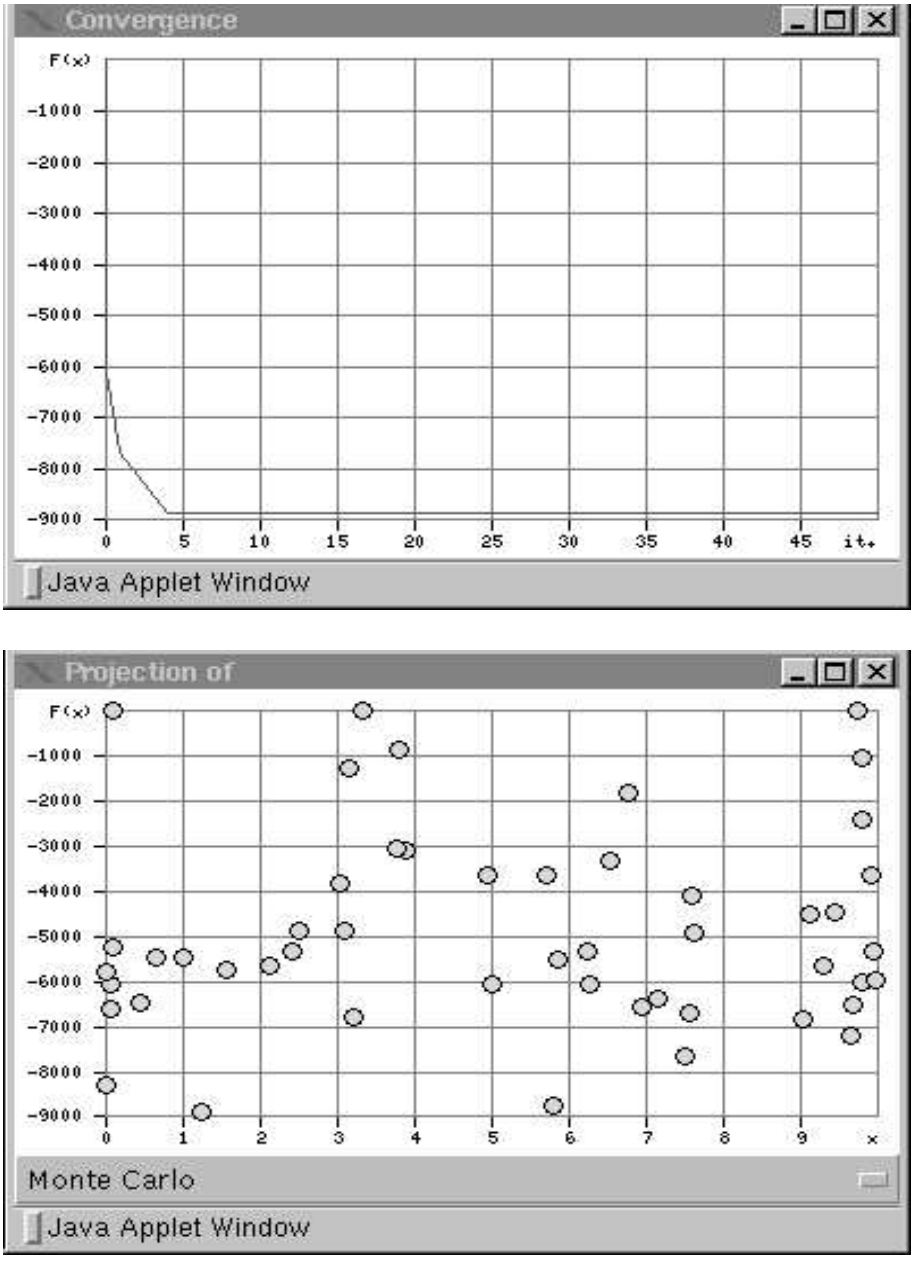


Figure 2.3. The best obtained value subject to iteration numbers (top), the objective function subject to proportions of the Monte Carlo randomization (bottom).

Figure 2.4 (top) shows how $f_K(x)$ changes subject to proportions of the linear randomization.

Figure 2.4 (bottom) shows how $f_K(x)$ changes subject to proportions of the pure greedy heuristics.

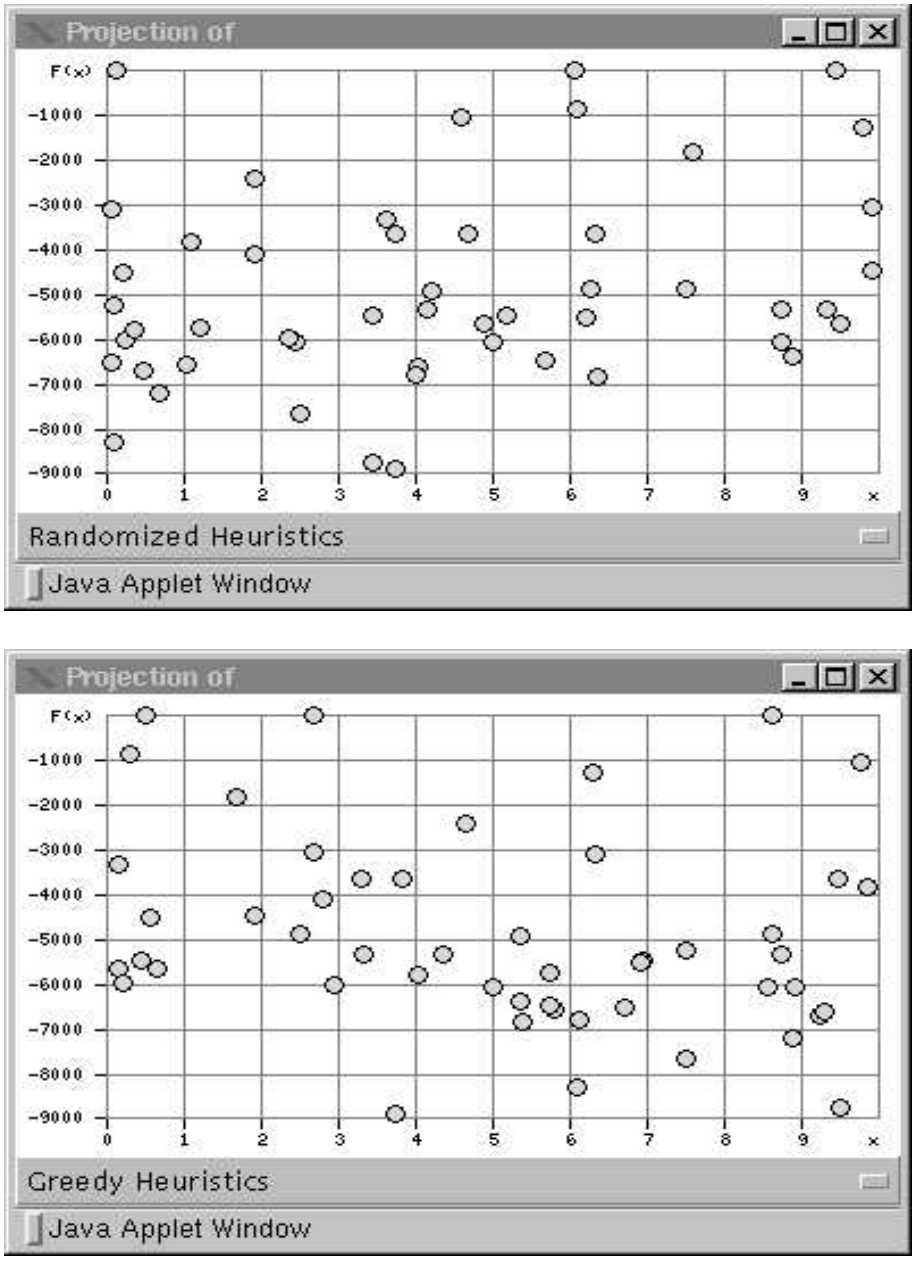


Figure 2.4. The objective function subject to proportions of the linear randomization (top), the objective function subject to proportions of the pure greedy heuristics (bottom).

It is hard to notice any regularity in the $Projection$ windows in Figures 2.3 (bottom), and 2.4. The reason is that all the variables change together during the optimization. To see good projections, one uses the method $Exkor$. Variables change one by one in this method (see Figures 9.4 and 9.5).

Results

Method	Iteation number	Maximal weight	Best iteation	Best function	Monte Carlo	Randomized Heuristics	Greedy Heuristics
Migl	1000	100	535	-66790	1,23	9,297	1,641
			854	-65920	2,576	9,548	8,733
			907	-65660	1,605	9,211	7,882
		1000	932	-488650	0,038	5,4	2,059
			735	-495230	0,047	3,296	0,445
			342	-488710	0,367	9,478	2,953
	500	100	152	-68940	1,237	9,533	6,913
			276	-66340	0,419	9,482	4,172
			269	-70360	4,334	9,528	5,683
		1000	463	-493640	0,103	7,129	1,504
			201	-487240	1,205	5,844	0,093
			439	-483170	2,542	6,819	0,634
Bayes	1000	100	851	-67640	2,731	8,574	9,539
			514	-65090	0,082	5,091	8,761
			595	-67490	1,265	9,508	2,819
		1000	741	-498850	0,473	9,947	0,644
			906	-504300	0,008	7,536	3,259
			949	-505170	0,073	8,612	0,198
	500	100	408	-65850	5,089	7,438	6,707
			213	-66890	0,018	8,213	5,339
			438	-66340	1,244	8,521	8,446
		1000	253	-495000	0,373	8,679	4,233
			37	-499860	0,417	5,745	0,239
			187	-502260	0,012	7,656	1,138

Figure 2.5. Table of results

Table 2.5 shows how the results of optimization depend on the optimization method (the first column), the number of iterations (the second column), and the weight limit (the third column). The fourth column shows the iteration number where the best value was obtained, the fifth column shows the best value[8]. The sixth, the seventh, and the eighth column define the optimal mixture of three search methods. These mixtures can be expressed in percentage terms dividing each of the three numbers by their sum and multiplying by 100.

The results suggest that

- doubling the iteration number (from 500 to 1000) we lower the average error just by 3%,
- if the weight limit g is high, the greedy heuristic is almost useless (the optimal mixture $x = (7/15, 7/15, 1/15)$ meaning that the greedy heuristics part $x(\infty)$ is just 1/15,
- if the weight limit g is low, the greedy heuristic is essential (the optimal mixture $x = (1/6, 1/6, 4/6)$ because the greedy heuristics part is ten times greater.

Now we consider some related problems. In these problems there are no obvious heuristics. Therefore, designing BHA algorithms one should consult experts. The experts may define decision rules defining greedy heuristics or suggest initial solutions needed for permutation heuristics.

[8]The sign is minus, because by default GMJ reduces the objective of the Task object.

4. Related Problems

4.1 Collecting Trains

The objective is to collect the minimal number of trains to remove m cars from a goods-station. Optimization variables are y_{ij}. Here $i = 1, ..., m$ denotes a car and $j = 1, ..., n$ denotes a train. The Boolean variable $y_{ij} = 1$, if the car i is assigned to the train j. Otherwise $y_{ijk} = 0$.

The constraints are

$$\sum_{i=1}^{m} a_i y_{ij} \leq A_j \tag{2.15}$$

$$\sum_{i=1}^{m} b_i y_{ij} \leq B_j \tag{2.16}$$

Here
a_i is the weight of car i.
A_j is the maximal weight of train j.
b_i is length of car i.
B_j is maximal length of train j.

The objective function is the number of trains

$$v(y) = \sum_{ij} y_{ij} \tag{2.17}$$

4.2 Optimal Cut

Formalization

The task is to cut a rod of length a into m pieces of length $a_i,\ i = 1, ..., m$ with minimal waist v. Optimization variables are y_{ik}.
Here
$i = 1, ..., m$ denotes a segment.
$k = 1, ..., K$ defines an ordinal number of the segment.

The Boolean variable $y_{ik} = 1$, if the segment i is assigned to the rod with ordinal number k.
Otherwise, $y_{i,k} = 0$.
The constraint is

$$\sum_{ik} a_i y_{ik} \leq a. \tag{2.18}$$

Here
a_i is length of segment i including the saw section.
a is length of rod. No gaps between segments.

The objective function is the waist. The waist $v(y)$ is defined as the difference between the length of the rod and the sum $a(y)$ of lengths of all the segments

cut,
$v(y) = a - a(y)$. One needs the optimization, if the sum of lengths of all segments to be cut $A > a$.

$$\begin{aligned} \min_y \ v(y) \\ y = (y_{ik},\ i = 1, ..., m,\ k = 1, ..., K), \\ a - a(y) \geq 0. \end{aligned} \tag{2.19}$$

Multi-Road Case

If there are $n > 1$ rods of length $a(j),\ j = 1, ..., n$ to be cut into segments then one minimizes the total waist

$$\begin{aligned} \min_y \ v(y) \\ y = (y_{ijk},\ i = 1, ..., m,\ j = 1, ..., n,\ k = 1, ..., K), \\ a(y(j)) \leq a(j). \end{aligned} \tag{2.20}$$

Here
$v(y) = \sum_j (a(j) - a(y(j)))$,
$y(j) = (y_{ijk},\ i = 1, ..., m,\ k = 1, ..., K)$,
$a(y(j)) = \sum_{i,k} a_i y_{ijk},\ j = 1, ...n$,
index $i = 1, ..., m$ is denotes a segment.
index $j = 1, ..., n$ denotes a rod,
index $k = 1, ..., K$ defines an ordinal number of the segment i in the rod j.

The Boolean variable $y_{ijk} = 1$, if the segment i is assigned to the rod j with ordinal number k.
Otherwise $y_{ijk} = 0$.
The optimization is needed if $\sum_i a_i > \sum_j a(j)$.

Solutions

The exact solutions are practical only for small problems. For larger problems one can apply all the heuristic methods described in the knapsack section 2. Only heuristics of the optimal cut problem are different from those used in the knapsack problems and should be defined by experts. The trivial heuristic h_i is the length a_i of a segment i. Using randomization techniques this means assigning higher probabilities to segments of greater length.

4.3 Sawmill Problem

The task is to produce m boards with minimal waist. There are n timbers. Optimization variables are y_{ijk}.
Here
$i = 1, ..., m$ is denotes a board.

$j = 1, ..., n$ denotes a timber.
$k = 1, ..., K$ defines an ordinal number of the board i in the timber j.

The Boolean variable $y_{ijk} = 1$,
if the board i is assigned to the timber j with ordinal number k.
Otherwise $y_{ijk} = 0$.
The constraints are

$$a_i y_{ijk} \leq d_{ijk}. \tag{2.21}$$

Here
a_i is width of board i including the saw section.
d_{ijk} is diameter of timber i at the location k of board i.
That means, d_{ijk} depends on the order and of boards in the timber j and their thickness b_i. No gaps between boards.

The objective function is the waist
$v(y),\ y = (y_{ijk},\ i = 1, ..., m,\ j = 1, ..., n,\ k = 1, ..., K.$ The waist $v(y)$ is defined as the difference between the cross-section area of all timbers used and the sawn section of all boards produc Expression (2.21) defines constraints for the simple composition of timbers (see Figure 2.6) For more complicated compositions, for example, the segment one (see Figure 2.7), different constraints should be defined.

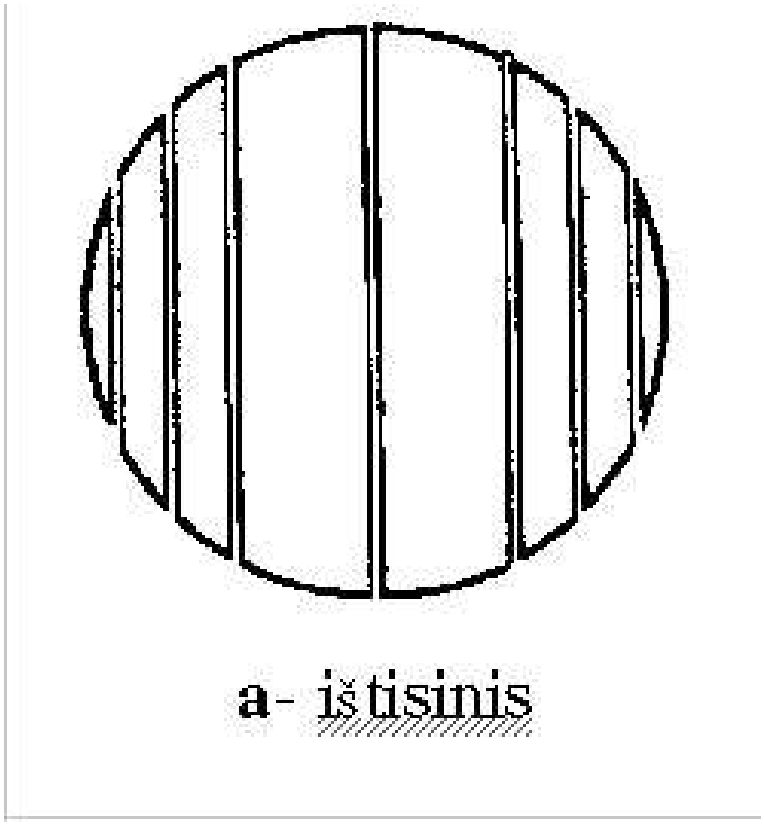


Figure 2.6. Simple composition

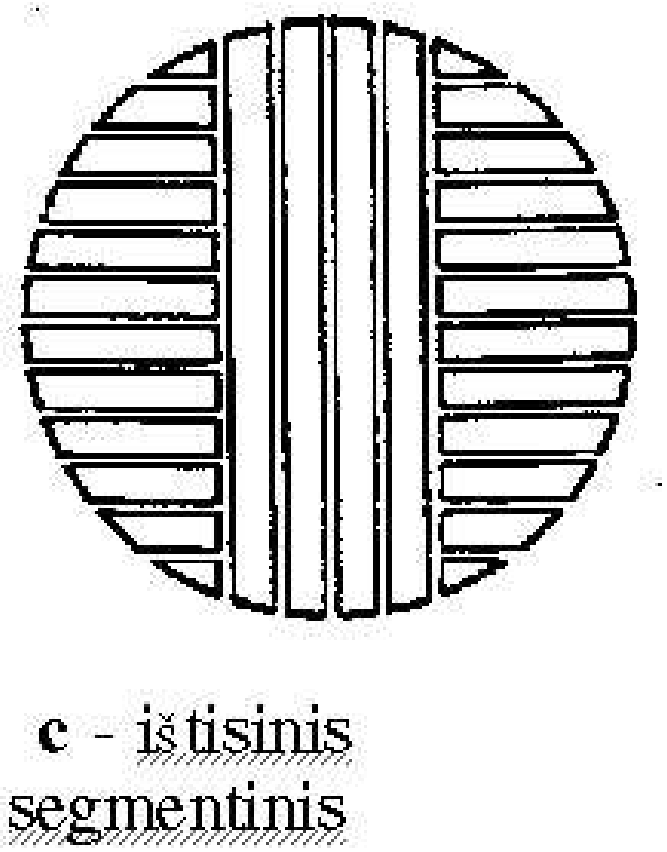


Figure 2.7. Segment composition

4.4 Optimal Diet

As usual, the food is sold in supermarkets and meals are served in restaurants only in some units. For example one cannot buy one third of bottle of milk or order two thirds of hamburger. In this case solving the optimal diet problem one faces difficulties similar to those in the knapsack problem. The reason is that some of optimization variables are integer including the binary ones that we shall regard separately.

Formal Description

One minimizes the extended cost[9] while keeping some health constraints

$$\min_y (\sum_{i=1}^{n} (c_i - s_i) y_i + L(\sum_{i=1}^{n} g_{i1} y_i - g_1) \quad (2.22)$$

$$\sum_{i=1}^{n} g_{ij} y_i \geq g_j, \; j = 1, ...m \quad (2.23)$$

$$y_i \in \{0, 1\}, \; i \leq n_{bin}, \; y_i \in \{0, 1, 2, ...\}, \; n_{bin} < i \leq n_{int}, \; y_i \geq 0. \quad (2.24)$$

Here
index i denotes an item of diet,
c_i denotes the unit price,
y_i is the number of units,
s_i is the amount of pleasure, obtained while consuming a unit of the food i and estimated as some benefit in money terms,
g_{i1} is the amount of calories in the item i,
$g_{ij}, \; j > 1$ is the amount of some basic food ingredients j, such as proteins, hydrocarbons, fats, etc., contained in a unit of item i,
g_j defines amounts needed to keep healthy,
$L > 0$ the unpleasant results due to excess calories estimated ass some losses in money terms.

Solution Methods

Exact Algorithms. The exact solution of the diet problem (2.22-(2.24) can be obtained by the Mixed Integer Linear Programming (MILP). The free MILP software in Java is in the web-sites (see section "Software Systems", under the title "LPJ"). However, one observes the exponential growth of time depending on the number of integer variables, mainly the binary ones. Therefore, one prefer approximate methods, if the number of those variables is large.

[9] Including the direct cost and indirect benefits and losses.

Continuous Approximation. This means approximating integer variables y_i by continuous ones. This way the Mixed Integer Linear Programming (MILP) problem is reduced to the simple Linear Programming (LP). The disadvantages are large round-off errors. The rounding uncertainty of binary variables makes LP solution almost irrelevant.

Heuristics. First we separate the easy part of the diet problem from the hard one. The original MILP problem (2.22-(2.24) is replaced by a set of LP problems corresponding to all possible combinations of integer variables represented by the vector
$d = (y_i \in \{0,1\},\ i \le n_{bin},\ y_i \in \{0,1,2,...,K\},\ n_{bin} < i \le n_{int})$.

$$y(d) = arg\ \min_y (\sum_{i=1}^{n}(c_i - s_i)y_i + L(\sum_{i=1}^{n} g_{i1}y_i - g_1)) \tag{2.25}$$

$$\sum_{i=1}^{n} g_{ij}y_i \ge g_j,\ j = 1,...m,\ y_i \ge 0, \tag{2.26}$$

$$\tag{2.27}$$

Theoretically, one obtains the exact solution by comparing all possible values of the discrete vector d.

$$\min_d y(d) \tag{2.28}$$

This is not practical, except for very small numbers n_{bin} and n_{int} because the number of different d is
$N_d = 2^{n_{bin}} + K^{n_{int}-n_{bin}}$.

The practical advantage of the formulation (2.25)-(2.28) is that using this formulation one can directly apply all the heuristic methods described in the knapsack example.

The specific structure of the diet problem is exploited by defining the greedy heuristics or by setting a proper initial diet to be improved later by permutations. The second way seems convenient for the diet problem, because one just improves the diet that a person already likes. By using greedy heuristics one designs new diets from scratch. In all the cases the BHA helps to optimize the heuristic parameters (see the section 2).

Software Example

Only continuous variables are optimized. That is appropriate while considering only the basic food items such as bread, milk, butter, juice, etc. The diet taste is included indirectly, first, by selecting the food list, then, by choosing the most favored item.

The Figure 2.8 shows the input data and the results of optimization. The

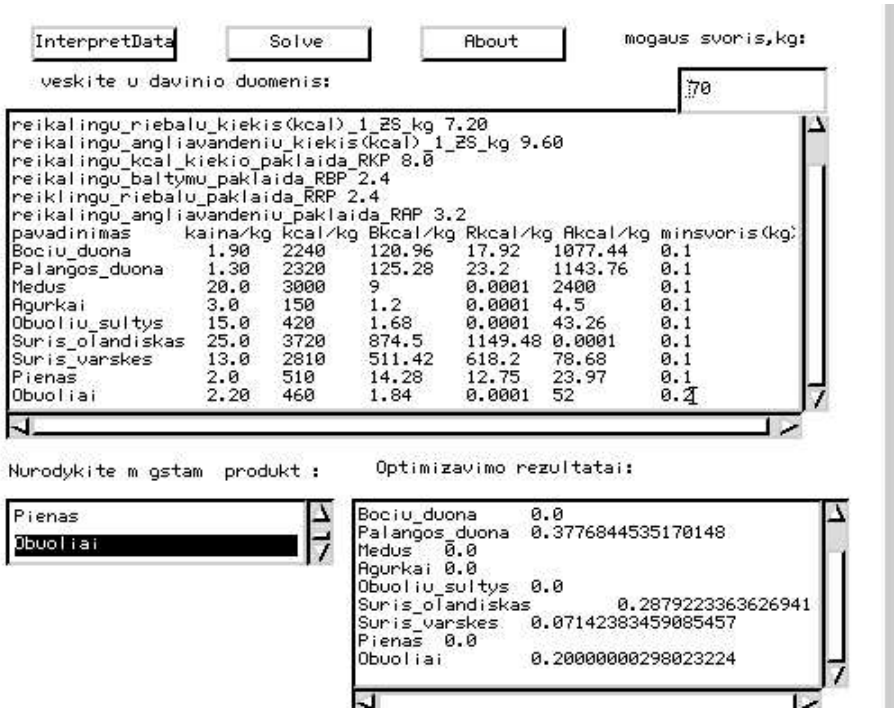


Figure 2.8. Input and Output of the Diet Problem.

selected list reflects the vegetarian taste. The most favored item are apples.

The upper window shows the selected list and related data. The lower-right window shows the optimization results.

Note, that there are only four non-zero items because there are only four constraints defining lower limits for calories, proteins, fats and carbohydrates. This is a specific property of the linear programming solutions.

5. Travelling Salesman Problem

The problem is to minimize the total path length of a salesman visiting I cities and returning home (see(Miller and Pekny, 1991)).

Formally, a decision d is a sequence of numbers $n_i \in 1, ..., I$ that defines the sequence of visits to diferent cities
$d = (n_i,\ i = 1, ..., I-1)$.

A decision is feasible $d \in D$ if there are no gaps in the path. The feasible decision is optimal if

$$\min_{d \in D}\ v(d), \tag{2.29}$$

where $v(d)$ is the length of path d.

$$v(d) = \sum_{i=1}^{I} c_i \tag{2.30}$$

$$c_i = ||z(i+1) - z(i)||,\ i = 1, ..., I-1,\ c_I = ||z(I) - z(1)||$$

Here the vector $z(i) = z_{n_i}$ defines coordinates of the city number n_i. In the two-dimensional space $z_{n_i} = (x_{n_i}, y_{n_i}),\ x_{n_i} \in R,\ y_{n_i} \in R$. The distances between the cities are defined as Euclidean distances between the corresponding points z_i.

In the following numerical experiments "cities" are considered to be points z_{n_i} in the ten-dimensional unit cube (Mockus et al., 1997). The multi-dimensional case is convenient for comparison of different methods. The reason is that the choice of alternative paths is wider here.

Besides, in some real life traveling salesman problems, more then two dimensions are regarded. For example, if a global traveling salesman is sensitive to the time lag then the third dimension is time. If a product passes through many separate reactors[10] then one minimizes differences of some parameters between adjacent reactors, in addition to geometric coordinates.

5.1 Greedy Heuristics

Consider the heuristics

$$h_i = -c_i \tag{2.31}$$

Here $-c_i$ is with the minus sign, because we regard greater heuristics as the better ones. The greedy decision is to visit nearest new[11] city

$$i(g) = arg \max_{j>i} h_j,\ i = 1, ..., I-1 \tag{2.32}$$

[10] Assuming that the sequence of reactors is not uniquely determined by the technological processes.
[11] New means that the city is not visited yet.

It is well known that the "nearest city" heuristic provides nearly optimal solutions, as ususal. Thus, this heuristic should be the main component of any efficient heuristic algorithm. However, to provide for convergence conditions (Mockus et al., 1997), some randomization should be introduced, too.

Considering the knapsack example the compromise between the efficiecy of pure greedy heuristics and the convergence conditions is provided by adding deterministic component (2.34) representing greedy heuristics to the stochatsic components represented as polynomial (2.33).

$$r_i^l = \frac{h_i^l}{\sum_{j=1}^n h_j^l},\ l = 0, 1, 2, ..., L, \tag{2.33}$$

and

$$r_i^\infty(m) \quad = \quad \begin{cases} 1, & \text{if } h_i = \max_j h_j, \\ 0, & \text{otherwise.} \end{cases} \tag{2.34}$$

Here the superscript $l = 0$ denotes the Monte-Carlo randomization. The superscripts $l = 1,\ l = 2, ..., l = L$ define a family of "polynomial" randomizations. The superscript ∞ denotes the greedy heuristics with no randomization.

In the traveling salesman example considered in (Mockus et al., 1997) the similar objective was approached by the "sharp" polynomial expression of probabilities

$$r_i^l = \frac{h_i^{Sl}}{\sum_{j=1}^n h_j^{Sl}},\ l = 0, 1, 2, ..., L, \tag{2.35}$$

where S is large.

Using the Bayesian Heuristic Approach (Mockus et al., 1997) the randomization functions $r_i^l,\ l = 0, 1, 2, ..., L, \infty$ are aplied with probabilities $x(l)$. The best results $f_K(x)$ obtained during K iterations are recorded.

The minimum of function $f_K(x)$ is achieved optimizing the "mixture" x of different randomization techniques r_i. The optimization of the mixture x is difficult because $f_K(x)$ is a stochastic function and the multimodal one, as usual. That is a reason why the Bayesian methods are applied to minimize $f_K(x)$.

In the numerical examples (Mockus et al., 1997) only three terms were considered $L = 2$. Best results were obtained using number $S = 32$. In this case, the probability of going to some distant city is almost zero. However, even very small deviation from the pure heuristics is significant. For example, the probability of not going to the nearest city (in a problem of 100 cities) is about 1% in one iteration and about 27% in all 100 iterations. One may consider these sparse "deviations" from the pure heuristic decisions as some new "initial" points preventing the heuristics to be trapped. This is a way to provide the convergence.

The number of cities I is from 100 to 500. I points (representing cities) are generated 300 times, by sampling from a uniform distribution in the 10-dimensional unit cube.

Usually the exact optimum is not obtained, because the number of problems (5×300) and the size of the problems (from 100 to 500 cities) is too large. Consequently we merely compare the Bayesian methods with the pure heuristics. Both a simple sequential nearest city algorithm and a more complicated local permutation algorithm are considered.

5.2 Permutation Heuristics

For purposes of the algorithm involving local permutations, some initial travel path must be used. One tries to improve this initial fixed path by selecting a pair i of adjacent cities $n_i - n_{i+1}$ and afterwards considering another pair j of adjacent cities $n_j - n_{j+1}$. The pairs are chosen so that reconnecting n_i to n_j and n_{i+1} to n_{j+1} we still obtain a path that visits all cities. We seek such a new path that is shorter. The initial path is selected by a greedy heuristic, then we repeat I times the following two steps:

- choose the first pair of cities at random;
- consider the remaining pairs in succession as long as a shorter path d_l is found, .

The local permutation heuristics $h_l = -v(d_l)$ where $v(d_l)$ is the total length of travelling salesman path corresponding to the permutation l.

5.3 Calculation Results

Table 2.3 illustrates the results of Bayesian algorithms employing simple greedy heuristics by selecting nearest neighbor. The results of pure greedy heuristics and that of pure local permutation are shown, too, for comparison. In

Table 2.3. Results of the Bayesian method using greedy heuristics.

I	S	f_G	f_B	f_{GL}	f_{G-B}	d_{G-B}
100	64	78.90	77.62	77	1.28	0.39
200	128	144.88	143.30	142	1.58	0.71
300	128	205.90	203.95	202	1.95	0.85
400	128	264.62	262.63	260	1.99	1.05
500	128	321.35	319.10	316	2.25	1.32

this table the letter B stands for the Bayesian method, the letter G denotes pure greedy heuristics, and the symbol GL denotes pure local permutations. The

table provides sample means f and sample variances d. Thus f_{G-B} denotes the mean of the difference between the results of a greedy heuristic and the Bayesian methods, and d_{G-B} denotes the corresponding variance. The symbol I stands for the number of cities, and the letter S is the power of the "sharp" polynomial (see expression (2.35). The Bayesian method performs 46 observations.

The results show that the Bayesian method is roughly 1% better than the pure greedy heuristic. The improvement declines with the size of the problem, from 1.6% for $I = 100$ to 0.7% for $I = 500$. Comparing the columns f_{GL} and f_{B-GL} we see the advantage of local permutation heuristics. Therefore, the Bayesian method is applied here, too. Table 2.4 demonstrates the results of Bayesian algorithms for the case of local permutation heuristics. The results of pure greedy heuristics and that of pure local permutation are shown, too. In Ta-

Table 2.4. Results of the Bayesian method using permutation heuristics.

I	S	f_G	f_{GL}	f_{BL}	f_{G-BL}	d_{G-BL}	f_{GL-BL}
100	32	78.7	77	76	2.7	0.6	1
200	32	144.5	142	140	4.5	0.7	2
300	32	206.1	202	200	6.1	1.2	2
400	32	265.3	260	258	7.3	1.4	2
500	32	321.5	316	313	8.5	1.5	3

ble 2.4 the symbol BL stands for the Bayesian method, the symbol GL denotes a pure local permutation heuristic. The table provides sample means f and sample variances d. Thus, the symbol f_{GL-BL} denotes the average difference between the pure local permutation heuristic and the Bayesian method. The symbol f_{G-BL} denotes the average difference between greedy heuristics and the Bayesian method using local permutations. The symbol d_{G-BL} denotes the corresponding variance. Here the Bayesian method performed 26 observations and the algorithm stops after 50 repetitions.

Sharp polynomial randomization (2.35) was chosen as a result of some additional experimentation. When the uniform distribution was included, the average value of f_{G-BL} was 1.5 for $I = 100$ (one hundred cities). Using expression (2.35) with $S = 1$, the average value of f_{G-BL} was 2.5 for the same $I = 100$. The best average gain was achieved with $S = 32$ (see Table 2.4). The distribution of coefficients x_i from expression (2.35) shows which powers of h_i are most important in calculating $r_i^0(m)$.

The results of Table 2.4 show that the Bayesian method is approximately 1% better than the pure heuristics (see the average gain f_{GL-BL}). The improvement is almost independent of the size of the problem. We consider 1% to be a good

result, because this improvement is obtained near the global minimum, where even a fraction of percent is important.

5.4 Software Example 1

Here four algorithms are implemented in Java 1.1:
the author calls them the $nearestneighbor$, the $repeatednearestneighbor$, the $twooptimal$, and the $twooptimal(random)$. Figures 2.9 show the nearest neighbor and the repeated nearest neighbor examples. The repeated nearest neighbor starts from random starting points. Two optimal is a version of the permutation algorithm described in the section 5.2. Two optimal (random) uses the results of nearest neighbor algorithm as the starting path. Figures 2.10 show the two optimal and random two optimal examples.

5.5 Software Example 2

In this example two algorithms are regarded:
the $nearestneighbor$ and the $bubble$. Figures 2.11 show the nearest neighbor and the bubble examples. The bubble algorithm is similar to nearest neighbor, however the bubble starts from three connected cities.

5.6 Nearest Tasks

The first task is to implement in Java the Bayesian Heuristic Algorithm (BHA) described in the section 3.2 and extend it to permutation heuristic. Using the permutation heuristics the Bayesian version of Simulated Annealing (see section 2.4) seems to be an interesting alternative to the BHA. To perform the statistical comparison of these algorithms one prefers Java1.2 implementation because it works about ten times faster in similar problems.

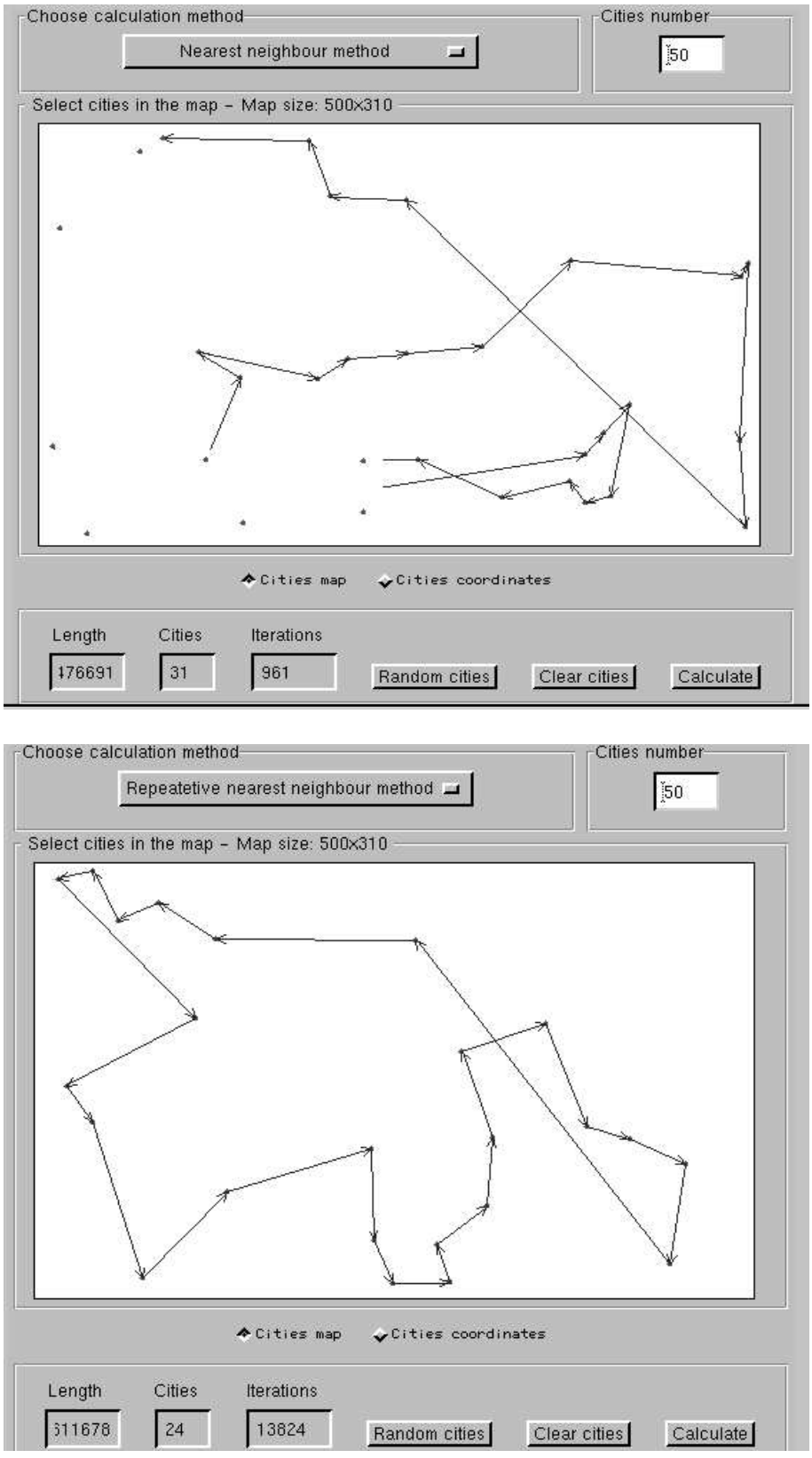


Figure 2.9. The nearest neighbor (top) and the repeated nearest neighbor (bottom) examples .

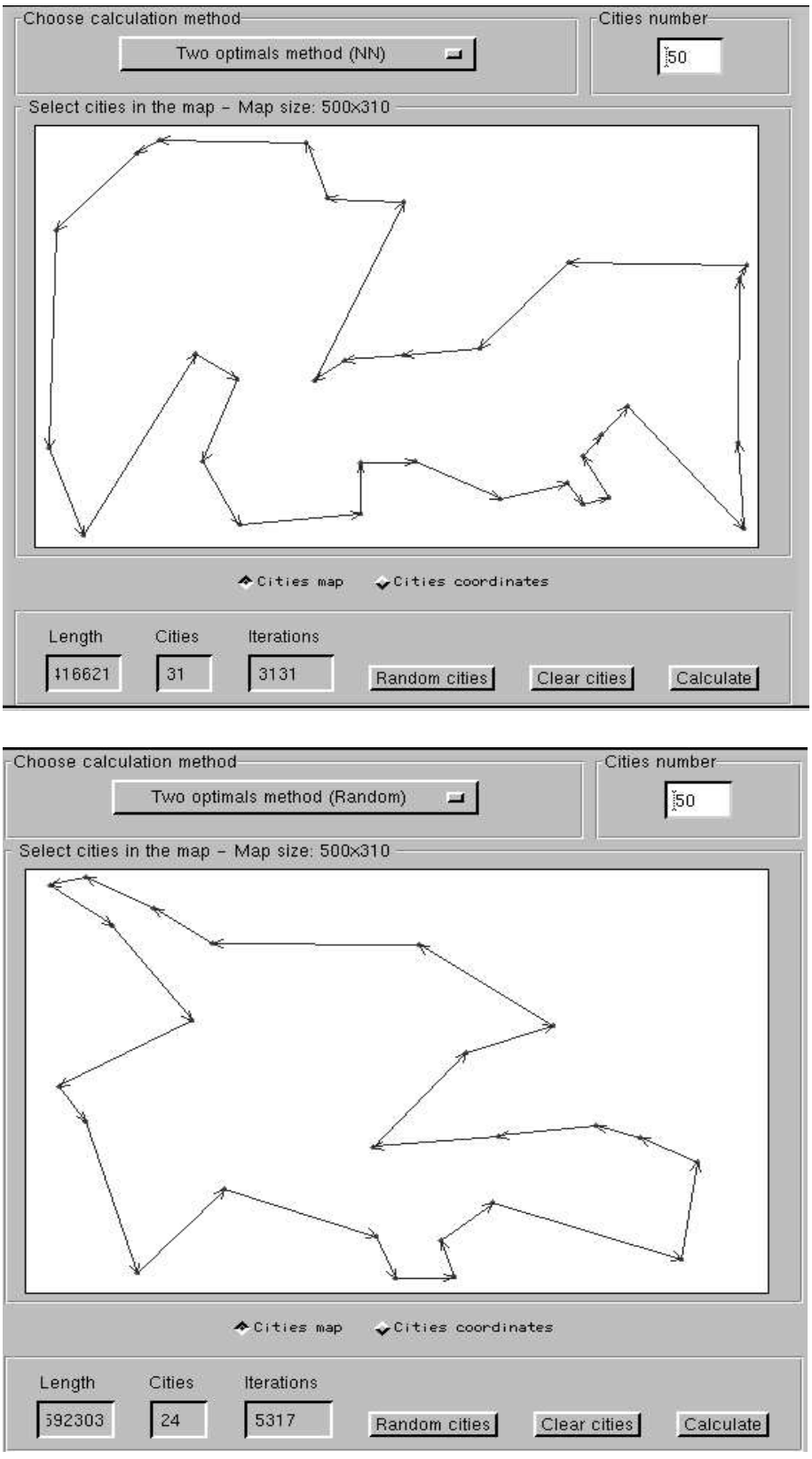


Figure 2.10. The two optimal (top) and random two optimal (bottom) examples .

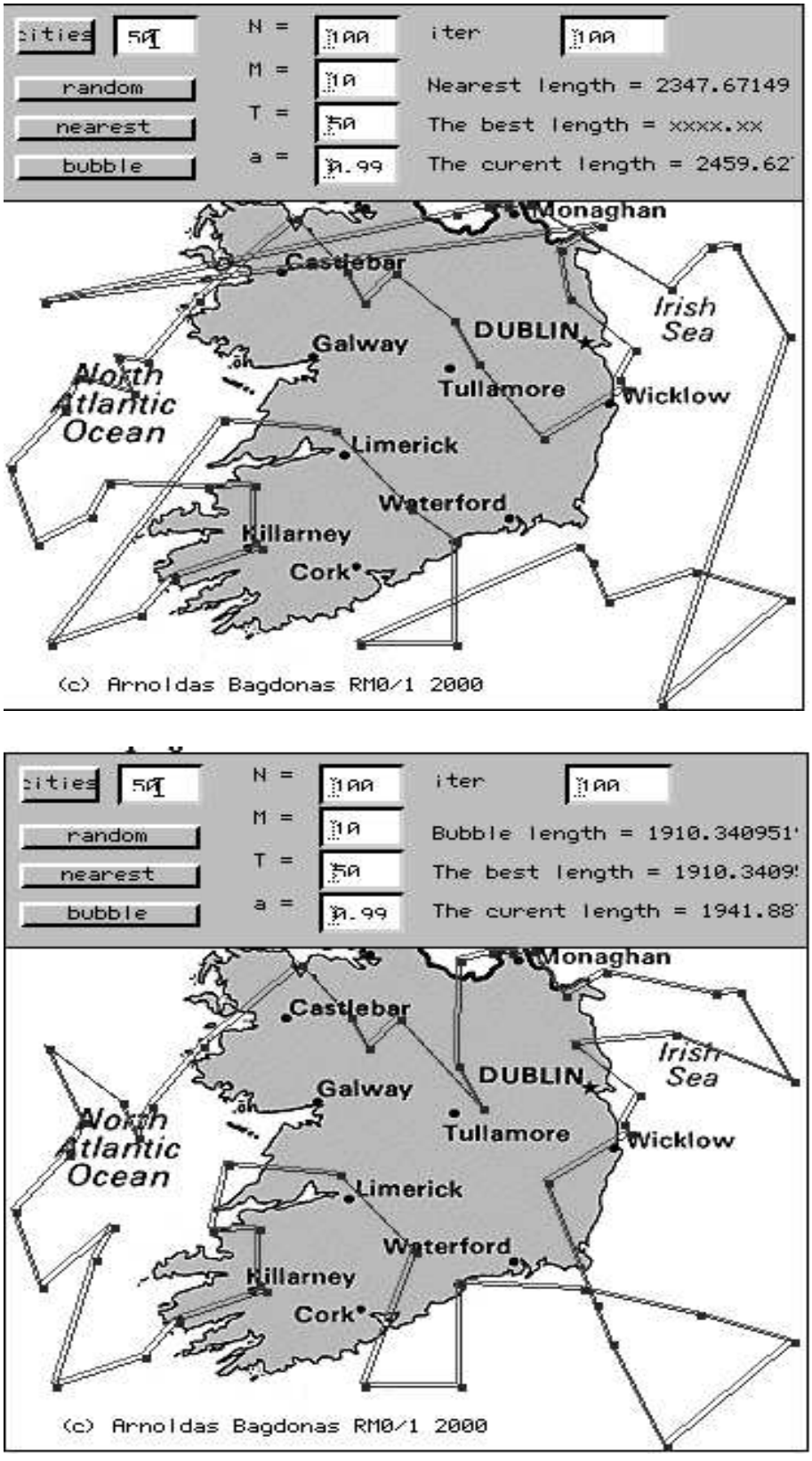


Figure 2.11. The nearest neighbor (top) and the bubble (bottom) examples .

II

SOFTWARE FOR GLOBAL OPTIMIZATION

Chapter 3

INTRODUCTION TO SOFTWARE

1. General Description

1.1 Background

The software system Global Minimizer (GM) was initiated in early eighties. An initial set of algorithms was selected considering results of the international "competition" of different global optimization methods (Dixon and Szego, 1978). The experience in real life optimization problems was used updating the set. The set of global optimization algorithms includes

- four versions of the Bayesian search,
- a version of clustering,
- a version of the uniform deterministic grid,
- a version of the pure Monte Carlo search.

As usual, the global optimization is concuded by some local method. Exceptions are two global algorithms: the Torn version of clustering (Torn and Zilinskas, 1989), and the Zilinskas version of the Bayesian technique (Torn and Zilinskas, 1989). Both these algorithms contain some simple local search algorithms. The additional local search is not necessary for those two methods, but it may be useful.

There are three local optimization methods:

- a method of sequential quadratic programming for constrained optimization of smooth functions (Schittkowski, 1985),
- a simplex type method of Nelder and Mead with penalty functions for constrained optimization of non-differentiable functions (Himmelblau, 1972),

- a method of stochastic approximation type for "noisy" functions (Mockus, 1989a).

1.2 Description of Methods

Global Methods

Bayes1 is the Bayesian method (Mockus et al., 1997).

Region : rectangular.

Objective : continuous (possibly with "noise").

Convergence : to global minimum (in probability, if noisy).

Comments :
for "expensive" objectives using not too many observations.

Mig1 is the Monte Carlo search.

Region : rectangular.

Objective : general.

Convergence : in probability.

Comments :
for inexpensive and irregular objectives using great number of observations.

Unt is the extrapolation type method (Torn and Zilinskas, 1989).

Region : rectangular.

Objective : continuous.

Convergence : to global minimum.

Comments :
for expensive objectives using not too many observations.

Exkor is the Bayesian coordinate line search method (Torn and Zilinskas, 1989).

Region : rectangular.

Objective : continuous.

Convergence : to global minimum on the search line.

Comments :
for approximately "separable" objectives and for preliminary exploration using the projection windows.

Glopt is the clustering method (Torn and Zilinskas, 1989).

Region : rectangular.

Objective : continuous.

Convergence : not provided.

Comments :
works well in many practical problems with a moderate number of local minima.

Lpmin is the uniform deterministic search (Sobolj, 1967, Dzemyda et al., 1984).

Region : rectangular.

Objective : general.

Convergence : to global minimum.

Comments :
for inexpensive objectives using many observations.

Local Methods

Nlp is the non linear programming method (Schittkowski, 1985).

Region : defined by linear and non-linear constraints.

Objective : differentiable.

Convergence : to local minimum.

Flexi is the simplex method (Himmelblau, 1972).

Region : defined by non-linear constraints.

Objective : non differentiable.

Convergence : not provided.

Lbayes is the stochastic approximation method with the Bayesian step size control (Mockus, 1989a).

Region : rectangular.

Objective : continuous with noise.

Convergence : to local minimum, in probability.

1.3 Application Areas

Each subroutine represents a global or a local method. The choice of methods follows the comparability idea. That means that the computational complexity of the method should roughly correspond to that of the objective function:

- The Bayesian methods are recommended for "expensive"[1] functions. Those methods need auxiliary calculations to make each observation more efficient.

- For "cheap" functions, simple grid methods, like the Monte Carlo or the uniform deterministic grid (Sobolj, 1967), can be better. Here observations are not so efficient, but auxiliary calculations are negligible. This explains the relative efficiency of simple methods when optimizing simple functions.

- The clustering techniques (Torn and Zilinskas, 1989) may be the best choice, if we expect the number of local minima to be small.

- A simple Bayesian technique is available (Torn and Zilinskas, 1989) for global optimization of one-dimensional functions.

- There are optimization problems where objective functions can be roughly represented as a sum of components depending on different variables. Here the Bayesian method doing a line search along each coordinate shows good results, as usual. This method globally optimizes one variable at a time by the one-dimensional Bayesian search. Therefore, results depend on the starting point. This is the difference of this method from other methods of global optimization.

  However, a deviation from the global minimum can be made as small as desired by applying a multi-start search from different uniformly distributed starting points. The important advantage are good $projections$. They show how an objective function depends on different variables (see Figures 9.4 and 9.5). The other methods present messy projections (see Figures 2.3 (bottom), and 2.4). The reason is that all the variables change together.

1.4 Constraints

All the global methods optimize in rectangular regions. Therefore, one represents linear and non-linear inequality constraints as penalty functions. The same applies to the local method of stochastic approximation type. In the local methods of simplex and sequential quadratic programming type, the linear and non-linear constraints are defined directly. This is done by constraint subroutines, supplied by users in addition to the objective function.

1.5 Computing Environment

There are several versions of global optimization software

- a portable Fortran library,

[1] In a sense of computing time.

- interactive software for Turbo C compilers and DOS operating systems,
- interactive software for C++ compilers and Unix operating systems,
- three versions of interactive software using Java (for JDK1.0, for JDK1.1, and for JDK1.2).

One may notice a cycle of portability in this sequence of software versions. The sequence is started by the portable Fortran library and is concluded by the Java systems. The systems between those two are more difficult to port. Fortran, Turbo C, and C++ versions are in (Mockus et al., 1997). All the software is on web-sites (see section 4 and Figure 3.1).

Figure 3.1. The Software Systems page.

2. Early Applications

Besides examples of this book, we briefly mention the examples considered in early publications (Mockus, 1989a, Mockus et al., 1997). Many of them are related to the optimization of parameters of mathematical models, represented as

some systems of non-linear differential equations. The objective function $f(x)$ depends on a solution of these equations. Variables x represent the controlled parameters of the system. The following examples illustrate this family of problems.

- Maximization of general yield of differential amplifiers (Mockus, 1989a).
- Optimization of a mechanical system of shock-absorber (Mockus, 1989a).
- Evaluation of parameters of non-linear regression of an immunological model (Mockus, 1989a).

The last example suggests a broad area for applications of global optimization. It is well known that in non-linear regression the square deviation and the likelihood function could be multi-modal, for some data. The number of local minima may be very large, even in simple cases. An example is the evaluation of unknown parameters of ARMA models (Mockus et al., 1997). There are other important families of global optimization problems.

- The engineering design is a large source of difficult global optimization problems. One optimizes parameters of some mathematical models, non-linear, as usual. The optimization of composite laminates (Mockus et al., 1997) serves as an example.
- Many laws of nature could be defined in terms of global optimization. An example is the "Disk" problem: minimization of the potential energy of organic molecules (Mockus et al., 1997).
- Often one cannot describe the behavior of new materials and technologies by mathematical models, because the information and knowledge are not complete. Here one optimizes by changing control variables and observing the results of direct experimentation. An example is the planning of extremal experiments of thermostable polymeric compositions (Mockus et al., 1997).

The Bayesian algorithms of the continuous global optimization are an essential part of BHA. One applies them to optimize the parameters of randomized heuristics while solving various discrete optimization problems (Mockus et al., 1997).

3. Different Versions

3.1 Fortran Library, Portable

The description of the global optimization software starts from the Fortran library (see Chapter 4). In other versions, mostly the same algorithms are used. Their parameters have a similar meaning. Therefore, the Turbo C, C++ and

Java users may obtain some useful information by reading the Fortran library description.

The portable Fortran version can run on any computer with a standard Fortran compiler. Users represent objective functions as
FUNCTION FI(X,N). Here X is an array of variables and N is its dimension. The lower bound array A and the upper bound array B define rectangular constraints. Using local methods of simplex and sequential quadratic programming type, one represents constraints by the subroutine
CONSTR(X,N,G,MC). Here, G is an array of length MC that contains values of constraints at a point X. MC is the number of constraints.

The advantage of the Fortran library is the portability. A disadvantage is limited interactive possibilities. Therefore, this version can be conveniently used for some well-defined optimization problems. For preliminary investigations of new problems, some interaction is essential.

3.2 Turbo C, Intreactive

This version is for objective functions represented in C. It needs a Turbo C compiler. Users represent the objective function $f(x)$ as the C subroutine $fi.c$. Users can select a global or a local method by a menu system. Current results are observed in both, tabular and graphical, forms. There are two graphical forms.

GRAPH : shows how the best value of an objective function depends on observation numbers,

PROJECTION : shows how current objective function values depend on a variable.

The set of methods, in both the Fortran and the Turbo C versions, remains the same, except a non-linear programming method. The Biggs method (Biggs, 1975) is implemented in Fortran to solve non-linear programming problems. In all other languages, the sequential quadratic programming (Schittkowski, 1985) is used for these problems.

Good interactive facilities is an advantage of the Turbo C version. A disadvantage is that this version can be used only in the DOS-compatible environment. Often one prefers Unix systems while solving large scale optimization problems.

3.3 Unix C++, Interactive

This is a global optimization software designed for UNIX and X -Window systems. It includes a usual set of interactive facilities. The main feature is the possibility to port to any environment that supports Unix and X-Window, including super computers and distributed computing systems.

3.4 MS-Windows C#, Interactive

This is a global optimization software in C# designed for Windows98 and later. The graphical interface is similar to that of the Unix C++ version. It can be used as a standalone application, in a way similar to that of the C++ version.

3.5 Java JDK1.0, Interactive

This is a global optimization software in Java, designed for the JDK1.0 tool kit. The graphical interface is similar to that of the Unix C++ version. It can be used as a standalone application, in a way similar to that of the C++ version. The software can be run directly by Internet as an applet, too. Any Internet browser works, if it supports JDK1.0. That includes older versions of Netscape and Internet Explorer. This is an advantage, because these versions are still used.

3.6 Java JDK1.1, Interactive

This is a global optimization software in Java, designed for JDK1.1. The graphical interface is open and can be extended by including new methods, additional tasks and updated result analysis facilities. The software can be used as a standalone application or as an applet by the Internet browsers supporting JDK1.1. That excludes early browsers such as Netscape-3. Rectangular constraints are defined by the graphical interface. Other constraints are involved by adding the penalty functions that are defined by users.

3.7 Java JDK1.1, Interactive

This is a global optimization software in Java, designed for JDK1.1. The graphical interface is open and can be extended by including new methods, additional tasks and updated result analysis facilities. The software can be used as a standalone application or as an applet by the Internet browsers supporting JDK1.1. That excludes early browsers such as Netscape-3. Rectangular constraints are defined by the graphical interface. Other constraints are involved by adding the penalty functions that are defined by users.

3.8 Java JDK1.2, Interactive

This is a new version of global optimization software. It exploits new possibilities of JDK1.2. This Java version works on browsers that support JDK1.2. Otherwise, the JDK1.2 kit should be installed. Then the command

```
appletviewer http://mockus.org/optimum/gmjg2j/gmj.html
```

starts the applet $gmj.html$ in the directory $optimum$ of the server $mockus.org$. This way one bypasses a browser but must write the complete path.

The main new features:

- $constraint$ functions can be involved by all optimization methods and other user defined functions,
- each objective function is accompanied by its own set of visualization algorithms and graphical interfaces, they appear only when this objective function is chosen.

The detail description of different software versions follows. Most versions use the same algorithms. Their parameters are similar. The software is on the web-sites.

4. Web-Sites

Convenient reading at any place is an obvious advantage of hard copy. The theoretical considerations, including descriptions of mathematical models and algorithms, are clear and understandable, on a printed page. The static presentation is a disadvantage. A hard copy cannot be updated. Therefore, new results wait for the next edition.

The aim of the web-sites is to supplement the models and algorithms by interactive software systems. These systems can be tested directly by users in different computing environments, using different languages and operating systems (see Figures 3.1, 3.2, 3.3). The dynamic presentation is the important advantage of web-sites. Results can be easily updated and made accessible for many readers, free of charge. Therefore, the author considers this book as a static introduction to dynamic web-sites. The mirror-sites:

1 $http://mockus.org/optimum$

2 $http://optimum2.mii.lt/\tilde{}jonas2$

3 $http://soften.ktu.lt/\tilde{}mockus$

4 $http://www.vtu.lt/mockus$

The first web-site is designed for the international audience. Only selected items are included in this site. The updates are done by replacing the existing items, only when obviously better results appear. Therefore, updating is less frequent. The site is available always.

The second web-site is experimental. It is updated frequently, and has larger volume. The accompanying text, mostly in English, is in txt, $html$, pdf, or ps files. The site is used by graduate students of Lithuanian universities. Therefore, a part of the text is both in English and in Lithuanian. Sometimes, during holidays this site is closed.

In the future, the web-sites will be regularly updated by new developments. However, the software will remain compatible with the present description.

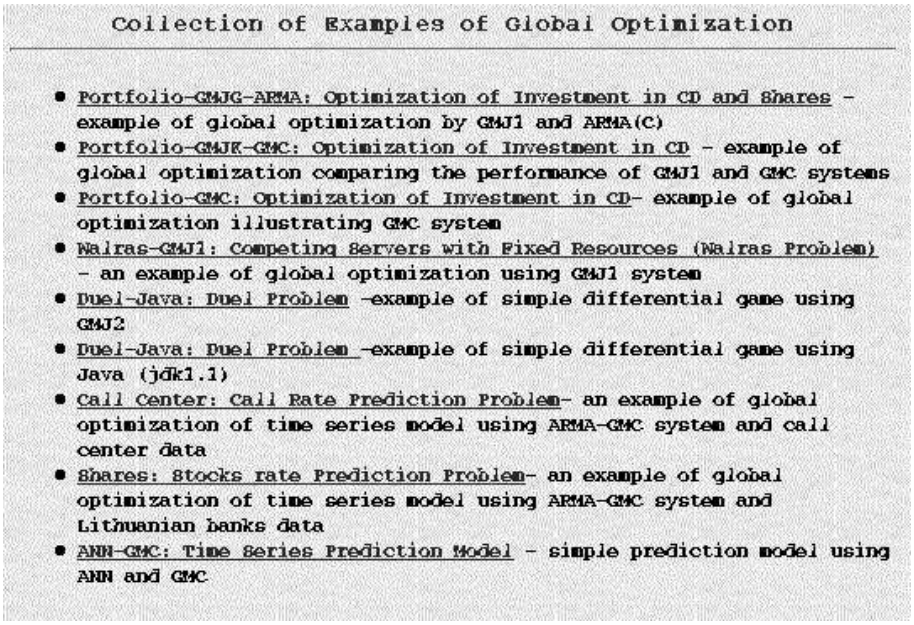


Figure 3.2. Examples of Global Optimization.

Contact e-mail addresses:
$jonas@optimum.mii.lt$
$jonas2@optimum2.mii.lt$
$audris@mockus.org$

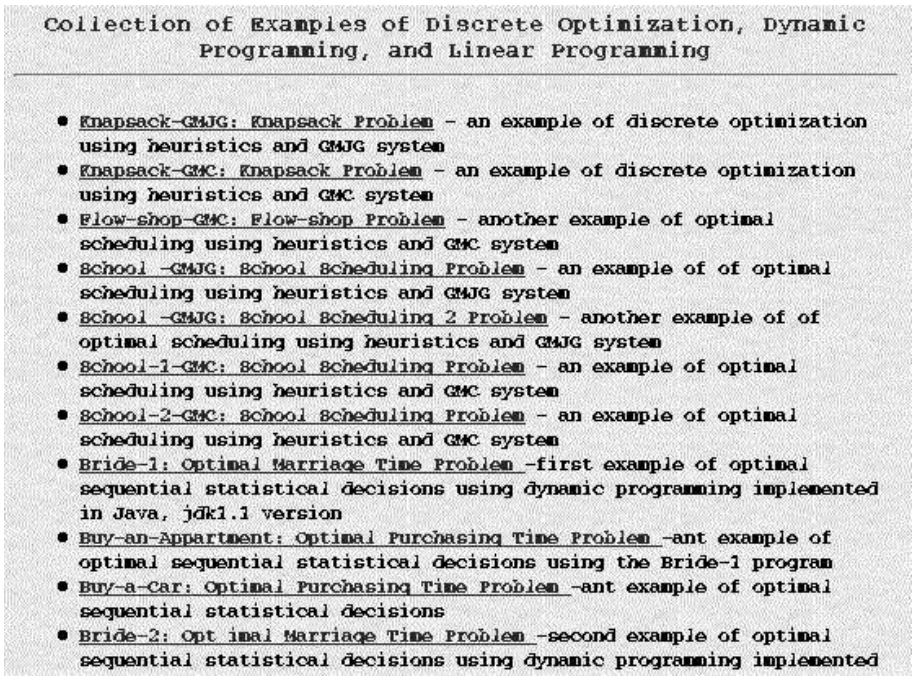


Figure 3.3. Examples of Discrete Optimization, Linear and Dynamic Programming.

Chapter 4

PORTABLE FORTRAN VERSION (GMF)

1. Introduction

The early Global Minimizer was in Fortran (GMF). Therefore, we describe the Fortran version as an introduction to later versions. The GMF system is to minimize a continuous function

$$f(x),\ x = (x_1, \ldots, x_n) \tag{4.1}$$

where $x \in A \subset R^n$ (Mockus et al., 1997). It is assumed for most methods that the set A is a rectangular parallelepiped

$$A = \{z : a_i \leq x_i \leq b_i, \quad i = 2, \ldots, n\}. \tag{4.2}$$

Other constraints are approximately reduced to (4.2) by the penalty function techniques.

A feature of all versions of the Global Minimum software, including GMF, is the Bayesian approach. The Bayesian methods are optimal, in the sense of an average deviation. Here the next observation is defined by minimizing the risk function $R(x)$. The risk is an expected deviation from the global minimum. The minimization of the risk function is justified, if the objective function $f(x)$ needs considerable computing. Otherwise, it could be better to do more observations, which need not to be planned optimally. In such a case, a good idea is to use a uniform search, for example, of LP type (Sobolj, 1967). Using the LP-search we lose some efficiency of observations but still have the convergence. However, the LP-search can be too expensive, if observations are very cheap. Then we apply the uniform random (Monte Carlo) search (Monte Carlo) which is very simple. Using the Monte Carlo we obtain the convergence only in probability. The LP-search (LPMIN) and the uniform random search (MIG1 and MIG2) algorithms. Both these algorithms are in GMF.

The method of clustering (Torn, 1990) is included as a good heuristic technique under the title GLOPT. It usually works well, if the number of local minima is small and known in advance, and if the "attraction area" of the global minimum is large. GLOPT is comparable to LPMIN by the computing time.

The global line search EXKOR is a "semi-global" method designed for approximately "additive" functions. EXKOR is convenient for visualization. One see clearly how values of the objective function depend on variables. That is important for preliminary investigations.

Global methods search the whole area, as usual. Otherwise, one can miss the global minimum. Obviously, the global search is not the most efficient way for local optimization. Therefore, in GM the global search is completed by the local one, as usual. The best result of a global search is defined by default as an initial point for a local search.

Methods of variable metrics type are widely used for local optimization of continuously differentiable functions without noise with non-linear constraints. The local mathod, called Recursive Quadratic Programming (Biggs, 1975), is included under the name REQP[1].

A specific Variable Metrics version for rectangular constraints (Tiesis, 1975) is called MIVAR4. For local optimization of non-differentiable functions with non-linear constraints the simplex type method (Himmelblau, 1972) is used under the name FLEXI.

Sometimes, one expects that the influence of some variables and their groups is considerably greater than that of others. Then the method LPMIN should be used for analysis of the structure before we start the regular optimization. The LPMIN orders variables by their "importance" using a sort of variance analysis techniques (Saltenis, 1989).

LBAYES is a stochastic approximation type algorithm with the "Bayesian" step length control. It is designed for local optimization of unimodal functions with a noise.

In GMF, no machine-dependent routines are used. Therefore, the software can be adapted to any computer with a standard FORTRAN compiler by adjusting standard machine-dependent constants.

2. General Discussion

2.1 Parameters

The parameters X, A, B, N, FM, IPAR, PAR, IPA, IPAA are used in all subroutines. Here
X is an array of length N which defines the coordinates of a point being con-

[1] In later versions a different algorithm (Schittkowski, 1985) is implemented under the name NLP.

sidered (initial, optimal or current),
A is an array of length N which defines the lower bounds of X,
B is an array of length N which defines the upper bounds of X,
N is the number of variables (dimension of X) usually N≤ 20,
FM is the value of the function FI(X, N) at the point X,
IPAR is the array of length 30 which defines the real control parameters,
IPA is a shift of integer control parameters,
IPAA is a shift of real control parameters. If only one method is used, then both shifts are zero: IPA $= 0$ and IPAA $= 0$. If several methods are used sequentially, then a shift for the next method must be equal to the sum of numbers of control parameters used before by other methods. The number of control parameters of different methods is given in Table 4.1. For all the methods the first integer control parameter is the printing parameter: IPAR(IPA+1)=IPR (IPA=0 if only one method is used). If IPR < 0,
then only diagnostic messages are printed.
If IPR $= 0$,
then the initial data, the final results and diagnostic messages are printed.
If IPR > 0,
then not only those but also the results of each IPR-th iteration are printed. The meaning of other control parameters will be explained when describing the corresponding subroutines.

2.2 List of Methods

Table 4.1 shows parmeters to combine methods into different sequences Table

Table 4.1. Table of Parameters.

No	Name	Method	No. of control parameters Integer IPAR	Real PAR
		Global optimization with rectangular constants		
1	BAYES1	Bayesian	3	0
2	UNT	Extrapolation	4	0
3	LPMIN	Uniform deterministic	N+3	0
4	GLOPT	Clustering	3	0
5	MIG1, MIG2	Uniform random	2	0
6	EXTR	Bayesian one-dimensional	3	2
7	EXKOR	Extension of EXTR to multi-dimensional case	5	N+1
		Local optimization		
8	MIVAR4	Variable metric with rectangular constants	4	4
9	REQP	Variable metrics with non-linear constraints	4	4
10	FLEXI	simplex with non-linear constraints	4	2
		Local optimization with noise		
11	LBAYES	stochastic approximation with Bayes step length and rectangular constraints	3	2

4.1 shows that to set the algorithm parameters one performs many adjustments. This task is easy using "ready-made" examples. They include sequences of different algorithms. One selects needed algorithms just by "commenting-out" the redundant ones. For example, the file 'gmfl/ex8.f' in the enclosed disk includes seven algorithms named BAYES1, MIVAR4, LBAYES, FLEXI, REQP, UNT, GLOPT. The file 'gmfl/ex9.f' includes only one algorithm EXKOR. The

software is on web-sites (see section 4). Methods are described in (Mockus et al., 1997) and (Mockus, 1989a).

3. Program Description

3.1 Common Blocks

Values of the function FI(X, N) are in the array /BS1/Y(100). The array of variables X is defined by the array XN of length MN=N*M.
In the array /STATIS/IFAIL, IT, IM, M:
IFAIL is a control indicator:
- if the initial data is not correct, then IFAIL $= 10$ and return to the main program,
- if the initial data is correct, then IFAIL $\neq 10$ and shows the number defining the stopping rule,
IT is the number of iterations,
IM is the number of the optional iterations,
M is the number of function evaluations (observations).

3.2 Objective Functions and Constraints

The function to be minimized should be represented as a real function FI(X,N). In most methods, only the lower and upper bounds are fixed by arrays A and B.
In methods with non-linear constraints, the subroutine
CONSTR (X, N, G, MC) is used. Here
G is a one-dimensional array of length MC that defines the constraints at the point X
MC is the number of constraints.

It is well known that local methods of optimization are sensitive to the scales of variables, as usual. The parameters of local methods in GMF are adjusted to the case when $A(I) = -1$ and $B(I) = 1,\ I = 1, N.$, as usual. Therefore, it is convenient to reduce the rectangular constraints $A \leq X \leq B$ to the N-dimensional hypercube $[-1, 1]^N$,. This can be done by the following reduction

formulae[2]).

$$X(I) = \frac{2}{B(I) - A(I)} X0(I) - \frac{B(I) + A(I)}{B(I) - A(I)},\ I = 1, N\ .$$

Here $X0$ are original variables, and X are variables scaled to fit into $[-1, 1]^N$.

Example. In most of the following examples, this test function $f(x)$[3] is used

$$f(x) = 2/N \sum_{i=1}^{N} (x_i^2 - \cos(18x_i)). \tag{4.3}$$

Here $N = 2,\ x_1 \in [-0.25;\ 0.5],\ x_2 \in [-0.125;\ 0.625]$. The subroutine is FURASN.

```
      FUNCTION FURASN(X,N)
      DIMENSION X(N)
      NN=N
      F=0.
      DO 10 I=1,NN
      XI=X(I)
   10 F=F+XI*XI-COS(18.*XI)
      AN=NN
      FURASN=F*(2./AN)
      RETURN
      END
```

Figure 4.1. Objective Function FURASN

3.3 Main Program

The main program defines the input data and the sequence of methods. At the beginning, one selects (using Table 4.1 or the ready-made example 'gmfl/ex8.f'), a desirable sequence of methods of optimization and analysis. Then a function FI(X, N) is written that evaluates the objective function at a fixed point X.

[2]in LBAYES those formulae are included in the algorithm, therefore no additional reduction is needed.
[3]This is the well known Rastrigin test function (Rastrigin, 1968).

If needed, the subroutine CONSTR(X, N, G, MC) is included. It evaluates the constraints at the point X. The length of arrays depends on the subroutines, as usual. The exception is the arrays IPAR and PAR. The length of these arrays is always thirty. The parameters of methods are included in the arrays IPAR and PAR by the given sequence of methods. Formal parameters IPA and IPAA are fixed using the rules that are given in the previous section. In the case, when only one method is used, IPA $=$ IPAA $= 0$.

3.4 Example of the Main Program

The global minimum of test function (4.3) is determined using the global Bayesian method BAYES1. Then the results of the global search are corrected by the local method of variable metrics MIVAR4.
The test function is represented as the function FURASN (X, N).
The arrays are: X, A, B, XN, HES, IPAR, PAR

It follows from Table 4.1 that, in the subroutine BAYES1, there are three integer parameters. In the subroutine MIVAR4, four integer parameters are used. In MIVAR4 four real parameters should be defined, too. This means that seven elements of IPAR and four elements of PAR should be fixed.
The main program is in Figure 4.2

```
      program main
      dimension x(2),a(2),b(2),xn(200),hes(3),
      ipar(30),par(30)
      data n,nm,nh,ipa,ipaa/2,200,3,0,0/
      data a/-0.25,-0.125/,b/0.5,0.625/
      data ipar/0,100,5,0,100,2,100,23*0/
      call bayes1(x,a,b,n,xn,nm,fm,ipar,ipa)
      ipa=3
      call mivar4(x,a,b,n,hes,nh,fm,ipar,par,ipa,ipaa)
      stop
      end
      function fi(x,n)
      dimension x(n)
      fi=furasn(x,n)
      return
      end
```

Figure 4.2. Example of MAIN

3.5 Installation

MS DOS version of GMF. Installing:

- copy the file 'gmf.arj' (see section 4).
- extract the GM files from the 'gmf.arj' archive

```
> arj e gmf.arj
```

This version was tested using the Digital Research Fortran-77 Version 4.1 and LINK-86 Linkage Editor Version 1.5f, Digital Research, Inc. The example:

- compile

```
> f77 ex9.for
> f77 i1mach1.for
> f77 exkor.for
```

- link

```
> link86 ex9,i1mach1,exkor
```

- run

```
> ex9
```

Here the file 'ex9.for' is an example of the Main program using EXKOR method to minimize FURASN function (see expression 4.3).
The file 'exkor.for' is the source of EXKOR.
The file 'i1mach1.for' contains portability codes and other service routines, such as independent random number generator, various test functions, etc.

Linux version of GMF. Installing:

- copy the file 'gmfl.tgz' (see section 4)
- extract the GM files from the 'gmfl.tgz' archive

```
> tar -zxf gmfl.tgz
```

This version was tested using the standard Linux f77-style shell script 'f77' to compile and load Fortran and assembly codes. The example:

- compile and load

  ```
  > f77 ex9.f i1mach1.f exkor.f
  ```

- run

  ```
  > a.out
  ```

Note, that here the Fortran file extensions are 'f'.

Chapter 5

TURBO C VERSION (TCGM)

1. General Discussion

A conceptual description of methods is given in (Mockus et al., 1997). Descriptions of real life examples of global optimization are in (Mockus et al., 1997) and in the next part of this book.

1.1 Purpose

The Turbo C version of the Global Minimizer TCGM is to solve global optimization problems similar to those of the Fortran version GMF. The optimization problem is defined in terms of variables describing the parameters that we can change, an objective function of those variables and some constraints restricting the possible change of variables. Formally, that is a nonlinear programming problem

$$\begin{aligned} \min_{a \le x \le b} f(x), \\ gi(x) \le 0,\ i = 1, ..., p \\ gi(x) = 0,\ i = p+1, ..., m. \end{aligned} \tag{5.1}$$

Here x is a vector of n variables. f(x) is an objective function. $a <= x <= b$ defines rectangular constraints. $gi(x) \le 0,\ i = 1, ..., p$ defines inequality constraints. $gi(x) = 0,\ i = p+1, ..., m$ defines equality constraints. The objective function is deterministic or stochastic (for some methods). Rectangular constraints are defined for all the methods. The linear and non-linear constraints

are defined just for two local methods: the Simplex by Nelder and Mead $Flexi$ (Himmelblau, 1972), and the non-linear programming by Schittkowski Nlp (Schittkowski, 1985).

2. User's Reference

2.1 Requirements

- Turbo C version 2.0 or above.
- An experience of writing simple C functions.

2.2 Installation

- Copy the file $tcgm.arj$ from a web-site (see Section 4). Extract the GM files from the $tcgm.arj$ archive

```
> arj e tcgm.arj
```

2.3 Initialization

- Run the batch file GM.BAT using the command GM, and enter the Turbo C editor.
- Define the objective function as some C function with the name f. This function has two parameters:
  - double array of values of each variable,
  - the number of variables.
- Define the general linear or non-linear constraints as a C function named constr. It has four parameters:
  - double array of values of variables,
  - number of variables,
  - double array of values of constraints,
  - number of constraints.
- Compile file FI.C using directive Alt+C and correct possible errors.
- Link file FI.C to GLOBAL MINIMUM system using directive Alt+L.

- Exit Turbo C using directive Alt+X.

Now the GLOBAL MINIMUM is ready to minimize the problem that is defined in the file FI.C. Descriptions of objective functions and constraints have to be in the same file. Running GM.BAT, the file FI.C is displayed on the screen by Turbo C editor. Rectangular constraints are defined using the screen. Two algorithms, Nlp and Flexi, are exceptions. Here the constraints are defined by C function constr in the file FI.C. The four basic steps of initialization are

- run GM.BAT,
- edit file FI.C,
- compile FI.C,
- link file FI.C to system GLOBAL MINIMUM.

2.4 Minimization

The system will ask the following questions.

- Question: Number of variables.
  Action: Enter the number of variables of the problem.

- Question: Number of constraints.
  Action: Enter the number of linear and non-linear constraints of the problem (not counting the number of rectangular constraints that are defined interactively).

Now one see the GLOBAL MINIMUM copyright notice. GLOBAL MINIMUM is implemented as menu, dialog box and windows system. Each object can be either active or passive. There can be only one currently active object. A user can work only with active objects. There is a rectangular caption at the top of each object (menu, dialog box, or window). If this caption is bright, the object is active.

2.5 Menu System

- Main Menu

- press any key and enter the MAIN MENU,
- select the type of optimization or exit the menu system selecting EXIT.

- Global Optimization Menu
  - select the GLOBAL item from the main menu,
  - select any of six available global optimization methods: LPMIN, MIG1, BAYES1, EXTR, UNT, GLOPT.

- Local Optimization Menu
  - select LOCAL item from the main menu,
  - select any of three available local optimization methods: NLP, FLEXI, or LBAYES. The initial point for local optimization is the result of previous global optimization, by default.

- Parameter Box
  - display of the PARAMETER DIALOG BOX follows the selection of method,
  - enter corresponding parameters of methods and press RETURN. Recommendations how to choose parameters are given in the description of methods (Mockus et al., 1997).

- Confirmation Menu
  - CONFIRMATION MENU appears following the selection of parameters,
  - RUN means run the method,
  - EXIT means return to LOCAL or GLOBAL OPTIMIZATION MENU.

- Operation Menu
  - OPERATION MENU appears following the selection of RUN item,
  - RUN means run the method, in case if it was stopped by selecting STOP item,
  - STOP means temporary stopping of the method and activating the OUTPUT DIALOG BOX,
  - press ENTER to return to the OPERATION MENU and to make the OUTPUT DIALOG BOX inactive,
  - EXIT means exit the method and return to GLOBAL or LOCAL in the OPTIMIZATION MENU,

- ENTER means return to GLOBAL or LOCAL in the OPTIMIZATION MENU, after the method finishes optimization,
- RECORD means turn RECORD WINDOW on/off,
- PROJECTION means turn PROJECTION WINDOW on/off.

■ Information Display

- OUTPUT DIALOG BOX shows the results of each iteration (usually best point and goal function value at this point) when the method runs (see Figure 5.1),

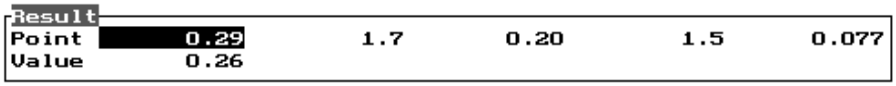


Figure 5.1. Table of TCGM results.

- RECORD WINDOW shows how the best function value depends on the iteration number (see Figure 5.2),

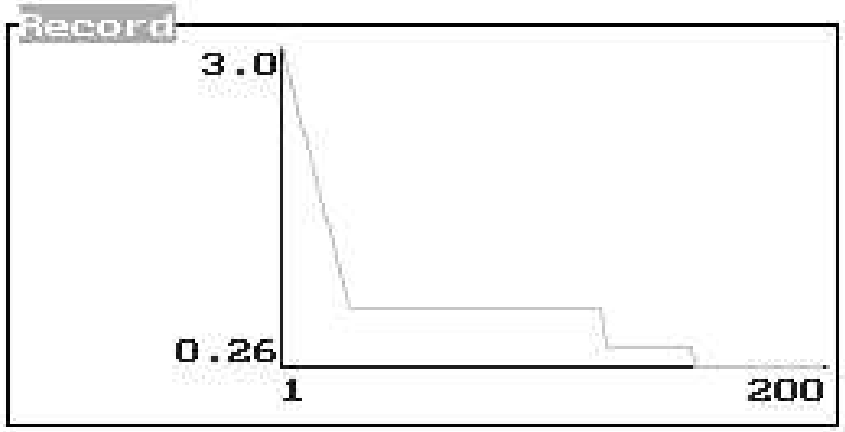


Figure 5.2. Graph of TCGM results.

- RECORD WINDOW is on when the method is started,
- if the graph on RECORD WINDOW is horizontal for a long time it can suggest that the minimum is reached or that the method exhausted itself for this problem and it is better to try another method,
- if one selects RECORD item in OPERATION MENU while running the method then the RECORD WINDOW becomes invisible, if it was visible, and vice versa,

  - PROJECTION WINDOW shows the projection of the observed (calculated) objective function values, the numbers of projection planes correspond to the numbers of variables,
  - if one selects a PROJECTION item in OPERATION MENU while running the method, then one will see projection menu consisting of two items: ON and OFF, meaning on/off the PROJECTION WINDOW,
  - if one selects ON then the projection dialog box appears and we can enter numbers defining the projection planes,
  - select the number, press ENTER and one see how the observed values of the goal function depend on the corresponding values of the selected variable, the values of all other variables will not be seen.
- RESULT DIALOG BOX shows the results of calculations after the method finishes optimization.

2.6 Navigation

- On Screen
  - screen is divided into three areas: bottom, middle and top, the functionality of these areas is different,
  - bottom area is for general information, such as description of variables, displaying error messages, displaying the state of an active object,
  - middle area is only for displaying and editing variables,
  - the top is a working area; menus, dialog boxes and windows are displayed here.
- On Menu
  - if some menu item is invisible, one opens it using corresponding cursor keys: UP, DOWN, LEFT, RIGHT,
  - the contents of the menu will scroll until the last available item will be reached.
- Dialog Box
  - on the DIALOG BOX one can see such variables as the parameters of method, coordinates of the best and current point, best and current values of goal function and so on, to view a variable one selects the variables name,
  - if the variable is a scalar, one see the corresponding value, next to the variable's name, the description of this variable is displayed at the

bottom area of a screen, the exact value of the variable is displayed at the middle area,

- if the variable is a vector then, after selecting the variables name, one can select the corresponding component of the vector using keys LEFT and RIGHT, the value of the selected component will be in the middle area, the number of component will be in the bottom area, along with the description of variable,
- to see invisible components of the vector, use the keys LEFT and RIGHT, the vector scrolls, if there are invisible components,
- press SPACE, to edit a variable, press ENTER, to exit the editor.

2.7 Moving

- Press the F3 functional key and the red border will appear, move the border with keys UP, DOWN, RIGHT, LEFT and press ENTER to stop.
- The previous image of the object will be deleted and the object will be placed inside the new border.

2.8 Sizing

- Press the F4 functional key, move the object and press ENTER. The red border appears.
- Size the border using the keys UP, DOWN, RIGHT, LEFT and press ENTER to stop.
- The previous image of the object is deleted and the object is placed into new borders.

No changes happens, if borders are greater than maximal size of the object, or less than its minimal size.

Chapter 6

C++ VERSION (GMC)

1. Purpose of GMC

The Global Minimizer's C++ version GMC is for continuous global optimization (Mockus et al., 1997). Users define their optimization problems describing

- parameters that can be changed (defined as variables x),
- function of those variables (defined as objective function $f(x)$),
- constraints restricting a possible change of variables (defined as inequalities $gi(x) \leq 0, i = 1, ..., p$, or equalities $gi(x) = 0, i = p + 1, ..., m$).

Formally

$$minf(x), \tag{6.1}$$
$$a \leq x \leq b \tag{6.2}$$
$$gi(x) \leq 0, i = 1, ..., p, \tag{6.3}$$
$$gi(x) = 0, i = p + 1, ..., m. \tag{6.4}$$

The objective function $f(x)$ may be deterministic or (for some methods) with "noise." As usual, the optimization region is defined by rectangular constraints (4.2). For some methods, Nlp and Flexi, this region is defined by linear and non-linear constraints (6.3) and (6.4).

2. User's Reference

2.1 Requirements

- C++ compiler, for example, GNU g++ compiler or any ANSI C++ compliant compiler,
- X-Window system,
- UNIX (Linux, Solaris, AIX),
- experience in writing simple C, or C++ functions.

2.2 Installation

Differences between various UNIX versions are reflected in the file 'Makefile'. An example for individual Linux users is in Figure 6.1. An AIX version for individual users is in (Mockus et al., 1997). A classroom (collective user) version for Solaris is there, too. Using some other UNIX version, the Makefile has to be changed accordingly. Installing GMC one should

- copy the archive file $gmc.tgz$ from the web-site (see section 4) by the unix command

```
> mcopy a:  gmc.tgz
```

- extract GM files from the archive 'gmc.tgz' by the command

```
> tar -zxf gmc.tgz
```

2.3 Initialization

- Define the objective as some function fi. The function fi has two parameters: an array of values of each variable and the number of variables. The function fi returns the value of objective, given the values of variables.
- Define "non-interactive" linear or non-linear constraints as a function named $constr$ It has four parameters: an array of values of variables, the number of variables, an array of values of constraints, and the number of constraints. Values of constraints are returned by the function $constr$.

A user can also define rectangular constraints interactively, during the optimization

- Describe objectives and constraints in a file named $fi.C$ (see Figure 6.2)
  - the objective function is implemented by the routine

    ```
    double fi (const double *xx, int dim)
    ```

  - the constraints are defined by the routine

    ```
    void  constr (const double *x,  int n, double *g,
    int ng)
    ```

- Start optimize the problem described in the file $fi.C$.
- set X-window
- compile

  ```
  >  rm  fi.o >  make
  ```

- run

  ```
  >  ./test
  ```

The opening menu of GMC is in Figure 6.3

- The Main Menu:
  Global, Local, Operations, Quit, Parameters, Results, Output.
- Global Optimization Menu:
  Bayes1, Mig1, Unt, Exkor, Glopt, Lpmin.
- Local Optimization Menu:
  Nlp, Flexi, Lbayes.
  By default the initial point for local optimization is the result of previous global optimization.
- Operations Menu:
  Run, Stop, Exit.
- Quit Command.
- Parameter Box.
  Enter the parameters of methods and point O.K.
  An example of the parameter box is in Figure 6.3.

- Results Box:
  Shows the best objective Y and the best variables X(I). If no changes are needed, point O.K.
- Output Menu:
  - Convergence, Projection, and Numeric Windows.
  - Convergence window shows how the objective depends on the iteration number (see the Figure 6.4.
  - Projection windows show how the objective depends on the different variables. One defines the projection by entering the number of variable. Figures 6.5 and 6.6 show two projections.
  - Numeric window shows the current values of objective and variables (see Figure 6.7).
- Using Menu:
  - To see "invisible" components of a vector, point to 'down' or 'up' arrows. The vector scrolls, if there are invisible components.
  - To edit a variable, touch BACKSPACE[1], then edit the variable and point to O.K.
  - Enter the Parameter Box immediately after selecting a method.

[1] One should use the 'xmodmap' command to adapt the keymap, if the BACKSPACE key is not working.

```
#this is for linux 2.0.0
# C++ compiler. If error is "Can not load ..."
# (try CC or gcc instead of g++)
C++ = g++ -I$(INCDIR) -L$(LIBDIR) -O2
# pathname to X11/*.h files
# if compilation errors are "Can not find include file ..."
 then
# try /usr/local/include or /usr/include/X11R5
INCDIR     = /usr/include
# pathname to libX11.a
# If error is "Can not find library ..."
# try /usr/local/lib or /usr/lib/X11R5
LIBDIR    = /usr/X11/lib
CPP        = /lib/cpp
F77        = f77
LIBS       = -lX11 -lm
OBJS       = test.o WINDOW.o MENU_ITEM.o  SMENU_ITEM.o
 work.o MENU.o DBOX.o CONTROL.o STATIC.o EDIT.o BUTTON.o
 LBOX_ITEM.o LBOX.o     SCROLL_BAR.o SBBUTTON.o
 OUTPUT.o mig1.o MIG1.o exkor.o  task.o
 bayes1.o SUBMENU.o BAYES1.o SHIT.o
 CLIENT.o lbayes.o PAR_DBOX.o ERROR.o
 PROJ.o VAR_DBOX.o unt.o UNT.o EXKOR.o RES_DBOX.o
 VEDIT.o CONV.o fi.o NUM.o shit.o IO_DBOX.o LBAYES.o
  lpmin.o LPMIN.o glopt.o GLOPT.o flexi.o FLEXI.o HELP.o
RES        = menu.res bayes1p.res nlpp.res var.res
lbayesp.res
     res.res num.res mig1p.res untp.res exkorp.res
 lpminp.res
             gloptp.res flexip.res help.res
TARGET     = test
.SUFFIXES: .C .o .rc .res .f
.f.o:   $(F77) -c $*.f
.C.o:
$(C++) -c $*.C
.rc.res:
$(CPP) -P $*.rc > $*.res
all: $(TARGET) $(RES)
$(TARGET): $(OBJS) Makefile
$(PURIFY) $(C++) -o $(TARGET) $(OBJS) $(LIBS)
# DO NOT DELETE THIS LINE -- make depend depends on it.
```

Figure 6.1. Example of 'Makefile' for Linux.

```
/* File: fi.C */
#include "fi.h"
#include <math.h>
int number_of_variables =2;
/* Objective function*/

double fi (const double *x, int n)
{
   int    i;
   double y = 0;

   for (i = 0; i < n; i++) {
double z = x [i];
y += z * z - cos (18. * z);
}
return 2 * y / n;
}
/*Constraints */

void constr (const double *x, int n, double *g, int ng)
{
/*
ng > 0 - calculates all constraints
ng < 0  - calculates only constraint with number (-ng)
(in this illustrative example there are no s,
except interactively defined ones)
*/
}
```

Figure 6.2. Example of the file 'fi.C.'

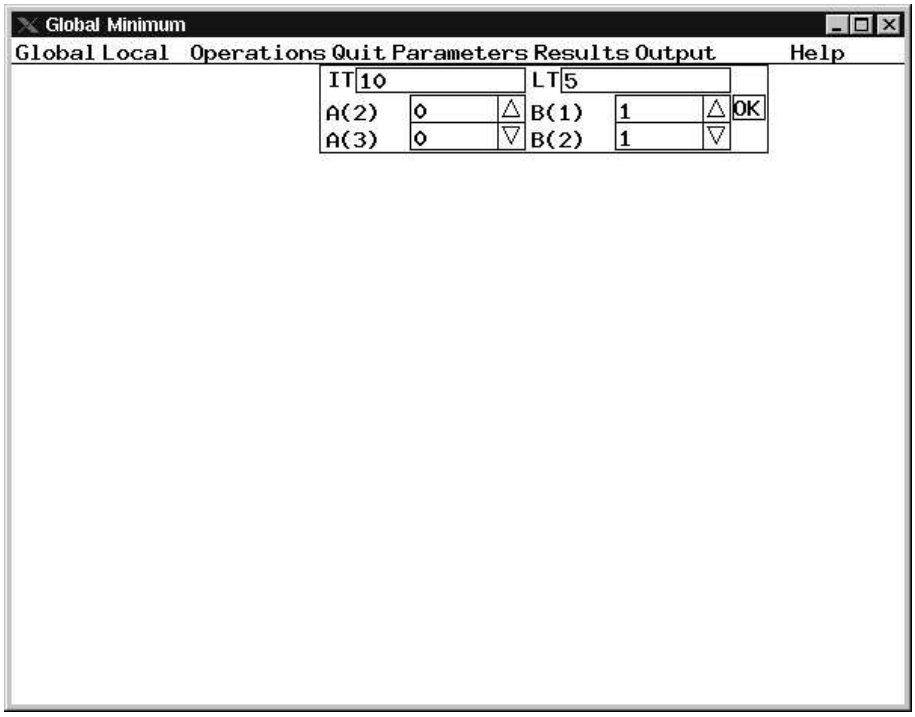


Figure 6.3. An example of GMC opening menu and parameter box.

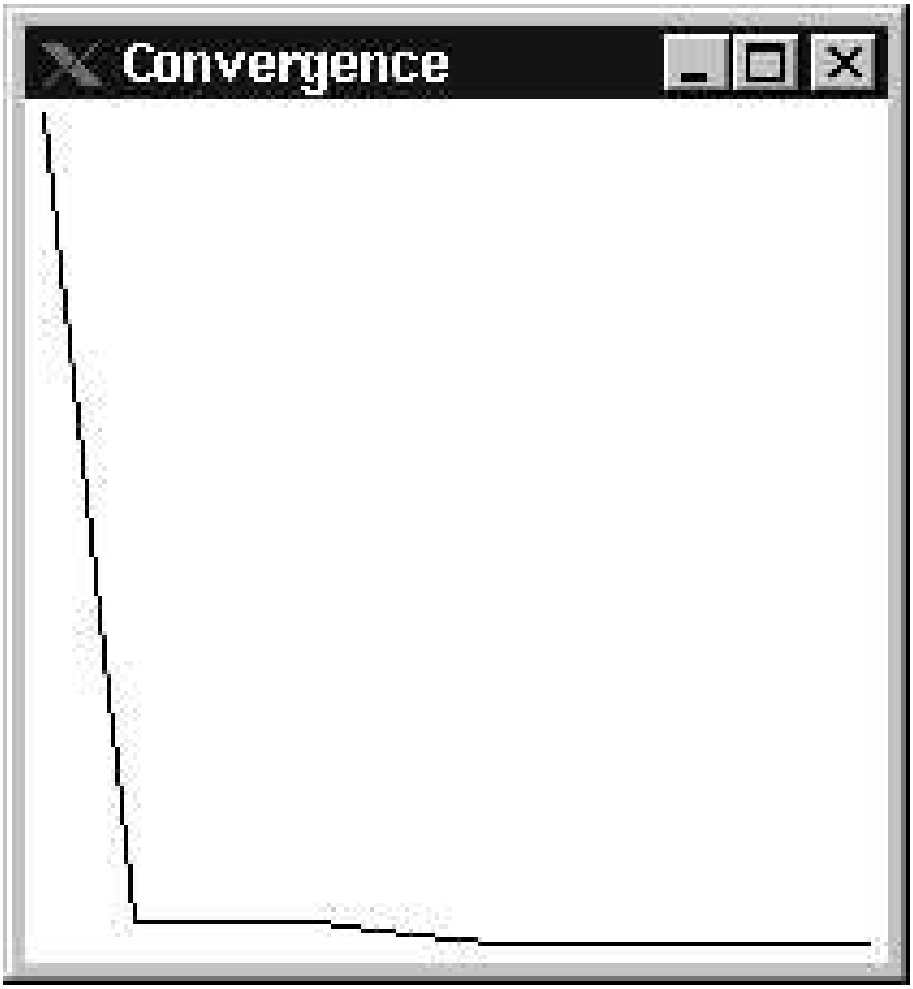


Figure 6.4. Convergence window.

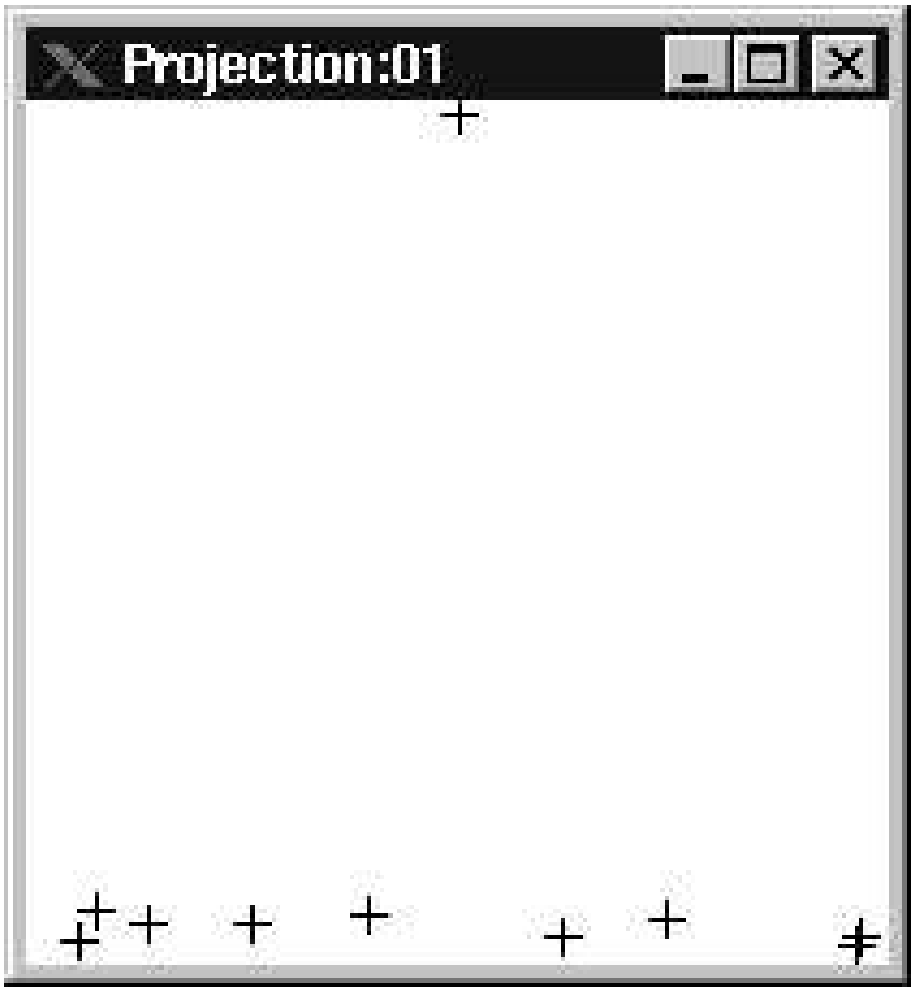


Figure 6.5. Projection 1 (shows how $f(x)$ depends on the first variable).

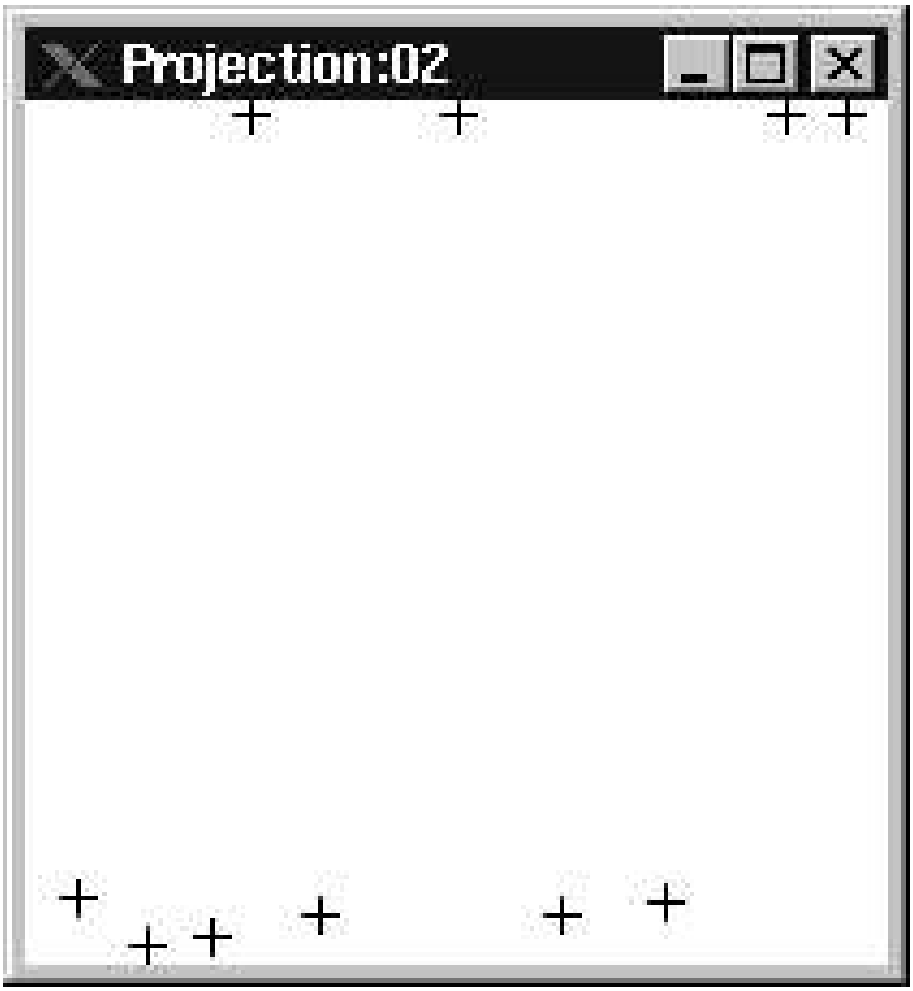


Figure 6.6. Projection 2 (shows how $f(x)$ depends on the second variable).

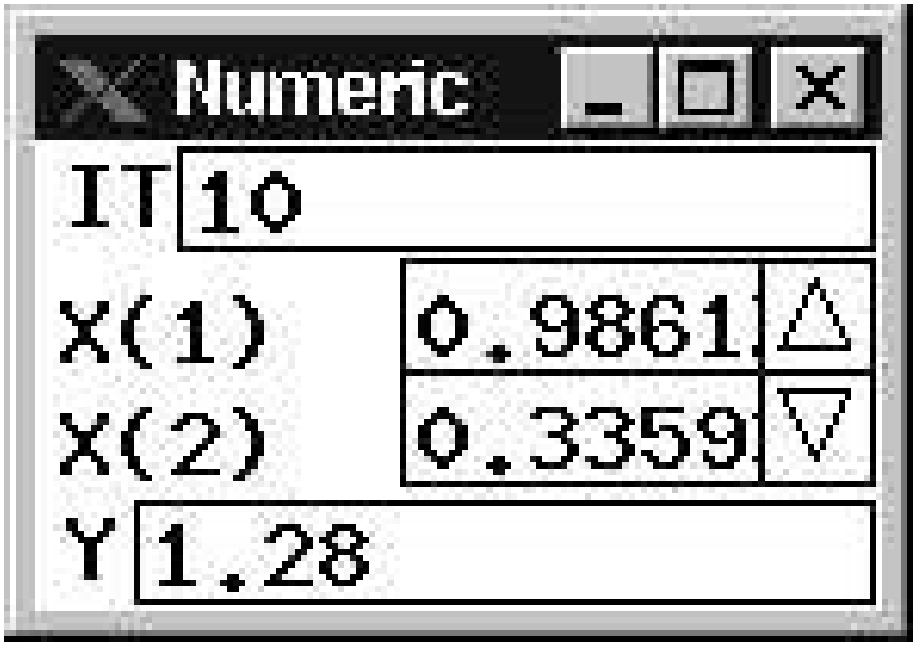


Figure 6.7. Current numeric results.

Chapter 7

C# VERSION (GMC#)

1. Users Guide

The GMC# system is class framework for implementing, testing and viewing custom made or already installed minimization tasks and methods. The application is flexible and can be extended in many ways. This application helps to understand and visualize optimization problems, both mathematical and software related (Mockus, 2000, Andriulaitis and Varnagiris, 2000).

1.1 Running GMC#

The GMC# system can be run only as C# Application and only under Win98/Win2000/NT/XP OS. C# is one of the newest Microsoft products and it is only in beta stage, but is stable enough. Run GMCS.exe file and you will see this screen 7.1: Note: You need to have installed Microsoft .NET framework. For more information go to http://msdn.microsoft.com/net/

GMC# have methods, tasks and analyzers. The task is the work to be done and method is way of doing that. The analyzers represents results in a visual way.

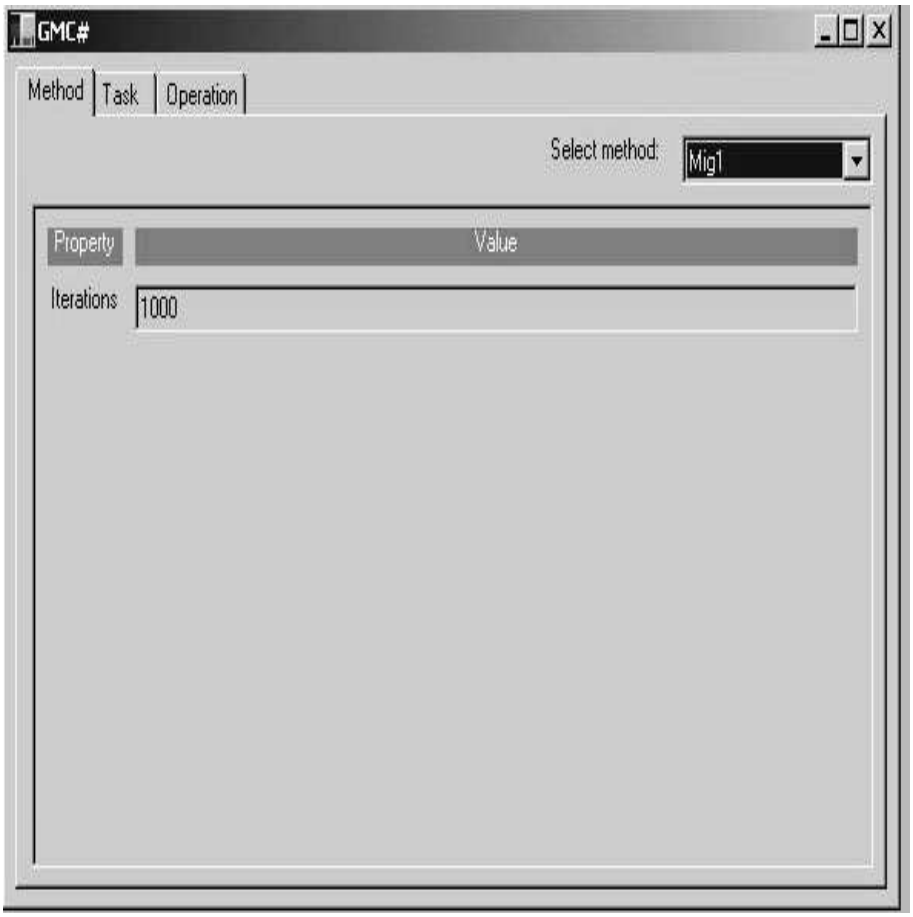


Figure 7.1. user1.

1.2 Method selection

The choice control at the top of the window shows the list of supported methods. One of them is to be selected. After selecting a method the window changes. The configuration window depends on the actual method selected, but usually it will have an "Iterations" field, where the number of iterations, can be changed. If the entered value is invalid, or the number is out of bounds, the appropriate error message is displayed.

1.3 Task selection

The choice control at the top of the window shows the list of available tasks. One of them should be selected and configured before optimization starts. After selecting a task the window changes. The configuration window depends on the actual task selected. It lists the properties of the task and then the range of the minimization domain 7.2.

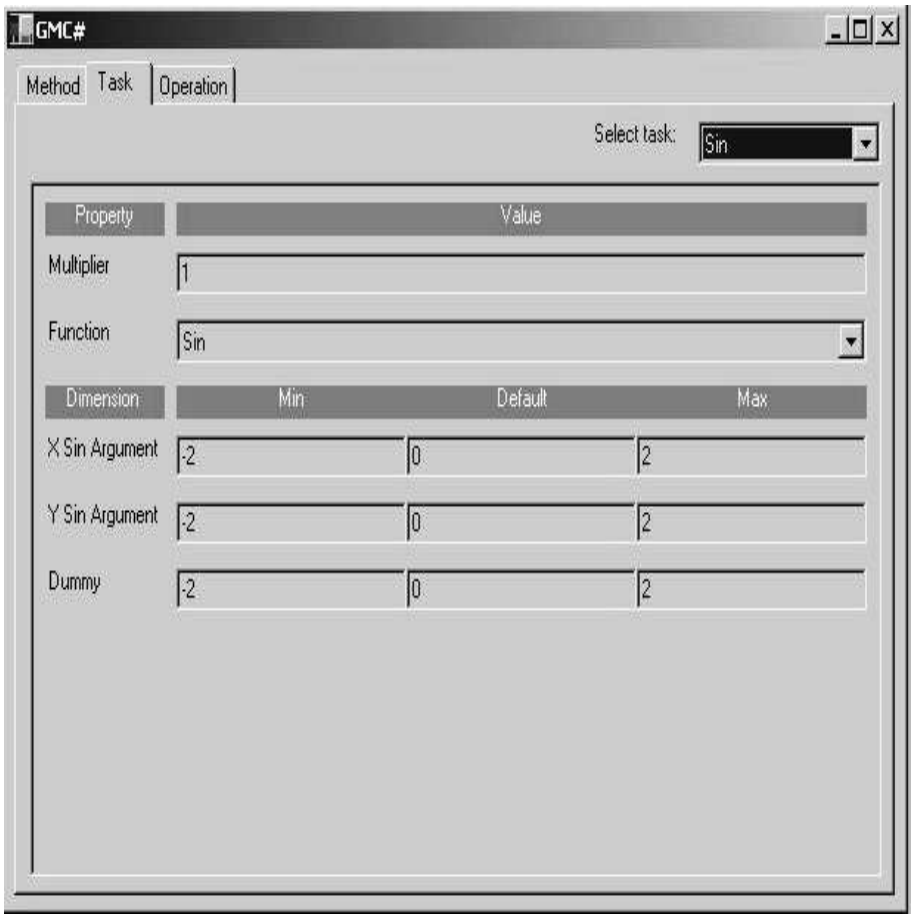


Figure 7.2. user2.

Tasks have some properties including dimensions that are configured.

1.4 Operations

Operations window 7.3:

Figure 7.3. user3

You can choose operations:

* Run. Starts selected task calculation by using selected method. GMC# shows current function value (0.9884 in this screen) and current iteration number (224 in this screen). * Pause. When RUN button is pressed, calculation starts. By pressing PAUSE, the user can pause calculation for unlimited time. * Stop. STOP button stops calculations and destroys gathered results. * Analysis. When calculation is complete, the user can view results. For more information see section: Analysis windows.

Analysis windows

After optimization one can view results.

The projection windows show the relation between one of the dimensions of the domain of the minimization function and the value of the function 7.4.

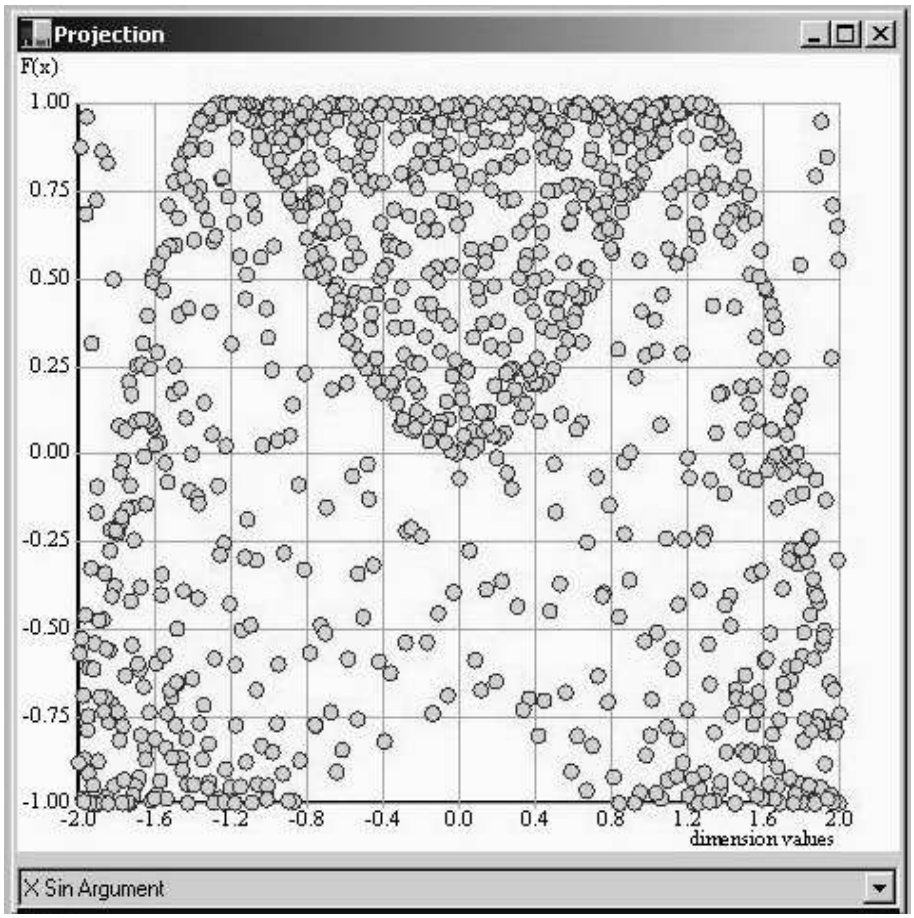


Figure 7.4. user4.

The convergence window shows the progress of minimization on iteration basis 7.5.

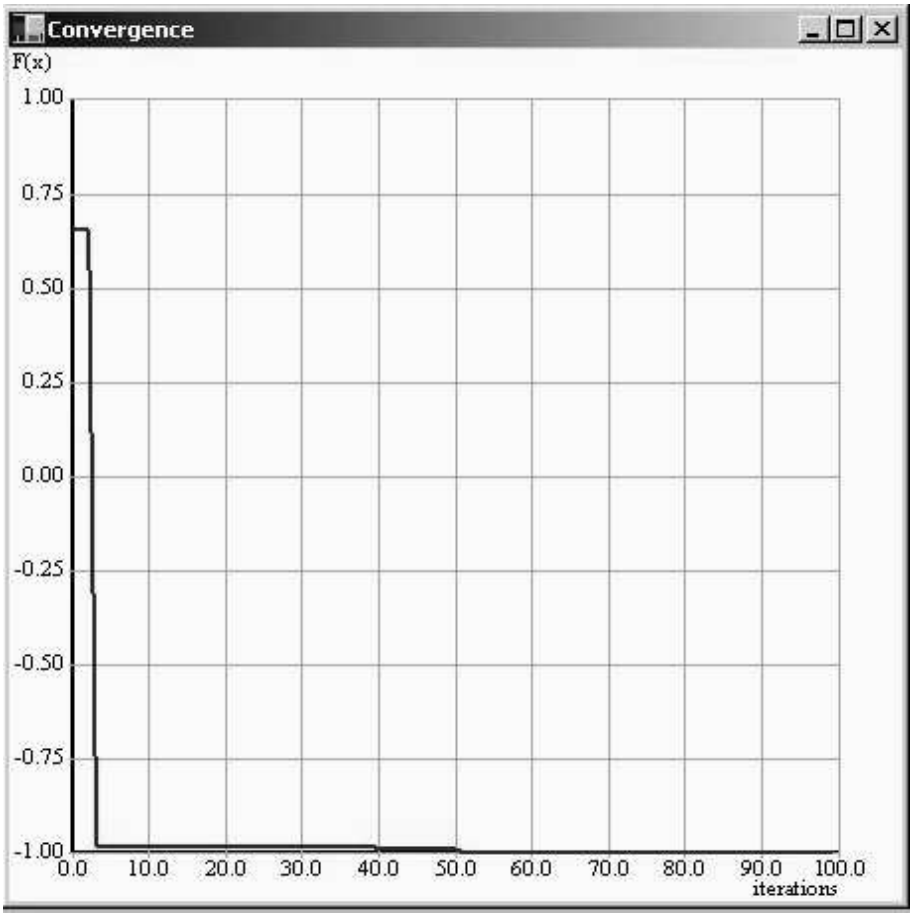


Figure 7.5. user5.

2. Design Documentation

The GMC# system is class framework for implementing and testing custom minimization tasks and methods. The application is highly flexible and can be extended in many ways. This document is dedicated for design explanation. Figure 7.6 shows the top level design .

2.1 Top Level Design

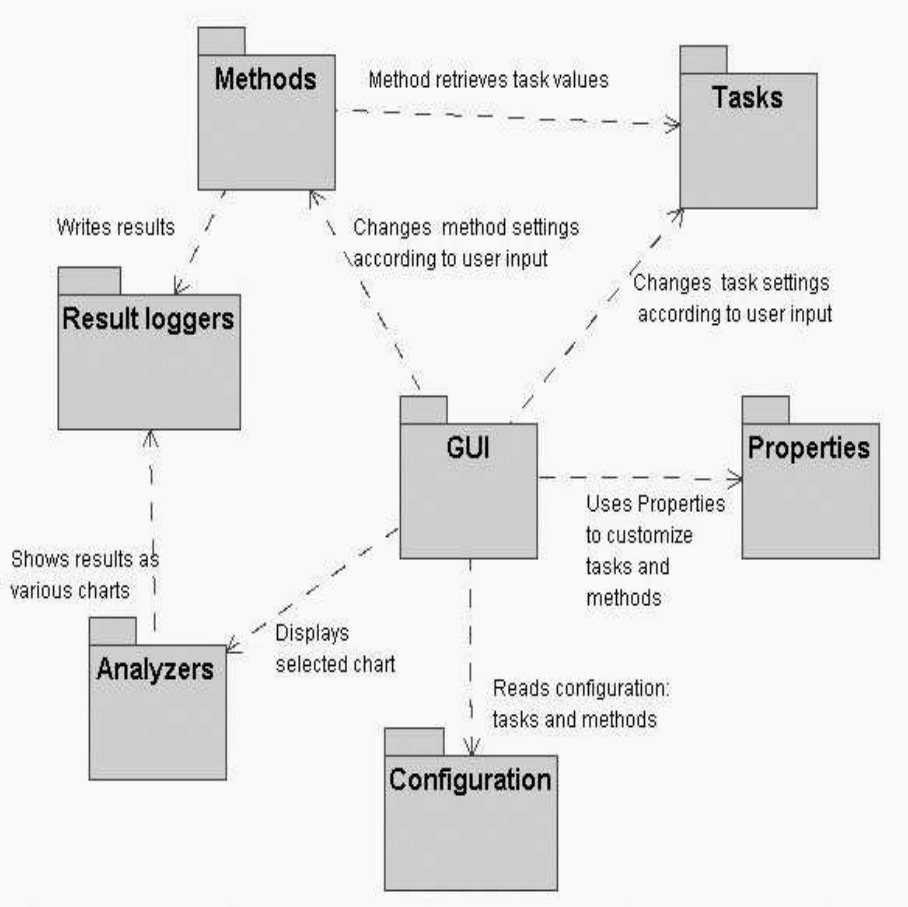


Figure 7.6. Top level design.

Top level diagram describes general relationships between C# packages.

2.2 GUI Package

Figure 7.7 shows the GUI package. This package is mediator between

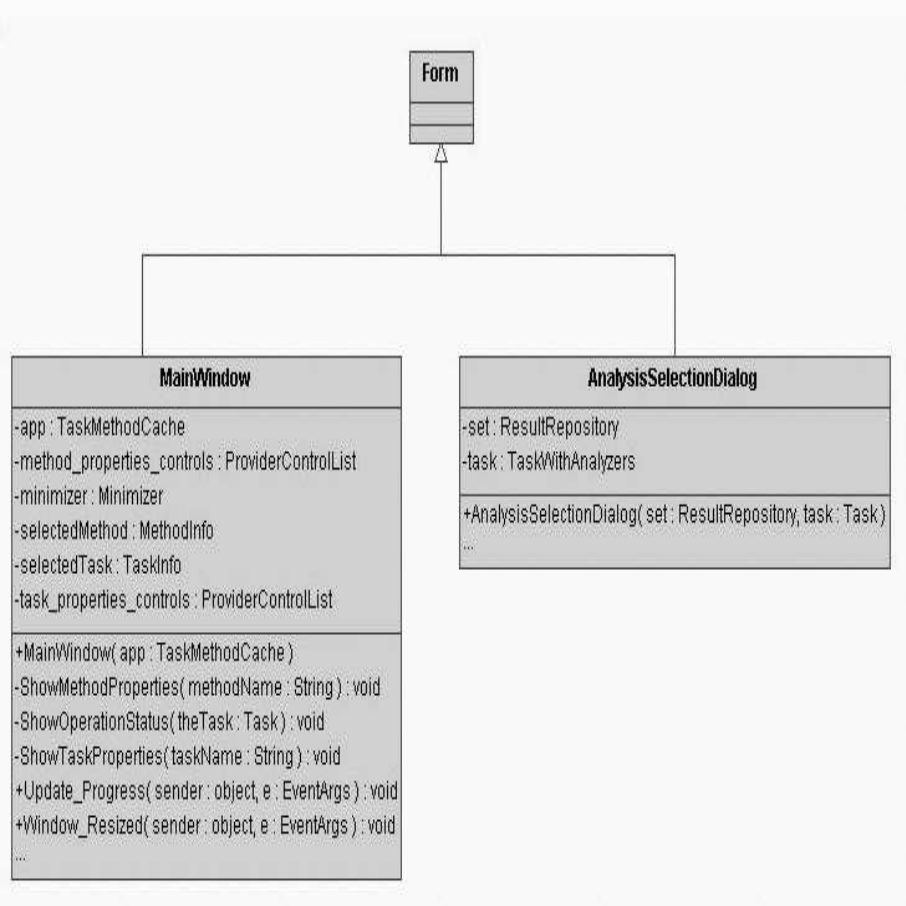


Figure 7.7. GUI package.

tasks, methods, result loggers, analyzers, properties and configuration packages. Also, it represents GUI for user. Here the class $*MainWindow$ is the main class. It concatenates:

1. Configuration package to load tasks and methods.
2. Displays methods and lets customize their properties.

3. Displays tasks and lets customize their properties.
4. Controls optimization workflow.
5. Displays analyzers.

$*AnalysisSelectionDialog$ class selects analyzer to display. It shows all available analyzers for selected task.

2.3 Methods Package

Figures 7.8 and show 7.8 the methods package This package describes

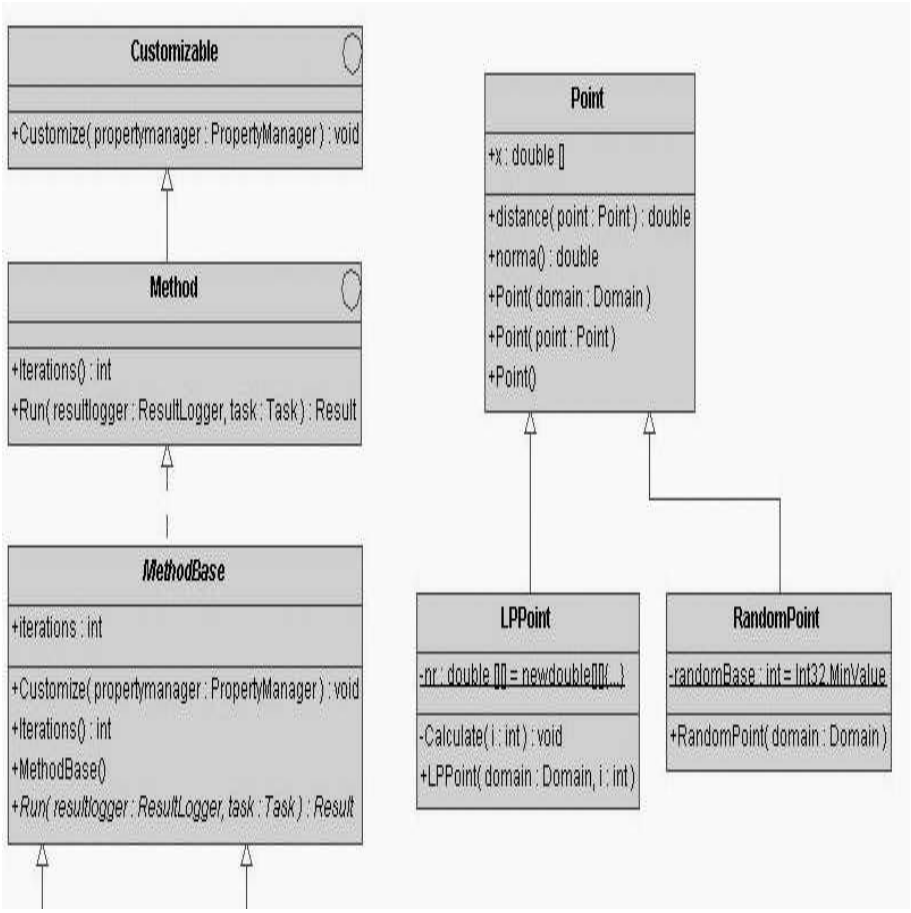


Figure 7.8. Methods package 1.

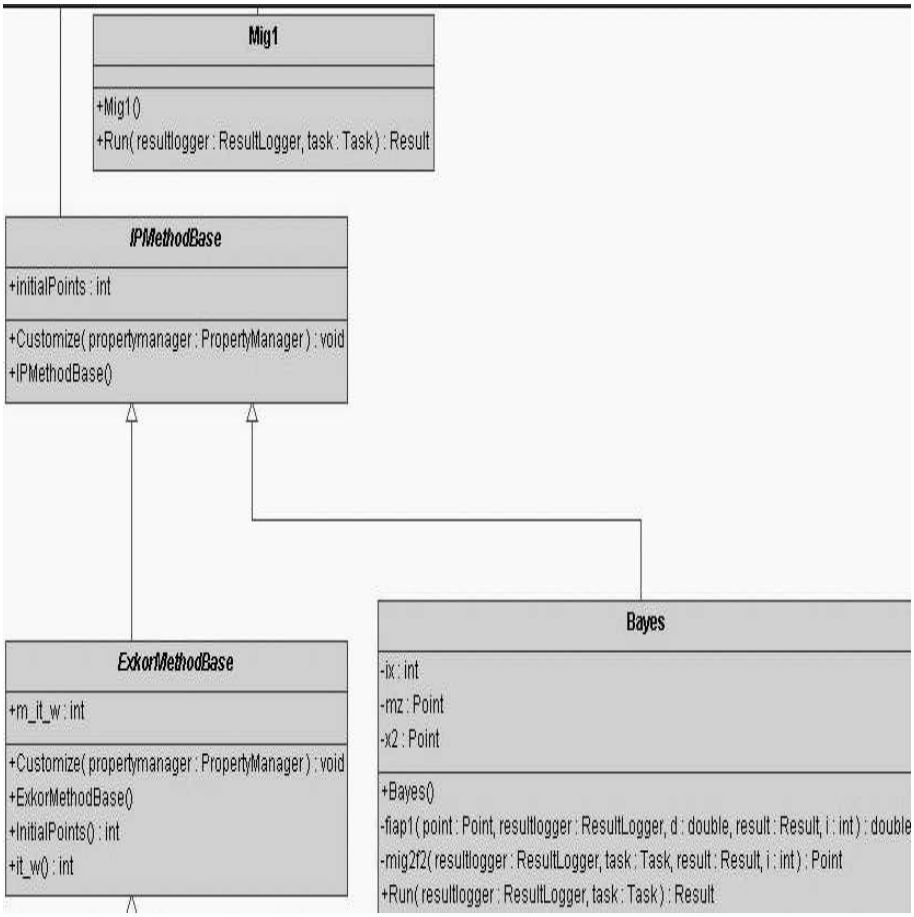


Figure 7.9. Methods package 2.

method interfaces and implemented methods.

The $*Customizable$ interface represents customizable property such as any value, choice value and others.

The $*Method$ interface defines for methods tasks, which must be optimized and result loggers, where to place results.

The $*MethodBase$ class adds customization possibility through GUI.

The $*Mig1$ Class implements the Monte Carlo method $Mig1$. Summarily the classes $*IPMethodBaseBayes$ and $ExkorMethodBaseExkor$ implement the $Bayes$ and $Exkor$ methods.

$*Point$ represents n-dimensional point of domain.

$*RandomPoint$ represens n-dimensional random point of domain.

$*LPPoint$ represents point used in Bayes method.

2.4 Tasks Package

Figure 7.10 shows the tasks package This package contains task classes (functions that must be optimized by various methods).

The $*Customizable$ interface represents customizable property such as a value, the choice value e.t.c. The $*Task$ interface defines the function to be optimized and the domain of variable parameters.

The $*TaskWithAnalyzers$ interface defines analyzers to display results.

The class $*Domain$ represents set of possible arguments.

The class $*Sin$ calculates the function Sin.

The class $*SinDomain$ defines a set of Sin arguments.

The class $*Square$ calculates the task $Square$.

The class $*SquareDomain$ defines a set of $Square$ arguments.

2.5 Properties Package

Figures 7.11,7.13, and 7.13 show the properties package

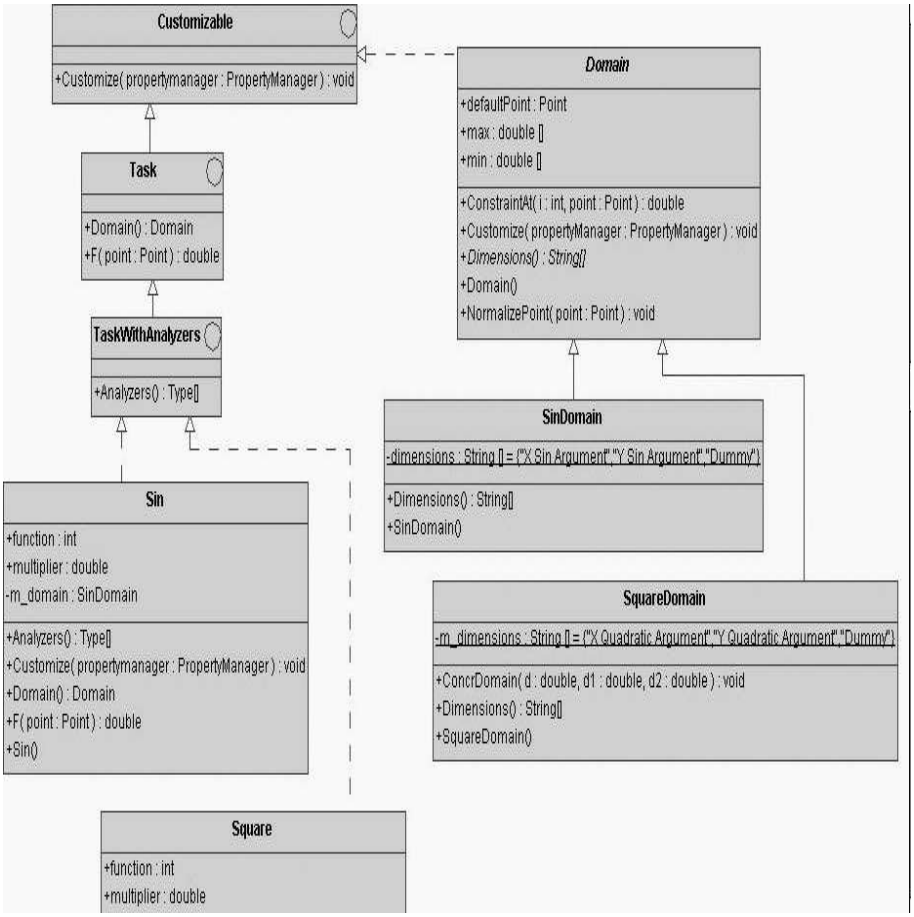


Figure 7.10. Tasks package.

This package is dedicated for user input gathering and converting to method and task data.

The $*Property$ interface t represents all customizable properties of methods and tasks.

The $*FieldProperty$ stores data from simple input field that manages FieldProvider .

The $*ChoiseProperty$ stores data from choice box.

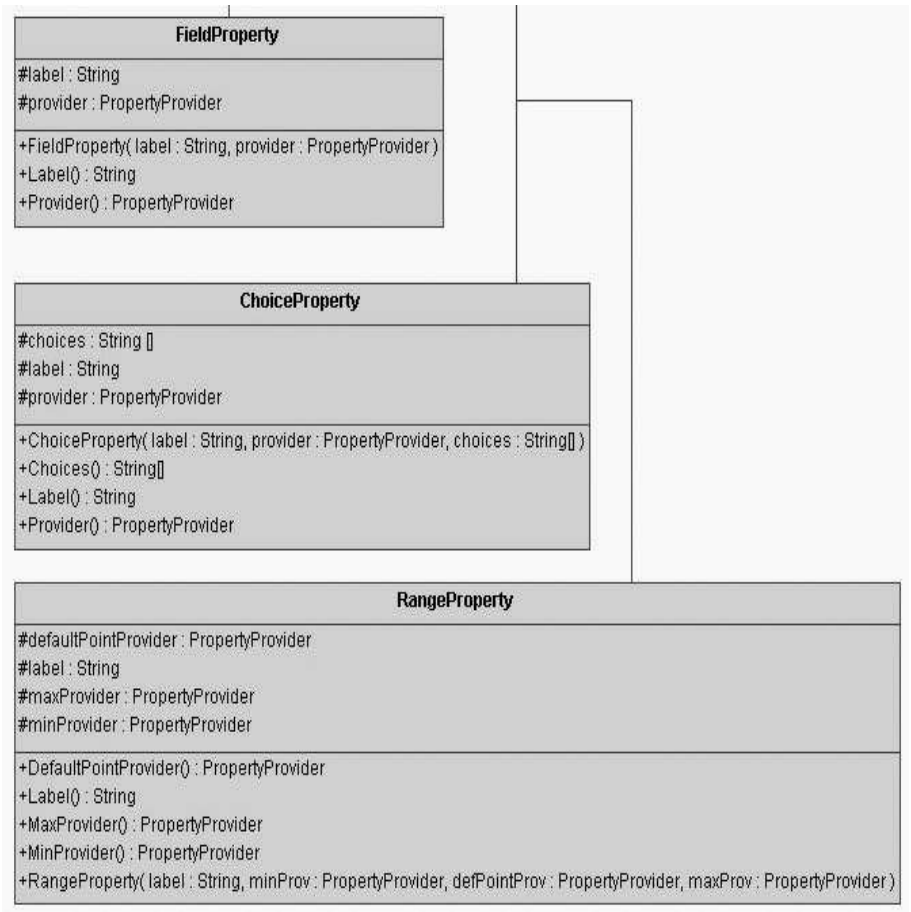


Figure 7.11. Properties package 1

The $*RangeProperty$ stores data from two simple input field(they manage two $FieldProvider$ objects) that is used for range specification.

The $*PropertyProvider$ interface provides methods for setting and getting values from GUI fields.

The $*FieldProvider$ class implements the $PropertyProvider$ for setting and getting data of single GUI fields.

The class $*ArrayElementProvider$ implements the $PropertyProvider$ for setting and getting data of multiple GUI fields.

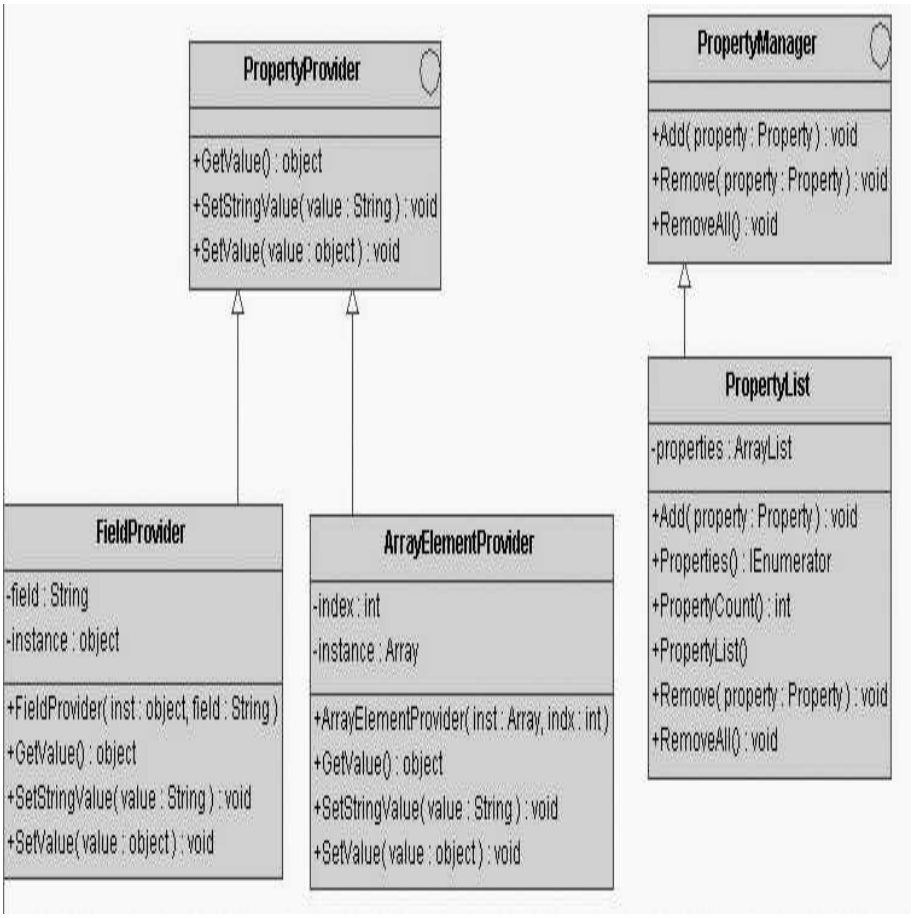


Figure 7.12. Properties package 2

The $*PropertyManager$ interface selects methods and properties.

The class $*PropertyList$ inherits the $PropertyManager$ and implements methods. The class $*TaskInfo$ stores task properties entered by user.

The class $*ProviderControlList$ stores and manages providers, applies changes to GUI.

The class $*MethodInfo$ stores method properties entered by user.

The class $*TaskMethodCache$ stores loaded and created tasks and methods, provides information about them.

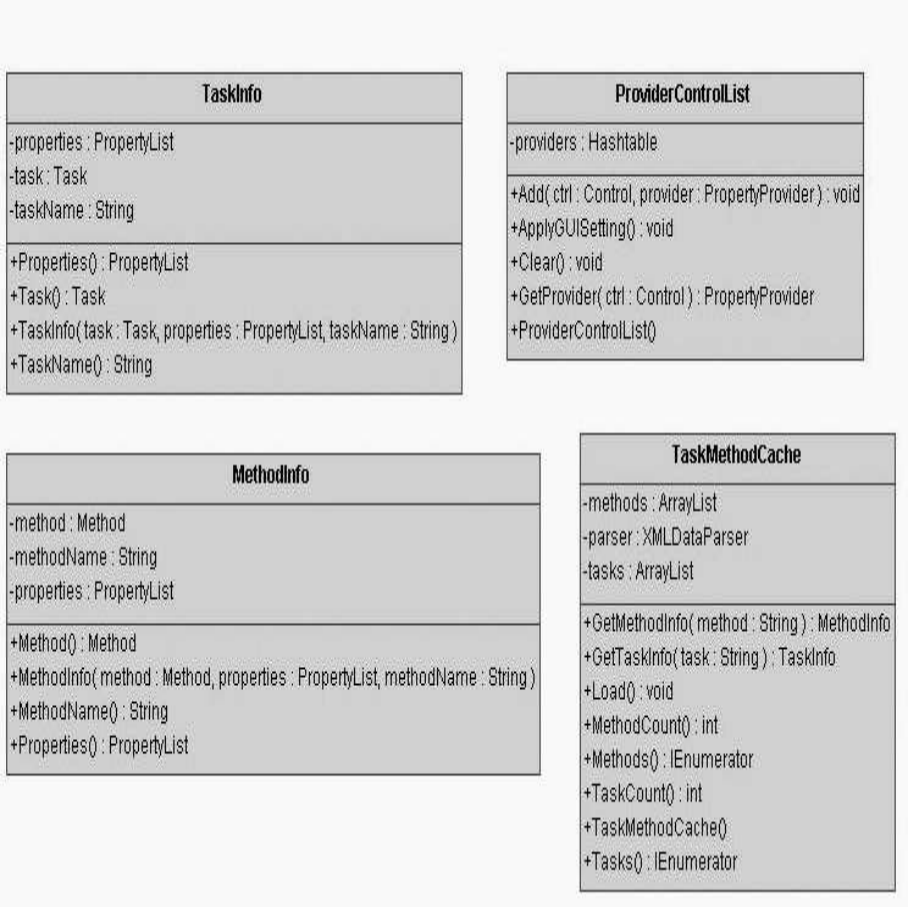


Figure 7.13. Properties package 3

2.6 Configuration Package

Figure 7.14 shows the configuration package This package is dedicated for configuration file parsing.

$*ConfigurationEntry$ represents configuration file entry: tasks or method and library, where stored these classes implementation.

The exception class $*ConfigurationException$ is thrown by configuration parsing error.

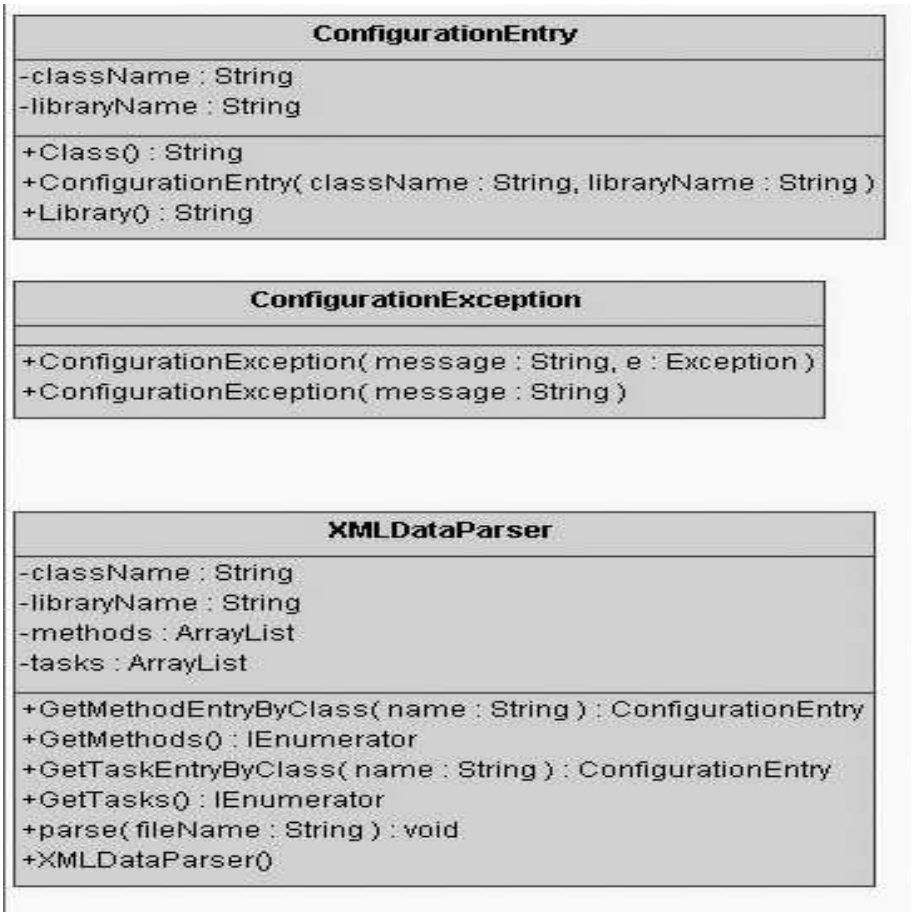


Figure 7.14. Configuration package

The class $*XMLDataParser$ is the main configuration parsing class. This object takes file name (where is stored configuration in XML format) and parses it (finds tasks, methods and libraries where methods and tasks are implemented).

2.7 Analysers Package

Figures 7.15 and 7.16 shows the analyzers package

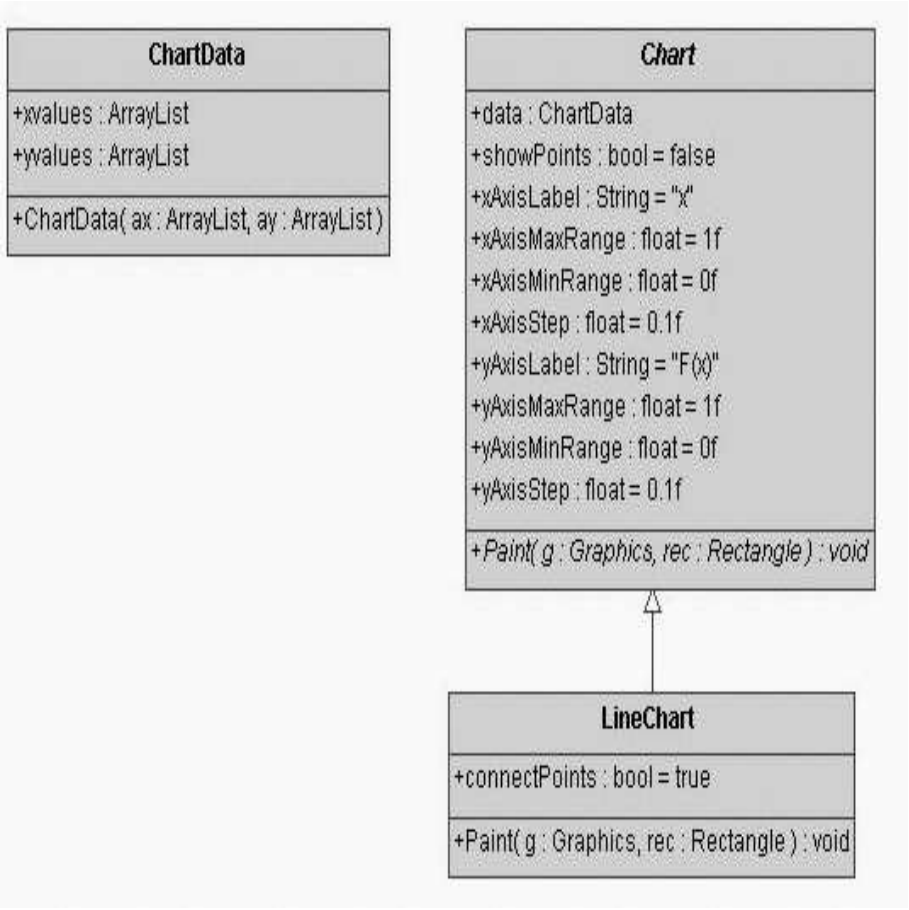


Figure 7.15. Analysers package 1.

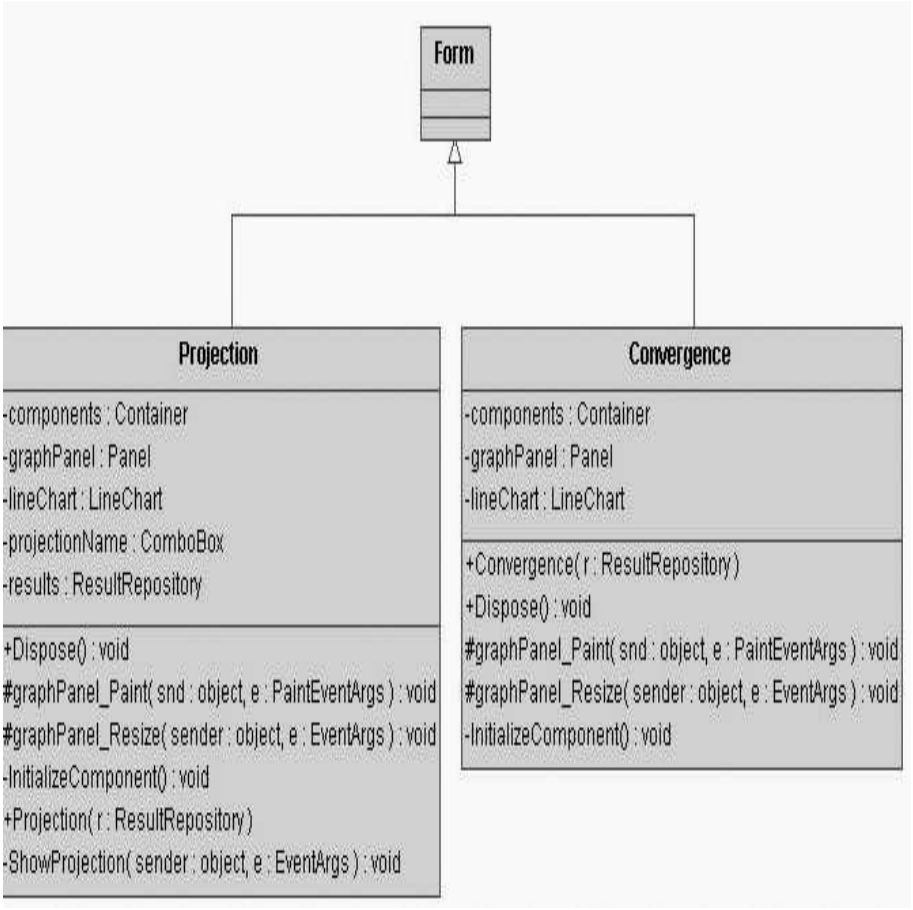


Figure 7.16. Analyzers package 2.

This package transforms result loggers data into own data and displays task results in various charts.

The class $*ChartData$ stores transformed logger data.

The abstract class $*Chart$ takes $ChartData$ as data and paints charts into the $Graphics$ objects.

$*LineChart$ draws line chart (lines could be concatenated or not).

The $*Projection$ dialog class transforms data and displays projection.

The $*Convergence$ dialog class transforms data and displays how the best obtained values depend on iteration numbers..

2.8 Result Loggers Package

Result loggers package 7.17 This package is used to access, manipulate and store results.

The class $*Result$ stores results at appropriate tasks domain point.

The $*ResultLogger$ interface is for logging results (uses method classes).

The $*ResultRepository$ interface manipulates results (uses analyzers classes).

The class $*ResultSet$ implements the $Result$ and the $ResultLogger$, stores results.

The event class $*NewValueEventArgs$ is thrown when new optimization results occurs.

The $*Minimizer$ class controls workflow of minimization, works as seperate thread and runs method with task.

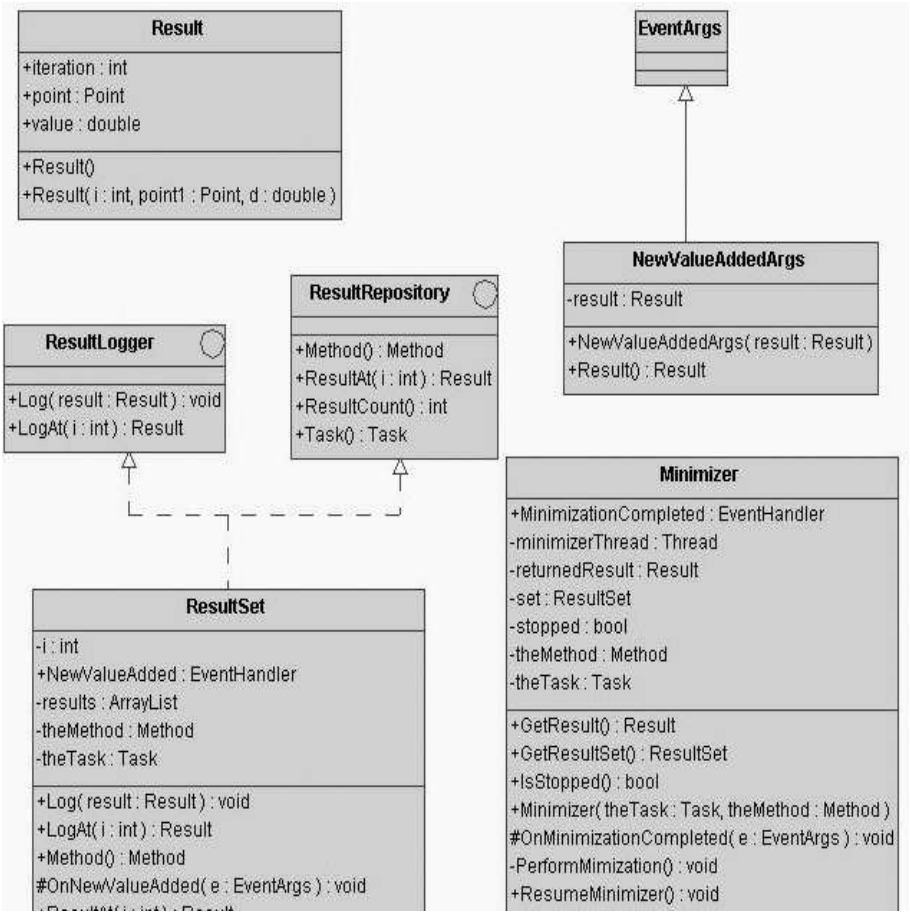


Figure 7.17. Result loggers package.

3. Developer's Guide

The GMC# system is class framework for implementing and testing custom minimization tasks and methods. The application is highly flexible and can be extended in many ways.

3.1 Adding new tasks and methods to GMC#

The sample Configuration.xml file is provided, which shows, how to set up configuration parameters of the GMC# framework. To add custom task or

method to the framework you need register information about task or method to the configuration file.

The file looks like this:

```
<configuration>
<task>
<classname>Sin</classname>
<library>GMTaskMethod.dll</library>
</task>
<task>
<classname>Square</classname>
<library>GMTaskMethod.dll</library>
</task>
<method>
<classname>Mig1</classname>
<library>GMTaskMethod.dll</library>
</method>
<method>
<classname>Bayes</classname>
<library>GMTaskMethod.dll</library>
</method>
<method>
<classname>Exkor</classname>
<library>GMTaskMethod.dll</library>
</method>
</configuration>
```

<task> tag indicates task such as Sin (sine) or Square. The GMC# framework must know your task name and library where this task is implemented. The name you provide will appear in the GMC# task list. Example:

```
<task>
<classname>Sin</classname>
<library>GMTaskMethod.dll</library>
</task>
```

Task class name is Sin and it is implemented in GMTaskMethod.dll file.

<method> tag indicates mathematical method such as Bayes or Mig1. The GMC# framework must know your method name and library where this method

is implemented. The name you provide will appear in the GMC# method list. Example:

```
<method>
<classname>Bayes</classname>
<library>GMTaskMethod.dll</library>
</method>
```

Method class name is Bayes and it is implemented in GMTaskMethod.dll file.

NOTE: the custom task or method library (dll) location must be added to PATH or the library should be in GMC# installation directory.

3.2 Implementing Minimization Tasks

In order to solve custom minimization tasks, users have to develop and compile their own classes. The task class should implement a Task interface:

```
public interface Task : Customizable
{
Domain Domain
{
get;
}
double F(Point point);
}
```

The interface Task is derived from Customizable, which requires it to be configurable by the user. For further information on how to implement the configuration of the properties, refer to the section 5. F calculates the value of the minimization function at the specified Point, which belongs to the Domain of the function. Domain returns the Domain of the minimization function, over which the function is to be evaluated. The task class should always return the same copy of the domain object, whenever domain is called. The Domain is a abstract class:

```
public abstract class Domain : Customizable
{
```

```
public double[] min;
public double[] max;
public Point defaultPoint;

public abstract string[] Dimensions
{
get;
}
public void Customize(PropertyManager propertyManager);
public double ConstraintAt(int i, Point point);
public void NormalizePoint(Point point)
public Domain();
}
```

The abstract property Dimensions should be overridden to return the descriptions of the dimensions of the domain of the minimization function. These descriptions are used in the analysis and configuration windows. Variables min, max and defaultPoint are created by Domain, but should be initialized by the derived class, as required. The Point class implements a point in a n-dimensional space of Domain:

public class Point public double[] x; public Point(Domain domain) ....

It constructs from a Dimension object and creates the required array accordingly. The Point class also has several useful methods to calculate distance between points, norms and etc.

Summary

To implement a new task the following should be done: 1. Derive a class from a class Domain
2. Provide a constructor for it to initialize the default dimension range and default point.
3. Override the property Dimensions and return descriptions of the dimensions of the domain.
4. Create a new class, which implements the interface Task.
5. Override the property Domain, which returns the single copy of the custom domain class.
6. Override the method F

Example

```
public class SinDomain : Domain
{
public override String[] Dimensions
{
get
{
return dimensions;
}
}
public SinDomain()
{
min[0] = min[1] = min[2] = -2D;
max[0] = max[1] = max[2] = 2D;
defaultPoint.x[0] = 0.0D;
defaultPoint.x[1] = 0.0D;
defaultPoint.x[2] = 0.0D;
}
static readonly String[] dimensions =
{ "X Sin Argument", "Y Sin Argument",
"Dummy" };
}

public class Sin : TaskWithAnalyzers
{
public Domain Domain
{
get
{
return (Domain)m_domain;
}
}
public double F(Point point)
{
return function != 0 ?
Math.Cos(point.x[0] * point.x[0] + point.x[1] *
point.x[1])
* multiplier :
Math.Sin(point.x[0] * point.x[0] + point.x[1] * point.x[1]) *
multiplier;
}
public Sin()
{
```

```
m_domain = new SinDomain();
multiplier = 1.0D;
function = 1;
}

private SinDomain m_domain;
public double multiplier;
public int function;
}
```

Implementing new minimization methods

GMC# can be extended by providing additional calculation methods, which might be more efficient, faster or more flexible. The method class should implement the Method interface:

```
public interface Method : Customizable
{
int Iterations();
Result Run(ResultLogger resultlogger, Task task);
}
```

The interface Method is derived from Customizable, which requires it to be configurable by the user. For further information on how to implement the configuration of the properties, refer to the section 5. Iterations returns the number of iterations, the method will do. As many of the minimization methods will have a configurable "Iterations" property, the MethodBase abstract class is implemented on top of Method interface, which overrides the iterations and customize methods. By inheriting from MethodBase only the method run has to be overridden. Run runs the method on the specified Task and writes results to the specified ResultLogger. The method should return the best result. ResultLogger gathers the generated results and stores them. The ResultLogger interface defines the following members:

```
public interface ResultLogger
{
void Log(Result result);
Result LogAt(int i);
}
```

Log should be used to log the result of every iteration of the method. Usually the class, which implements ResultLogger will update the user interface objects to reflect the progress of the method, when log is called. Some methods require the history of the minimization to be available to move on. The LogAt method returns the Result, which has been logged previously (in iteration it) using log method.

A Result class is a container of the iteration state:

```
public sealed class Result
{
public Result();
public Result(int i, Point point1, double d);

public int iteration;
public Point point;
public double value;
}
```

It holds the iteration number, the point, where the value of the Task.F was calculated, and the value of Task.F at this point.

3.3 Configuring $PropertyManager$

Most of the worker objects have to be configured by the user before usage. For example, the user may want to change the number of iterations, the method should do, or the range of the domain of the task function. A PropertyManager is used to list properties, which may be changed. A PropertyManager is a property list, which is described by the interface PropertyManager:

```
public interface PropertyManager
{
void RemoveAll();
void Add(Property property);
void Remove(Property property);
}
```

Method Add should be used to add new Property objects to the list. Currently the Property interface contains method for label retrieving:

```
public interface Property
{
string Label
{
get;
}
}
```

The GMC# framework has such Property implementations:

* FieldProperty - a single value the user can change mapped to text field control. * ChoiceProperty - several values mapped to choice control. The user selects only one value . * RangeProperty - three values: min, default, max mapped to three text fields.

The most important component of the Property is a PropertyProvider. It's an object, which knows how to retrieve the value of the property and how to store it after it has been changed by the user. The GMC# package provides a set useful property providers:

* ArrayElementProvider accesses the array of elements. * FieldProvider accesses specified attribute of some class instance.

Both providers uses .NET Framework reflection so works with array elements and class attributes of any type.

Examples

The sample code shows how to add a property named "Iterations", which is stored as attribute of the current class:

```
propertymanager.Add(new FieldProperty("Iterations",
new FieldProvider(this,
"iterations")));
```

```
Another example with ChoiceProperty:
```

```
string[] m_as = new string[] { "Sin", "Cos" };
propertymanager.Add(new ChoiceProperty("Function",
new FieldProvider(this,
"function"), m_as));
```

The user can choice from "Sin" and "Cos", when the user selects "Sin" value, if the selected "Sin" 1 is assigned to "function" attribute of the current class, and if the user selected "Cos" 2 is assigned to the attribute.

And example with RangeProperty:

```
ArrayElementProvider provider1 = new ArrayElementProvider(min,i);
ArrayElementProvider provider3 = new ArrayElementProvider(max,i);
ArrayElementProvider provider2 =
new ArrayElementProvider(defaultPoint.x,i);
RangeProperty theProperty = new
RangeProperty(Dimensions[i],provider1,provider2,provider3);

Where min, max, defaultPoint is declared in such way:

min = new double[Dimensions.Length];
max = new double[Dimensions.Length];
defaultPoint = new Point(this);
```

3.4 Writing Custom $PropertyProvider$

There is no need to write custom property provided, because GMC# framework uses reflection, so available GMC# property providers will work with any data type.

3.5 Customizing Analyzers

Now GMC# framework supports two analyzers: convergence and projection. GMC# framework does not limit analyzers classes with any restriction except: .analyzer class must has constructor which takes ResultRepository as parameter .analyzer class must inherit Form class (to show some data)

Example:

```
public class Convergence : Form
{
public Convergence(ResultRepository r) { ... }
...
}
```

GMC# has simple graphical library to draw line and points charts (system is open to new graphical library implementation). This library draws 2D charts and takes two dimensional coordinates ChartData. ChartData stores X and Y value arrays. LineChart class ChartData object as data and draws 2D chart. LineChart may be easily configured:

```
ChartData chartData = new ChartData(x_array, y_array);
lineChart = new LineChart();
lineChart.data = chartData;
lineChart.xAxisLabel = "iterations"; //x-Axis name
lineChart.yAxisLabel = "F(x)"; //y-Axis name
lineChart.xAxisStep = r.ResultCount() / 10f; //
x-Axis grid size
lineChart.yAxisStep = (float)( maxY - minY ) / 8f; //
y-Axis grid size
lineChart.xAxisMaxRange = r.ResultCount(); //x-Axis max value
lineChart.xAxisMinRange = 0f; //x-Axis min value
lineChart.yAxisMaxRange = maxY; //y-Axis max value
lineChart.yAxisMinRange = minY; //y-Axis min value
lineChart.showPoints = false; //indicates points visibility
lineChart.connectPoints = true; //
indicates lines between points visiblity
```

LineChart method: void Paint(Graphics g, Rectangle rec) takes simple graphical surface and draws line chart on it (uses only Rectangle size surface of all surface). To join task result analyzer with line chart, register paint event handler, which will forward request for repaint to line chart object.

Example:

```
class Convergence : Form
public Convergence(ResultRepository r) { ... }

//standard component initialization method
private void InitializeComponent()
{
...
this.components = new Container();
this.graphPanel = new SPanel();
this.projectionName = new ComboBox();
graphPanel.BackColor = SystemColors.ControlLightLight;
```

```
graphPanel.Size = new Size(400, 400);
graphPanel.TabIndex = 0;
graphPanel.Paint += new PaintEventHandler(graphPanel_Paint);
graphPanel.Resize += new EventHandler(graphPanel_Resize);
this.Controls.Add(graphPanel);
...
}

//repaint event handler protected void
graphPanel_Paint(object sender, PaintEventArgs e)
{
...
lineChart.Paint(e.Graphics, new Rectangle(0, 0, graphPanel.Width,
graphPanel.Height));
...
}
...
}
```

Chapter 8

JAVA JDK1.0 VERSION (GMJ0)

1. Introduction

The Global Minimizer GMJ0 (Krapauskas, 1997) implements in Java JDK1.0 the C++ global optimization software GMC. That makes GMJ0 familiar to GMC users. GMJ0 includes all the global methods. Lbayes is the only local method. Implementation of the remaining two local methods, Nlp and Flexi, is difficult. We need to introduce specific 'Constraints' objects[1] to represent constraints, in these methods. The graphical interface and the menu system both are similar to those of GMC.

All 'java' and 'class' files are in the archive on web-sites (see section 4). The applet 'gm.html' is started by a browser. The command

```
appletviewer (URL address)gm.html
```

starts the applet, if JDK1.0.x[2] is on, and one may reach the corresponding URL address. The applet optimizes the objective function defined by 'Task.java'. In the demonstration applet the 'Task.java' is sin.

Users replace an illustrative function in the file 'Task.java' by his own objective function. Then the new 'Task.class' is made by the 'javac' command.

[1]These objects are implemented in the JDK1.2 version (GMJ2).
[2]The command 'appletviewer' may not work properly, if one uses JDK1.1 or higher. That is a disadvantage of GMJ0.

Graphical possibilities of GMJ0 include the convergence, projection and numeric parts, similar to those in GMC.

2. Running GMJ0

2.1 Defining Objective

The file 'Task.java' represents the objective function. Figure 8.1 shows the illustrative example when the objective function is a sinus. The file 'Task.java' describes the Task class. The class variable NUMBER OF VARIABLES defines the number of optimization parameters. The arrays 'lowerBounds' and 'upperBounds'[3] define lower and upper bounds of optimization parameters. In the illustrative example, that means variation of parameters from minus two to plus two.

The array 'initialPoints' defines the initial approximation needed by local methods and by some global ones, for example by 'Exkor'. The numbers $-1.0\ \ 0.0\ \ 1.0$ define default values of arrays 'lowerBounds', initialPoints', and 'uppperBounds' correspondingly[4]. The 'return' value is the objective function, in the illustrative example that is the product of a sinus of the first and second variables.

2.2 Compiling and Running

One compiles when changes the objective function $f(x)$. This function is defined by in the file 'Task.java'. The compiling command is 'javac Task.java'. To run the program by a local java JDK1.0.x interpreter, one uses the command 'java Gm'. The JDK1.0.x command 'appletviewer gm.html' is for running the applet 'gm.html' from the command line. To run by browser just open the web-site and start the applet 'gm.html' of the GMJ0 system (see section 4).

2.3 Graphical Interface

Starting the applet 'gm.html', a window with 'Show' and 'Hidden' buttons appears (see Figure 8.2). One starts the main program and opens the menu

[3] zero element defines the first element and so on.
[4] Note that the constructor of class Task can be deleted if one is satisfied by the default values.

```
import java.lang.Object;
              // Task
 class Task extends Object
{
// Number of variables
public static final int NUMBER_OF_VARIABLES = 2;
public double lowerBounds[];
public double upperBounds[];
public double initialPoints[];

public Task()
{
// Initialize upper and lower bounds or initial points
 here:
lowerBounds = new double[NUMBER_OF_VARIABLES];
upperBounds = new double[NUMBER_OF_VARIABLES];
initialPoints = new double[NUMBER_OF_VARIABLES];
for ( int i = 0; i < NUMBER_OF_VARIABLES; i++ )
{
lowerBounds[i] = -5.0;
upperBounds[i] = 5.0;
}
initialPoints[0] = 4;//2;
initialPoints[1] = 0;//-4;
}
static public final double f(double x[], int n)
{
  return Math.pow(x[0],2)+x[1];
}
}
```

Figure 8.1. Example of the 'Task.java.'

window (see Figure 8.3) by clicking the 'Show' button . The menu window controls the optimization process and is similar to that of the GMC (C++ version). To hide this window press 'Hide'. To delete, select the menu item 'File, Exit' and answer "yes." Then the 'Restart' button appears, replacing the 'Show' and 'Hide'. One creates a new menu window by clicking 'Restart'. One opens the menu window from the local interpreter by the command 'java Gm.' In this window,

- $File$ provides the $Exit$ possibility.

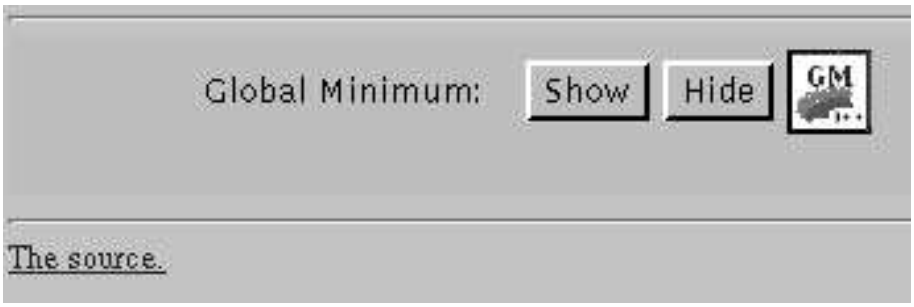


Figure 8.2. GMJ0 opening.

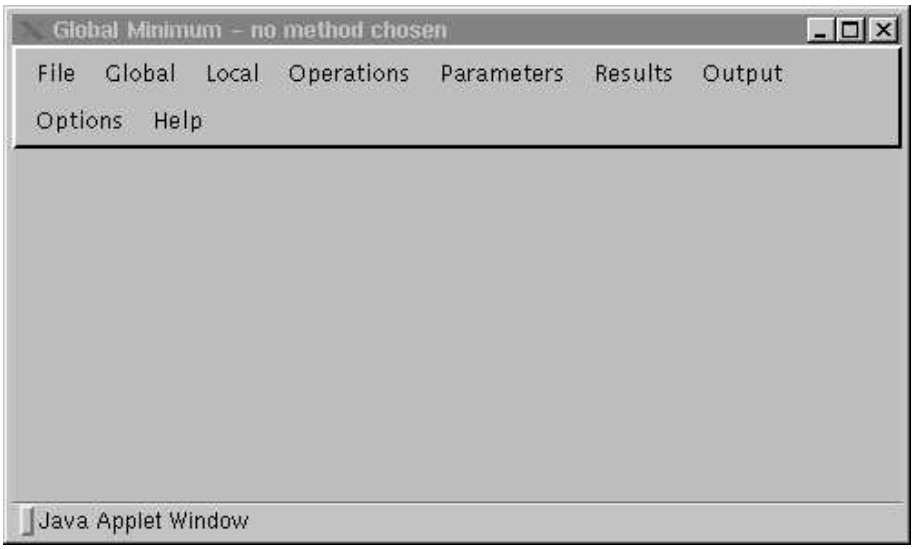


Figure 8.3. GMJ0 opening menu.

- *Global* provides the choice of global optimization methods: $Bayes1$, Unt, $Mig1$, $Exkor$, $Glopt$, $Lpmin$.

- 'Local' provides the choice of local methods: Nlp, $Flexi$, $Lbayes$ (now only $Lbayes$ is there).

- $Operation$ is for the computing control: Run is for starting or restarting, $Stop$ is for temporary stopping, and $Terminate$ ends the computing.

- $Parameters$ is just for $Set\ initial\ parameters$ that opens the parameter window. The window shape depends on the method chosen. One may

change the parameters at any moment but the new settings will be ignored until the next 'run'.

- $Results$ is for $Watch\ Results$. It opens the results window showing the best obtained objective function and the corresponding parameters.
- $Output$ is for choosing a visualization format.
  - $Numeric$ shows in numbers the current objective and parameters (see Figure 8.4).

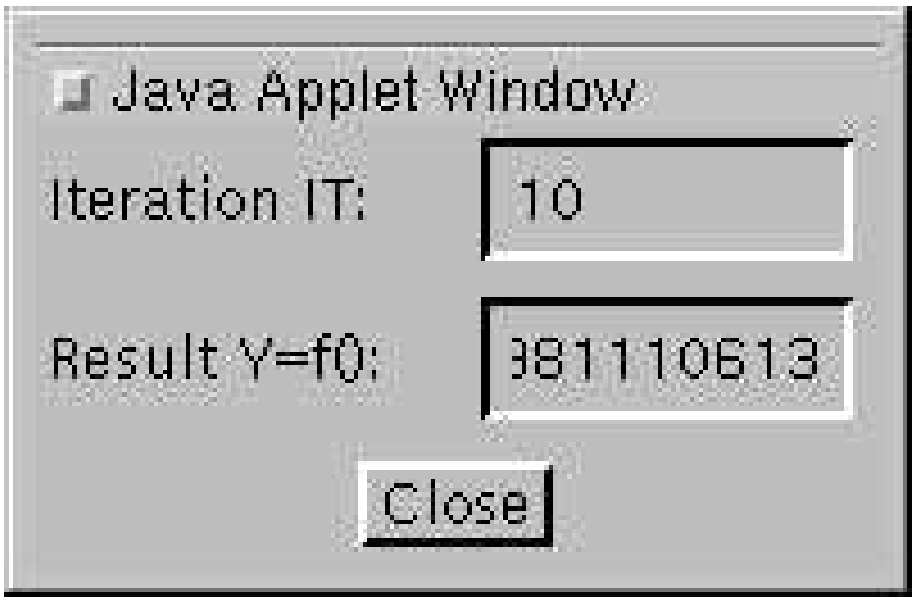


Figure 8.4. Current results.

  - $Convergence$ shows how the best obtained objective value depends on the iteration number (see Figure 8.5).
  - $Projection$ shows how the objective depends on some fixed parameter (see Figures 8.6 and 8.7).
- $Options$ is for changing the form of points on the $Projection$ window: $Cross\ points$ or $Circle\ points$.
- $Help$ just opens the window $About$.

All the visual tools takes CPU time. Therefore, closing them speeds up the computing.

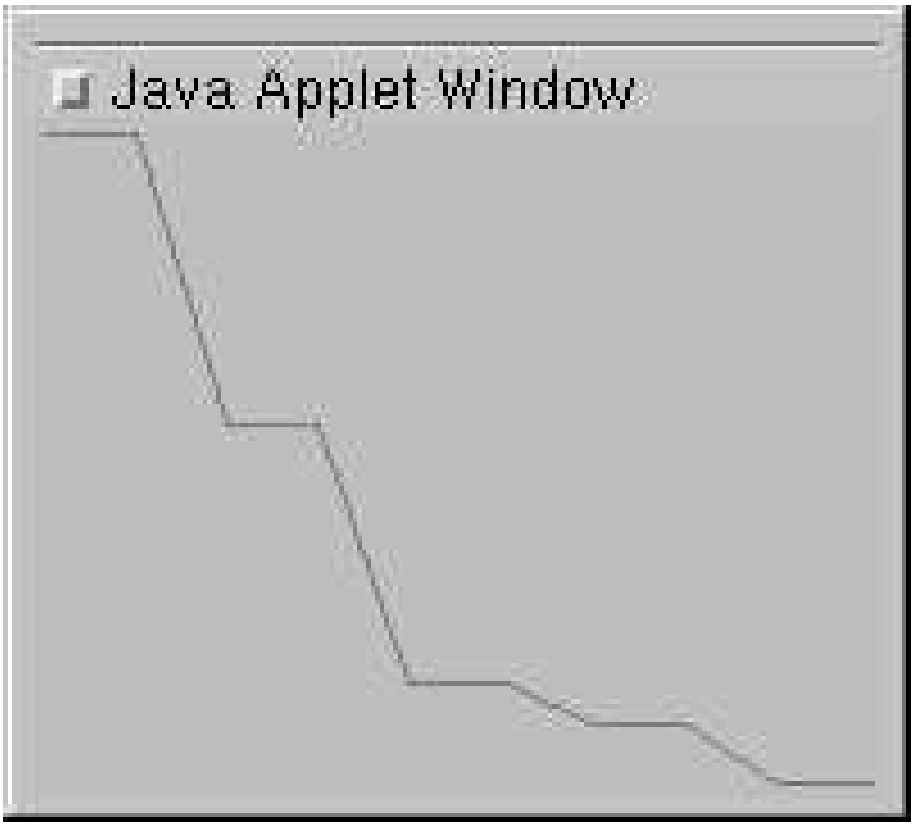


Figure 8.5. Best objective depending on iteration number.

2.4 Implementing New Method

The class of a new method is created using the 'Method' class. The minimal class of a new method is shown in Figure 8.8. The parameters of the 'Method' constructor $output$ is the object controlling a visual output. n defines computing parameters (lower and upper bounds, initial points e.t.c.). r defines the results. fm and xn define the arrays of current results.

An optimization algorithm is called by the method 'run()'. In the example, the method is 'newmethodcalculations (...)'. The name and parameters of this method may be fixed according to users needs. In the example, the internal values x, a, and b of the object created by the class 'Method' are passed. Here x, a, and b are some arrays of initial points, lower and upper bounds, respectively.

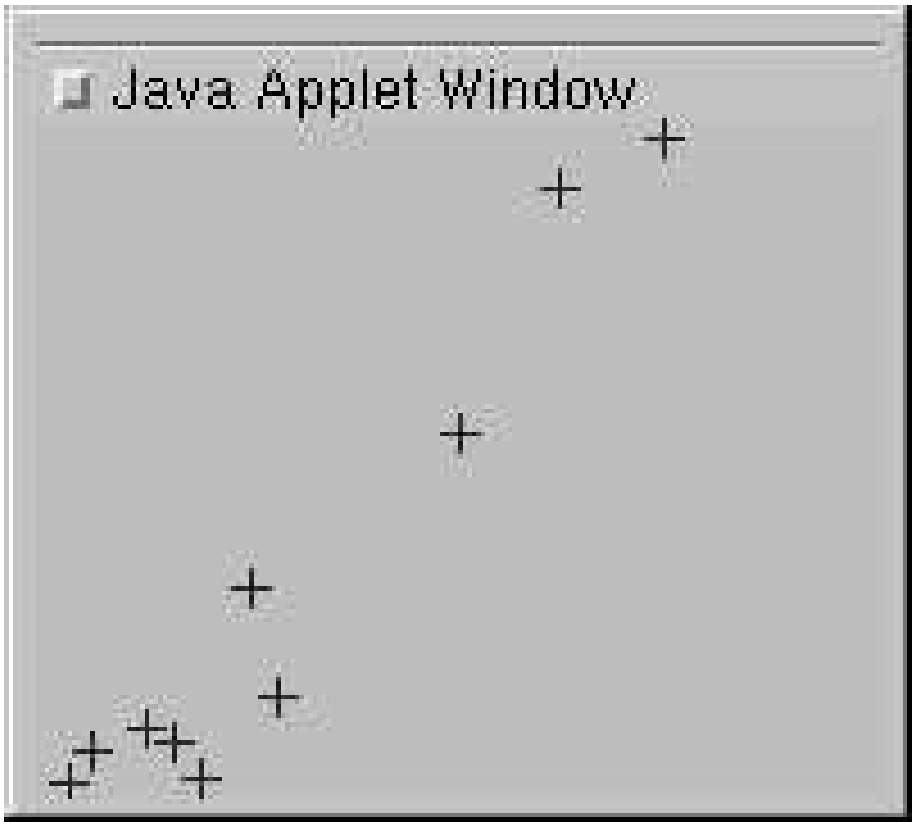


Figure 8.6. Projection 1.

n is the number of variables. it is the number of iterations. fm and xn are arrays of current results.

All these parameters are initialized by the constructor of the 'Method' class. In the 'Method' class, the internal variable r of the class 'Results' is defined. This variable is initiated by the constructor of the 'Method' class. Using the variable r, we inform the program about the change of iteration (r.iteration=new.iteration), about new minimal value (r.y=new.min), about the change of the current best point
(copy.array(r.x=new.x.n)). Here the integer 'new.iteration' shows the iteration number. The double array "new.min' defines the coordinates of the new best point. All these commands are in the method 'newmethodcalculations'. Finishing the method 'run()', the method 'Finish()' of the class 'Method' is called. Then the program returns to the initial state.

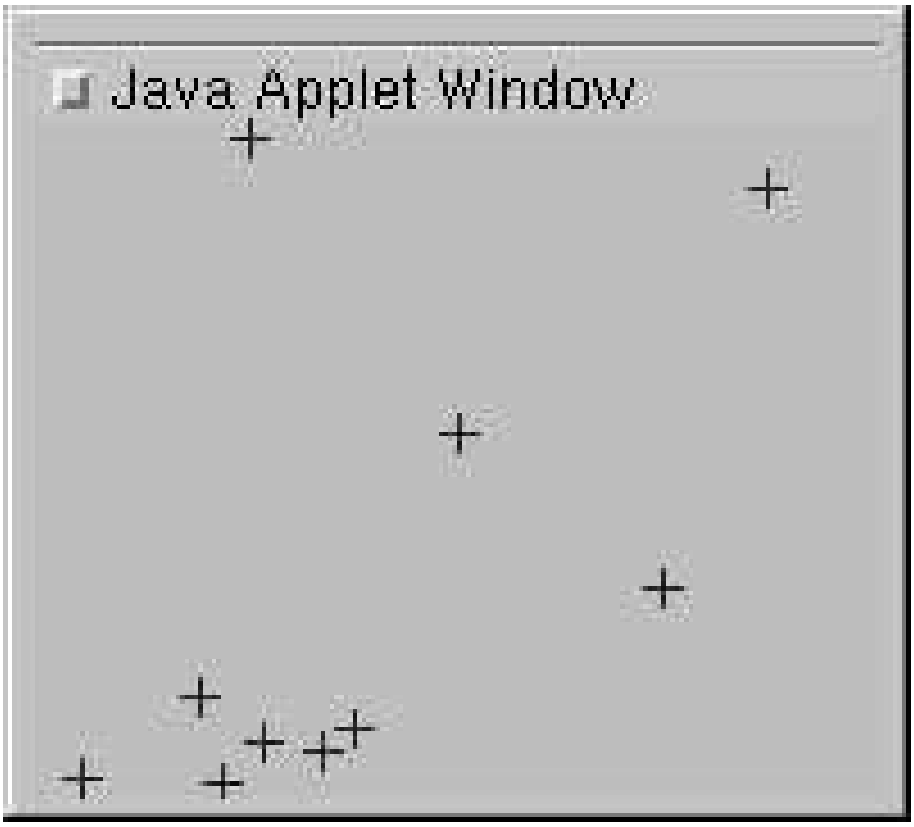


Figure 8.7. Projection 2.

One may add a new method and extend the graphical possibilities of Java global optimization software in a simpler way by using other Java versions. These versions called as GMJ1 and GMJ2 are described in the next section. Note, that GMJ1 needs JDK1.1 support and GMJ2 needs JDK1.2 support. Thus, GMJ1 does not work on some older browsers such as Netscape-3, for example. One installs the Java Developers Kit JDK1.2 to apply GMJ2.

```
import Method
class NewMethod extends Method
{
public NewMethod( Output output, Parameters p, Results r,
double fm[], double xn[])
{
super(output, p, r, fm, xn);
}

public void run()
{
newmethodcalculations(x, a, b, n, it, fm, xn);
finish();
}

private void newmethodcalculations( double x[], double a[],
double b[], int n, int it, double fm[], double xn[][])
{
//Kodas
}
}
```

Figure 8.8. Example of the minimal class of the new method.

Chapter 9

JAVA JDK1.1 AND JDK1.2 VERSIONS, GMJ1 AND GMJ2

1. Introduction

The Java JDK1.1 and JDK1.2 versions (in short, GMJ1 and GMJ2) of global optimization software are class frameworks (Grybauskas, 1998). Both GMJ versions implement:
global optimization algorithms ($METHODS$),
functions to be optimized ($TASKS$),
objects to be represented visually ($ANALYSIS$).

GMJ can be run as an applet or as a stand-alone application. One can run applet over the Web. This cannot be done by an application. Security restrictions is the disadvantage of applets. We need security permits to read/write files and to connect third parties (see Section 4.6).

These restrictions are lifted using stand-alone applications. Using stand-alone applications, the most recent Java features supported by the Java Development Kit (JDK) can be exploited. The reason is that web browsers are late adapting new Java Developer Kits (JDK), as usual. If JDK is installed, then applets can be loaded directly by appletviewers[1].

[1]For example, by the command
$appletviewer\ http://optimum.mii.lt/\ jonas/gmj2j/gmj.html.$

The JDK1.2 system provides new facilities as compared with JDK1.1. For example, Just-in-Time compilers (jitc) increase the speed several times[2]. The 'policy' files of JDK1.2 exempts trusted users from some security restrictions.

2. Starting GMJ

In this and flowing sections, the user guide of GMJ is described. First we consider the parts that are common for both JDK1.1 and JDK1.2. Then the features of JDK1.2 are mentioned.

2.1 Configuring Applet

The sample file $gmj.html$ shows how to set configuration parameters of the gmj applet. The $< applet >$ tag of GMJ1 looks like this:

```
<html>
<head>
   <META HTTP-EQUIV="Content-Language"  CONTENT="LT">
    <META HTTP-EQUIV="Content-Type" CONTENT= "text/html;
charset=windows-1257; charset=ISO-8859-13">
    <META NAME="Author" CONTENT="Modestas Grybauskas">
 <title>GMJ Applet,    Set of Methods, Tasks, and Analysis
Objects</title>
 </head>
 <body bgcolor="#E0E0E0">
 <h1>GMJ</h1>
<p>
<applet
 code="GMJext.class"
  codebase= "Lib" height=1200 width=800
  archive= "gmjg_allnew.jar"
 >
 <patram name="TASKS"
value="lt.ktu.gmj.tasks.Sin|
 Tvarka|lt.ktu.gmj.tasks.Portfolio|
lt.ktu.gmj.tasks.WalrasModel|lt.ktu.gmj.tasks.Knapsack">
<param name="METHODS" value="lt.ktu.gmj.methods.Mig1|
lt.ktu.gmj.methods.Bayes|lt.ktu.gmj.methods.Globt|
lt.ktu.gmj.methods.Unt|lt.ktu.gmj.methods.LBayes|
```

[2]These compilers are available in JDK1.1, too, starting from JDK1.1.7.

```
lt.ktu.gmj.methods.Exkor|lt.ktu.gmj.methods.Flexi|
lt.ktu.gmj.methods.Lpmin">
<param name="ANALYSIS"
 value="lt.ktu.gmj.analysis.Convergence|
lt.ktu.gmj.analysis.Spectrum|
lt.ktu.gmj.analysis.Projection|
TvarkaAnalyser|lt.ktu.gmj.analysis.WalrasProfit">
  <p>You need Java compatible browser
 to see this applet </p>
 </applet>
  </p>
 </body>
 </html>
```

In this example, $code$ specifies the applet class and should not be changed.
$codebase$ specifies the relative URL of the applet class archive file and user class files. The URL is relative to the location of the HTML file where the $< applet >$ tag resides or is an absolute path.
$archive$ lists the class archive files.
$width$ and $height$ specifies the applet size, as it appears on the HTML page.
$TASKS$ lists tasks which are available when the applet loads. Complete class names (package and class name) are separated by | symbol.
$METHODS$ lists methods which are supported when the applet loads.
$ANALYSIS$ lists visual analysis classes which are supported when the applet loads.

Here is the fragment of the GMJ2 $< applet >$ tag, which differs from GMJ1:

```
<applet
 code="lt.ktu.gmj.ui.GMJ.class"
  codebase= "Lib"
  archive= "gmj20.jar"
 >
 <patram name="TASKS"
value="lt.ktu.gmj.tasks.Sin">
<param name="METHODS" value="lt.ktu.gmj.methods.Bayes">
<param name="ANALYSIS"
 value="lt.ktu.gmj.analysis.Convergence|
lt.ktu.gmj.analysis.Spectrum|
lt.ktu.gmj.analysis.Projection">
 </applet>
```

The difference is that less methods, tasks and analysis objects are included in the GMJ2 example. Different archive file $gmj20.jar$ is used.

2.2 Configuring Stand-Alone Application

To run GMJ as a stand-alone application, the Java Runtime Environment (JRE) is required. First, $CLASSPATH$ should be configured, so that the GMJ classes can be found by the JRE. An application is started by loading the $lt.ktu.gmj.ui.GMJ$ class. Using Linux the command is:

```
java -classpath .
:/usr/local/jdk1.2/lib/classes.zip:
/home/jonas/public_html/gmj2j/Lib/gmj20.jar
lt.ktu.gmj.ui.GMJ  lt.ktu.gmj.tasks.Sin
 lt.ktu.gmj.methods.Bayes
lt.ktu.gmj.analysis.Projection
```

The application accepts three optional parameters: tasks, methods, and analysis objects. In the example, only one task Sin, one method $Bayes$ and one analysis object $Projection$ is included, for simplicity.

3. Running GMJ

3.1 Display and Control

GMJ displays a tab control that has three choices: $method$, $task$, and $operation$. The appropriate pages can be selected by clicking on the page tabs.

Figures 9.1,9.2, and 9.3 show the method , the task and the operation pages for the $Exkor$ method and the Sin task using GMJ1.
Figure 9.4 shows how the function depends on the first variable x
Figure 9.5 shows how the function depends on the second variable y. Figures 9.4 and 9.5 show that using the $Exkor$ method one gets clear projections defining how the objective function values depend on different variables. That provides a sort of "visualization" needed for the preliminary investigations. Using other

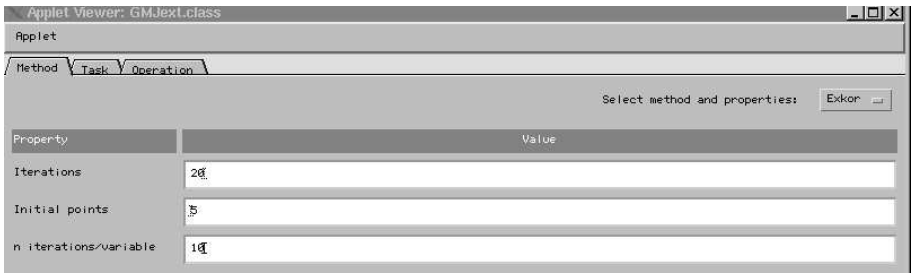


Figure 9.1. Method page with parameter fields.

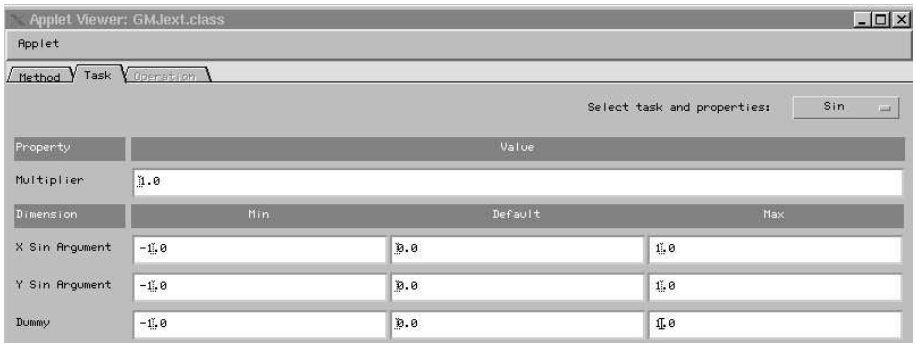


Figure 9.2. Task page with input fields.

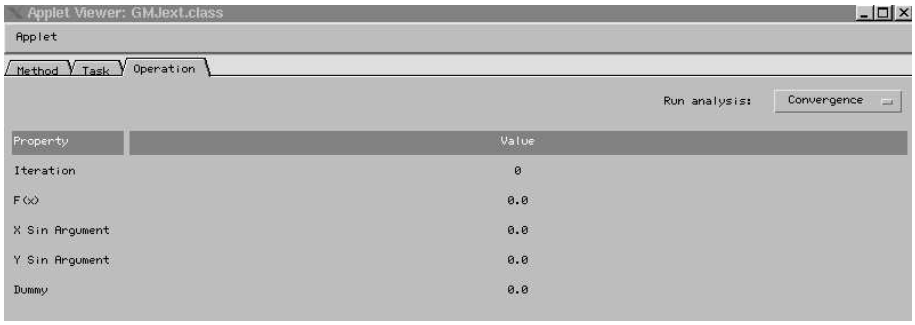


Figure 9.3. Output page with results fields.

methods, $Projection$ windows show fuzzy "projections." The reason is that variables change together.

Figures 2.3 (bottom), and 2.4 show projections obtained while optimizing the randomization parameters for the knapsack problem by the method $Bayes$.

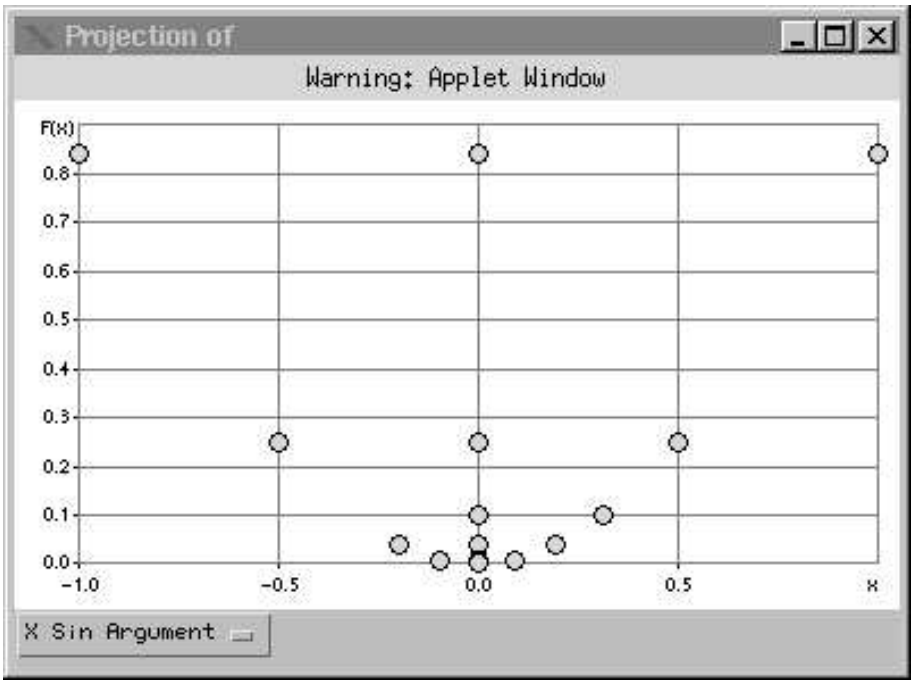


Figure 9.4. x-projection.

Figure 2.3 (top) shows how the best current value of the knapsack depends on iteration numbers.
The $Spectrum$ window[3] shows how depends the frequency of the objective function $f(x)$ values on iteration numbers. Different frequencies are represented by different colors.

3.2 Task Selection

Six tasks are installed in GMJ1.

Sin : the Sine Function.

Tvarka : the School Scheduling.

Portfolio : the Optimal Investment, the "Portfolio" problem.

[3]This window can be open by selecting the $Spectrum$ mode on the $Convergence$ menu.

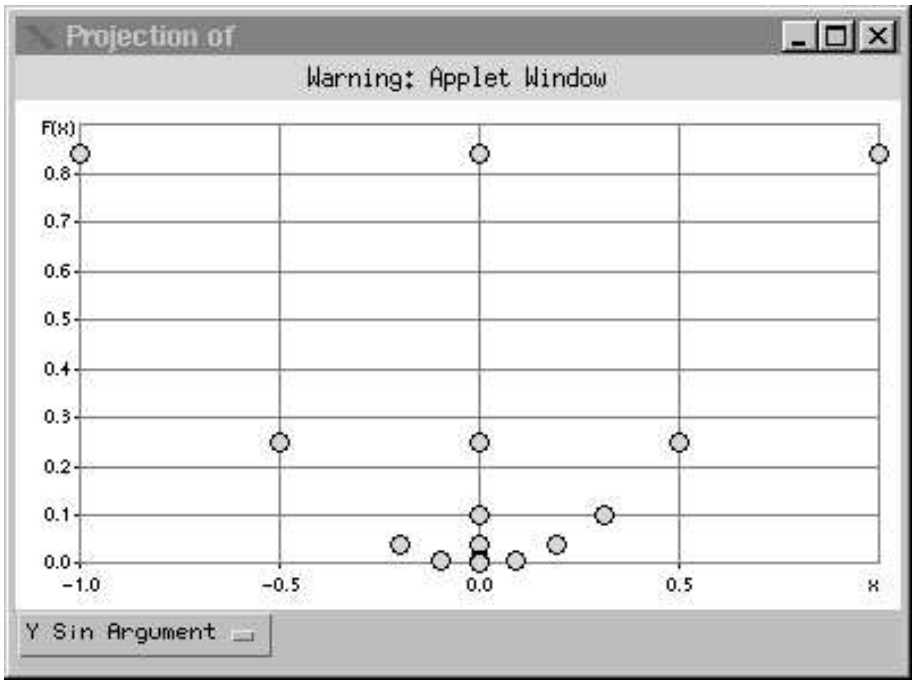


Figure 9.5. y-projection.

WalrasModel : the Walras Equilibrium.

Knapsack : the Knapsack problem.

The detail description of these examples is the subject of the next part of this book. By clicking the $tasks$ label, one opens a list of available tasks. The task is selected by clicking the label of this task.

Figure 9.2 shows the selected task Sin. After selection, the tasks implementation class is loaded and the configuration window is opened. The configuration window depends on the actual task selected.

Figure 9.2 shows the window for Sin. There are fields that define properties of tasks and the minimization domain. One can keep the default values or change them. If the entered value is invalid or out of bounds, the field turns red and the value is not stored.

3.3 Operation Control

The minimization process is controlled by the operation page (see bottom of Figure 9.3). There are three control buttons: **run, pause, stop**. The **run** button starts the optimization. The **pause** button interrupts the calculations. To abandon them completely, press the **stop** button.

3.4 Analysis of Results

The numerical optimization results are shown in the output table on Figure 9.3 displaying the following parameters:

$Iteration$ shows current iteration numbers,

$F(x)$, shows current values of the objective function $f(x)$,

$X,\ Y$, show coordinates of current points.

To select the form of graphical results, one clicks the 'Convergence' field that is at the right side on the top. Three analysis objects are installed.

$Convergence$ shows how the best obtained objective function value depends on iteration number (see Figure 2.3),

$Projection$ shows how the objective function depends on the first variable (see Figure 9.4), the variable number can be changed by clicking at the small button in the lower-left part of the window,

$Spectrum$ shows how depends the frequency[4] of objective function values on the iteration number.

3.5 Method selection

In GMJ1 five global methods are installed:

Mig1 : Monte Carlo.

[4]Defined by different colors.

Bayes : Bayesian by Mockus.

Unt : Extrapolation by Zilinskas.

Globt : Clustering by Torn.

Exkor : Bayesian coordinate search by Zilinskas.

and two local methods:

LBayes : Stochastic Approximation by Mockus.

Flexi : Simplex Nonlinear by Nelder and Mead[5].

For the detail descriptions see (Mockus et al., 1997).

One opens a list of available methods by clicking the $methods$ label of the GMJ tab control. The method is selected by clicking the label of this method. Figure 9.1 shows the selected method $Exkor$. After the selection the method's implementation class is loaded and the configuration window is opened.

The configuration window depends on the actual method selected. As usual, there is the $Iterations$ field where the number of iterations is defined, by keeping the default values or by changing them. The field $Initialpoints$ (see Figure 9.1) shows the number of observations for a preliminary survey. The preliminary survey is done by some methods including $Bayes$. If the entered value is invalid or out of bounds, the field turns red and the value is not stored.

4. Updating GMJ

Compiling is needed to add new methods, new tasks, new analysis objects. One compiles, too, to change the data which is not on the graphical interface. For example, to add a new task 'NewTask.java', a new method 'NewMethod.java', and a new analysis object
'NewAnalysis.java' one does these operations:

- compile the 'NewTask.java' , 'NewMethod.java',
  and 'NewAnalysis.java'
  making the corresponding $class$ files

[5]This is a reduced Flexi version. Only rectangular constraints are implemented.

```
javac  -verbose
 -g -classpath .:
/home/jonas/public_html/gmjgallj
/Lib/gmj_allnew.jar:
/usr/local/jdk1.1.3/lib/classes.zip
/home/jonas/public_html/gmjgallj
/lt/ktu/gmj/tasks/NewTask.java
/home/jonas/public_html/gmjgallj
/lt/ktu/gmj/methods/NewMethod.java
/home/jonas/public_html/gmjgallj
/lt/ktu/gmj/analysis/NewAnalysis.java
```

- create the additional archive file '$gmjg\_add.jar$' by including the new class files

```
 jar -cvf   /home/jonas/public_html/gmjgallj
 /Lib/gmjg_add.jar
*class lt/*
```

- update the applet 'gmjall.html' by updating the $TASKS$, $METHODS$, and $ANALYSIS$ parameters and by adding the new archive file

```
<html>
<head>
   <META HTTP-EQUIV="Content-Language"
CONTENT="LT">
   <META HTTP-EQUIV="Content-Type" CONTENT=
"text/html; charset=windows-1257; charset=ISO-8859-13">
   <META NAME="Author" CONTENT="Modestas Grybauskas">
<title>GMJ Applet,
Set of Methods, Tasks, and Analysis Objects</title>
</head>
<body bgcolor="#E0E0E0">
<h1>GMJ</h1>
<p>
<applet
 code="GMJext.class"

codebase= "Lib" height=1200 width=800

archive= "gmjg_allnew.jar  gmjg_add.jar"
 >
```

```
<patram name="TASKS"
value="lt.ktu.gmj.tasks.Sin|
Tvarka|lt.ktu.gmj.tasks.Portfolio|
lt.ktu.gmj.tasks.WalrasModel|lt.ktu.gmj.tasks.Knapsack|
lt.ktu.gmj.tasks.NewTask">

<param name="METHODS" value="lt.ktu.gmj.methods.Mig1|
lt.ktu.gmj.methods.Bayes|lt.ktu.gmj.methods.Globt|
lt.ktu.gmj.methods.Unt|lt.ktu.gmj.methods.LBayes|
lt.ktu.gmj.methods.Exkor|lt.ktu.gmj.methods.Lpmin|
lt.ktu.gmj.methods.Flexi||lt.ktu.gmj.methods.NewMethod">

<param name="ANALYSIS"
 value="lt.ktu.gmj.analysis.Convergence|
lt.ktu.gmj.analysis.Spectrum|
lt.ktu.gmj.analysis.Projection|
TvarkaAnalyser|lt.ktu.gmj.analysis.WalrasProfit|
lt.ktu.gmj.analysis.NewAnalysis">
 <p>You need Java compatible browser
 to see this applet </p>
</applet>

</p>
</body>
</html>
```

4.1 New Tasks

The task class implements the Task interface:

```
public interface Task extends Customizable
{
Domain domain ();
double f (Point pt);
};
```

The interface $Task$ is derived from $Customizable$. This means that it is configurable by users. Details are explained in Section 4.3.
A user defined method $doublef$ calculates values of the objective function at the $Point\ pt$
The class $Point\ pt$ defines the array of optimization variables belonging to the $Domain$.

The object $domain$ returns the class $Domain$, which defines a set of feasible decisions.

In GMJ1 this is a rectangular region defined by the lower and upper bounds of optimization variables[6].
The $Domain$ is an abstract class:

```
public abstract class Domain implements Customizable
{ public double min[]; public double max[];
 public Point defaultPoint;
public abstract String[] dimensions (); ..... };
```

The abstract function $dimensions$ should be overridden to return the descriptions of the domain dimensions. These descriptions are in analysis and configuration windows.
Variables min, max and $defaultPoint$ are created by the class $Domain$ and initialized by the derived class, as required. The $Point$ class implements a point in a n-dimensional set $Domain$.

```
public class Point { public double x[]; public Point
 (Domain domain) ... };
```

The class $Point$ constructs from a $Dimension$ object.

One does the following six steps to implement a new task:

- derive a class from the class $Domain$,
- provide the constructor to initialize the default dimension range and the default point,
- override the method $dimensions$ and return descriptions of the dimensions of the domain,
- create the new class, which implements the interface $Task$, or extend an abstract class AbstractTask to make the code compatible with future GMJ releases,
- override the method $domain$,
- override the method f.

[6]Note, that in GMJ2 the feasible set is given by a user defined $constraint$ function. Therefore, there are no restrictions on the shape of feasible set.

Example of Task Sine.

```
class SinDomain extends Domain
{
static final String [] dimensions=
{"X Sin Argument"};
public String [] dimensions ()
{return dimensions
}
SinDomain ()
{
min [0]=-2;
max [0]= 2;
defaultPoint.x[0]=0;
}
};
public class Sin implements Task
{
private SinDomain domain=new SinDomain ();
public Domain domain ()
{return domain;}
public double f (Point pt)
{
return Math.sin (pt.x[0]);
}
};
```

4.2 New Methods

GMJ can be extended providing additional optimization methods, which may be more efficient or more flexible. The method class should implement the $Method$ interface:

```
public interface Method extends Customizable
{
int iterations ();
Result run (ResultLogger  1, Task t);
};
```

The interface $Method$ is derived from the interface $Customizable$, which requires it to be configurable by the user. $iterations$ returns the total number of iterations.

Most of the optimization methods have the configurable $iterations$ property. Therefore, the $MethodBase$ abstract class is implemented on top of the $Method$ interface. It overrides the $iterations$ and the $customize$ methods. The method run is overridden by inheriting from $MethodBase$. The method run runs the optimization method on the selected $Task$, writes the results to the specified $ResultLogger$, and returns the best result.

The $ResultLogger$ interface gathers the generated results and stores them. The $ResultLoger$ interface defines the following members:

```
public interface ResultLogger  { void log (Result r);
ResultlogAt (int it); };
```

The method log is to log the result of each iteration of the method.
Usually, the class implementing the $ResultLoger$ updates the user interface objects reflecting the method's progress when log is called. Some methods need a history of optimization. The method $logAt$ provides that by returning the $Result$ logged previously in the iteration it using the method log. The class $Result$ contains the iteration state:

```
public final class Result  {
public int iteration; public Point
point; public double value; ..... };
```

4.3 Configuring Property Manager

Many objects are configured by users. For example, a user may want to change the number of iterations or the ranges of the optimization domain. $PropertyManager$ shows the properties, which may be changed. The $PropertyManager$ is a property list that is described by this interface:

```
public interface PropertyManager
{
void removeAll ();
void add (Property property);
void validate ();
};
```

The method add is to add new $Property$ objects to the list. The property consists of a property name or a label and a value editor. The value editor is for changing the value of the property.

The important component of the $Property$ object is the $PropertyProvider$. This is an object, which knows how to retrieve the value of the property and how to store it after the changes made by the editor. The custom value retrieval and storage methods are implemented because in Java one cannot to pass primitive objects by reference.

A set of property providers is in the GMJ package, including:

- the class $DoubleArrayProvider$ accesses the double array elements
- the class $FieldProvider$ uses the JDK1.1 Reflection classes providing access to any field of a class

Example. The following sample code shows how to add a property called the "Initial Speed" stored as the second element of the array $customData$. The initial speed can change in the range from 10 to 50 km/h. by this interface:

```
manager.add
{
new DoubleProperty
{"Initial Speed (km/h)",
new DoubleArrayProvider (customData, 1), 10, 50 }
};
```

Here is an example how to add a property named "Offset" stored in a public variable $offset$ of the calling class

```
manager.add
{
new DoubleProperty
{"Offset",
new FieldProvider (this, "offset"), 0, 1 }
};
```

Adding a property to the property manager, an editor of the appropriate type is created. If the user changes the value, the value is automatically checked, validated and stored in the second element of the array. $customData$.

Writing Custom Property Provider

When a property value is stored as a field, then a custom access class should be provided to reach that value. This class implements the $PropertyProvider$ interface requiring many functions to be overridden. Most of them are implemented in the $SimplePropertyProvider$ class, which can be used as a base.

Example. This sample code shows how to provide an access to the field $multiplier$ of the Sin class, which is $double$:

```
class SinMultiplierProvider extends SimplePropertyProvider
{
public Sin sin;
public SinMultiplierProvider (Sin _sin)
{sin = _sin;}
public Object get ()
{return new Double (sin.multiplier);}
public void set (Object value)
throws InvalidPropertyException
{sin.multiplier=((Double) value).doubleValue ();}
};
```

Note, that in Java 1.2 reflection is always available. Therefore, applying GMJ2 one applies $FieldProvider$, instead of writing custom providers.

4.4 New Analysis Objects

GMJ provides three analysis objects: $Convergence$, $Projection$ and $Spectrum$ (see section 3.4). The new tasks may need the new analysis objects. The new analyzers may be derived from any base class. They can be implemented as some windows, or they may log data to some external files. The only requirement is to implement the following $ResultAnalyser$ interface:

```
public interface ResultAnalyser
{
void prepare (ResultRepository r);
void dispose ();
};
```

To construct analysis objects the method $prepare$ is called. It asks the analysis objects to initialize. A $ResultRepository$ object is passed as a parameter. It contains information to initialize the analyzer.

The method $dispose$ is called when the analyzer is being deleted. It flush the open buffers or close associated windows. The data observer-notifier pattern is to deliver the results of real time calculations to the analysis objects. The data notifier has a list of data observers, which are notified when new data is available. To receive these notifications, the object implements the $ResultObserver$ interface and registers on the appropriate $ResultRepository$ class. The registration time is in the $prepare$ method: before leaving.

```
public interface ResultObserver
{
void newResult (ResultRepository 1);
void newProgressResult (ResultRepository 1);
void finalResult (ResultRepository 1);
void newLog (ResultRepository 1);
void oldLog (ResultRepository 1);
};
```

The method $logStarted$ signals that new calculations are going to be started.
The method $newResult$ signals, that a new result has just been added to the $ResultLoger$.
The method $newProgressResult$ signals that a method has made progress and did find the better point.
The method $finalResult$ signals that the optimization is over.
The method $logDisposed$ signals that the $ResultLog$ is depreciated and cannot be used anymore. In such a case the observers should delete all the references to the $ResultLogger$ objects.

Example. Here is a simple analyzer to print the progress to console windows

```
public class Demo
 ResultObserver  implements ResultAnalyser,
ResultObserver {
protected ResultRepository results=null};

public void prepare {ResultRepository n}
{
results=n;
```

```
results.addObserver {this};
}

public synchronized void logStarted (ResultRepository n)
{
System.out.println {"START"};
}
void newResult (ResultRepository 1) {}
void finalResult (ResultRepository 1) {}

void newProgressResult (ResultRepository 1)
{
System.ot.println {"Progress was made at iteration "
+ 1.lastResult ().iteration);
}

public void logDisposed (ResultRepository r)
{
System.out.println ("STOP");
dispose ();
}
};
```

4.5 Programming Tips

Aborting Calculations

If optimization lasts too long, a user wants to abort the calculations, as usual. When the $STOP$ button is pressed or the applet is about to exit, then an $interrupted$ flag is set for the $calculation$ thread. Therefore, $task$ and $method$ writers are advised to test for

```
if (Thread.interrupted ())
 throw new ResultLogger.AbortedEror ();
```

Implementing Interfaces

Creating a custom method or task, the class implements a corresponding interface, as usual. The better alternative is to extend an abstract class, if available. For example, "extend $AbstractMethod$" is better as compared with "implement $Method$." This makes a code more compatible with future GMJ releases.

URL Address

The original web-sites are attached to the servers 'optimum.mii.lt' or 'mockus.org'. Therefore, in some examples, the URL addresses should be updated to extract data from the local server .

On the 'Task' menu of GMJ1 global optimization software the examples are: "Portfolio," "Tvarka," and "Knapsack." In the task of school scheduling 'Tvarka.java', the URL address of the applet host is defined automatically and is shown on the graphical interface. In the task "Knapsack", some URL address is shown on the screen. It should be updated by users. In the "Portfolio" task, the URL is defined in the file 'Portfolio.java'. Therefore, recompiling is needed to extract the data.

4.6 Security Restrictions

stand-alone Application

Loading applets from their hosts, called as "applet hosts," one meets security restrictions. Applying GMJ one often needs to read/write a file on a client site or to connect to a strange host. A host that is different from the applet host is called as the "strange host."

If GMJ is loaded as a stand-alone application, one can accomplish these tasks without security restrictions. Loading GMJ as an applet one is running in a "sand-box." That means: no reading/writing files on client sites, no connections to strange hosts.

Signed Applets

A way to escape the sand-box is to sign the applet. The signature is a guarantee to others that the applet is friendly. The signature does not prevent damage, but it serves as a stamp of good intention. It identifies the responsible party, if damage does occur.

A way to guarantee that a signature is authentic is to notarize it by a Certificate Authority (CA). The Certificate Authority authenticates the identity of the applicant and then issues a signing certificate.

Anything signed with a signing certificate will bear the name of CA besides the signer name. The signature can be traced, by way of CA, to the signer.

"VeriSign,Inc." is the recognized CA. Signing certificates can be obtained at:
$http://digitalid.verisign.com/software\_publishers.html$
For an individual, it costs $40 per year for both Netscape and Internet Explorer. The prices are expected to drop. Here is a fragment of a signed applet:

```
<applet code="WriteFile.class"
ARCHIVE="signedWriteFile.jar"
<param name="CABBASE"
value="signedWriteFile.cab">
</applet>
```

The ARCHIVE parameter is parsed by Netscape. The CABBASE parameter is parsed by Internet Explorer. Different browsers use different signatures.

For example, the "Netscape" requires to change some parts of Java software, besides signing jar files.

Signing in "Sun Appletviewer" is simpler. The description is in the JDK documentation. Here a user needs the certificate and a "policy" file, where all his rights are listed.

To sign "Internet Explorer" one downloads the Microsoft Java Developer Kit. The additional information is in (Vitek and Jensen, 2000) and on web-sites:
$http://www.fastlane.net.\tilde{}tlandry/javafaq.txt$
$http://ourworld.compuserve.com/homepages/$
$jozart/article/index.html$

Note, that one should not change the existing 'gmj20.jar' file in GMJ2. Otherwise, security restrictions might be violated. Therefore, extending GMJ a user creates his own jar files[7].

[7] In the "applet.html" file one can list several jar files.

5. Features of GMJ2

5.1 Custom Result Analyzers

Visualization is a convenient way to represent the input of tasks and the results. As usual, the different tasks need different visualization. For example, considering the school scheduling (see chapter 3), one wants to see the best obtained schedule. Considering the existence of the equilibrium in the Walras model (see chapter 11), it is important to see how profit functions depend on prices, resources, and other parameters.

In such cases the output is implemented as an analysis object (see section 4.4). However, we need many analysis objects to satisfy all the tasks. To show only task-specific analysis objects, in GMJ2 the $Task$ class implements the $yzers0$ interface

```
public interface yzers 0
 extends Task
{
Class () analyzers () throws dException 0
;
};
```

The $analyzers$ function returns a class list of custom analyzers of the task.

Example. This is a sample task implementation. It states, that there exists a custom analyzer, called "Diagram".

```
class Demo extends AbstractTask implements ers 0

{
.....
public Class () analyzers () throws ClassNotFoundException
{
Class digramClass.forName ("lt.ktu.gmj.demo.Diagram");
return new Class () (diagramClass );
}
...
};
```

5.2 Domain Constraint Function

Using GMJ1, the methods consider rectangular feasible regions. These regions are bounded by lower and upper limits defined on the screen. Therefore, in GMJ1 "non-rectangular" constraints[8] are included using penalty functions. This is not always convenient.

In GMJ2 the class $Domain$ is extended to represent any non-rectangular domain. This domain includes the function.

```
public double constraintAt {Point pt}
```

This function defines a constraint value at the point pt. It returns number 1 by default (see Figure 9.6). Writing new methods in GMJ2, one checks the constraint function before calling the objective function f of the $Task$. If a method does not support constraints, one multiplies the constraint value[9] by f and minimizes the result.

Implementing a task in GMJ2, the custom domain class can be derived from the $DomainWithConstraint$ class. This enables a user to specify the constraint function at run time. $DomainWithConstraint$ adds an entry field in the task configuration panel. In this panel, any algebraic expression can be entered and compiled at run time (see Figure 9.6). Here the constraint is entered by algebraic expression $-X1^2 - X2^2 + 1$ meaning that $-x_1^2 - x_2^2 + 1 \geq 0$. Using these tools, GMJ2 users can specify any constraint function.

5.3 Video-conferencing by GMJ2

Using GMJ2 in the videoconferencing, one needs Multiple Monitor Support. One can render on multiple screens by creating $Frame$, $JFrame$, $Window$, or $JWindow$ objects with the $GraphicsConfiguration$ of the target $GraphicsDevice$.

[8] By "non-rectangular" we mean constraints defined as a system of linear and nonlinear equations and inequalities.
[9] The default value is 1.

Figure 9.6. Task page for the Sin task with one nonlinear constraint.

The new Frame constructor,
$Frame(GraphicsConfiguration)$, enables creation of a Frame object on a different screen device.

The new Window constructor,
$Window(Window, GraphicsConfiguration)$, constructs a new invisible window with the specified window as its owner and the $GraphicsConfiguration$ of a screen device.

The new $GraphicsConfiguration\ getBounds$ method returns the bounds of the $GraphicsConfiguration$ in device coordinates. If you have a virtual device, then the device coordinates returned from $getBounds$ are virtual device coordinates.

Finally, the new Component $getGraphicsConfiguration$ method returns the $GraphicsConfiguration$ with which the $Component$ was created.

In a virtual device configuration consisting of more than one physical screen device, the $GraphicsConfiguration$ objects' coordinates are relative to the virtual coordinate system. For this reason, virtual coordinates must be used when calling the $setLocation$ method of a $Frame$ or $Window$. Similarly, calling $getBounds$ of a $GraphicsConfiguration$ in a virtual device environment returns virtual device coordinates.

The following code sample creates a $JFrame$ object for each $GraphicsConfiguration$ on each screen device in the $GraphicsEnvironment$. It offsets the coordinates of the intended location of the $JFrame$ with the bounds of the $GraphicsConfiguration$ to ensure that the $JFrame$ appears on the screen of the specified $GraphicsConfiguration$.

```
  GraphicsEnvironment ge =
GraphicsEnvironment.getLocalGraphicsEnvironment();
  GraphicsDevice[] gs = ge.getScreenDevices();
                       for (int j = 0; j < gs.length; j++)
 {
GraphicsDevice gd = gs[j];
GraphicsConfiguration[] gc = gd.getConfigurations();
                           for (int i=0; i < gc.length; i++)
{
 JFrame f = new JFrame(gs[j].getDefaultConfiguration());
           Canvas c = new Canvas(gc[i]);
           Rectangle gcBounds = gc[i].getBounds();
           int xoffs = gcBounds.x;
           int yoffs = gcBounds.y;
           f.getContentPane().add(c);
           f.setSize(300, 150);
           f.setLocation((i*50)+xoffs, (i*60)+yoffs);
           f.show();
  }
   }
```

If the bounds of a $GraphicsConfiguration$ is not taken into account in this sample, a $JFrame$ would appear at location $(i * 50, i * 60)$ on the primary screen, which might be different than the screen of the specified $GraphicsConfiguration$.

For more information on how the Java 2D API supports multi-screen environments, see Rendering in a Multi-Screen Environment in the Programmer's Guide to the Java 2D API.

This applies to other examples using Java1.2.

5.4 Running GMJ2

The *constraint* function is included into the $Tasks$ page of of GMJ2 (see Figure 9.6). In this Figure the $Sinn$ task is active. There are several different tasks: $Tvarka$, $WalrasModel$, $DuelStarwars$, $FlowShop$ and four types of sine and cosine. $Sinn$ is sine with a non-linear constraint. $Sinl$ is sine with a linear constraint. $Sinc$ is sine without constraints and $Cosc$ is cosine without constraints. The task $Tvarka$ is for the school scheduling (Nemajunas and Remeikis, 1999). $WalrasModel$ is for the Walras problem and $DuelStarwars$ is for the Duel problem.

The $Methods$ menu of GMJ2 is in Figure 9.7. There are five global methods: $Mig1$, $Bayes$,$Exkor$,$Globt$, and Unt and three local methods $Shitk$, $Flexi$, and $Lbayes$. The method $Shitk$ is the Java version (Margelevicius, 1999) of the non-linear programming method by Schittkowski (Schittkowski, 1985). Names of remaining methods are the same as in GMJ1.

The $Analysis$ menu of GMJ2 is in Figure 9.8. Three analysis objects: $Convergence$, $Projection$, and $Spectrum$ are common for all tasks. Four analysis objects: $Tvarka$, $WalrasProfit$ and $DemoAnalyzer$ are task-specific. $Tvarka$ shows the initial and the optimized school schedules. $WalrasProfit$ shows how the server profit depends on prices and other parameters. The task-specific analysis objects should be seen, if corresponding tasks are on. Otherwise, they should be closed. The analysis object $DemoAnalyzer$ satisfies this condition. $DemoAnalyzer$ is the graphical interface of the duel problem. One see it, if the $DuelStarwars$ task is on. Figure 9.9 shows this.
Optimization results and the convergence line is in Figure 9.10.

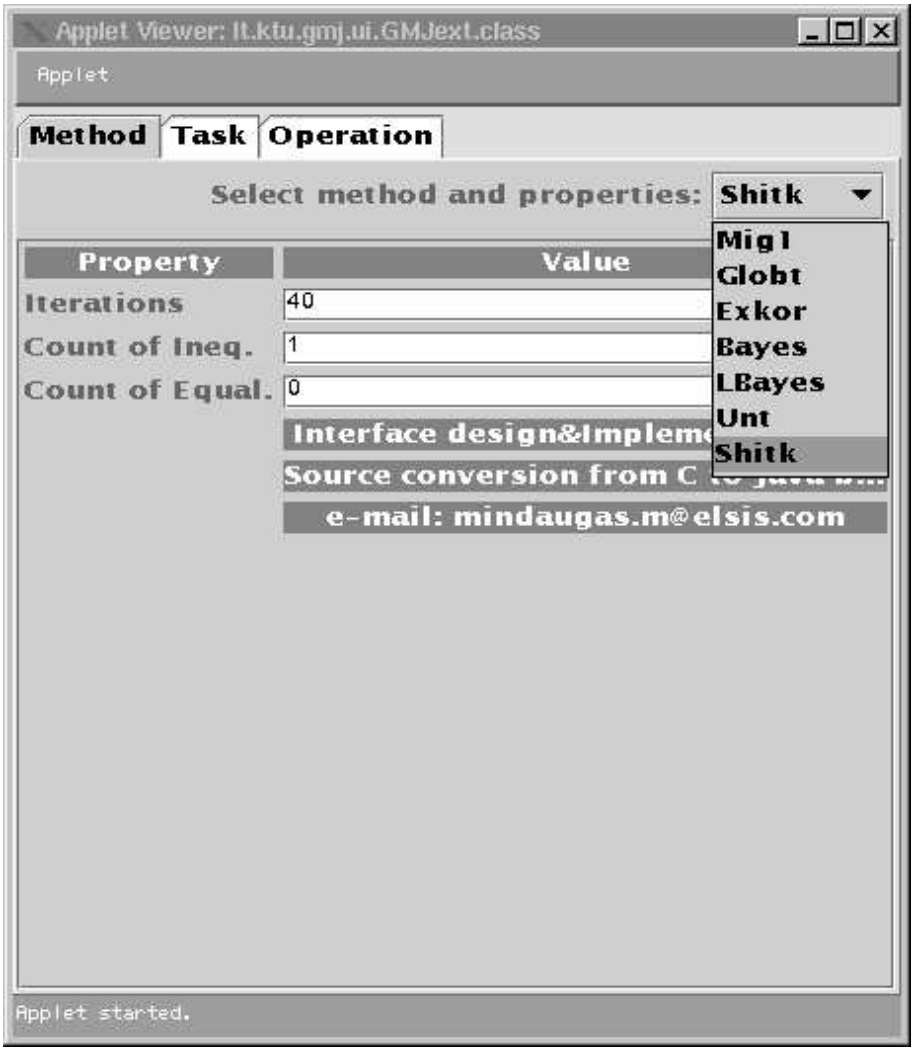


Figure 9.7. The method window of GMJ2.

In the 'Spectrum' window there is an additional option. The window shows how depends the distribution of $f(x)$ values on each variable x_i (see Figure 9.11).

Figure 9.8. The analysis menu.

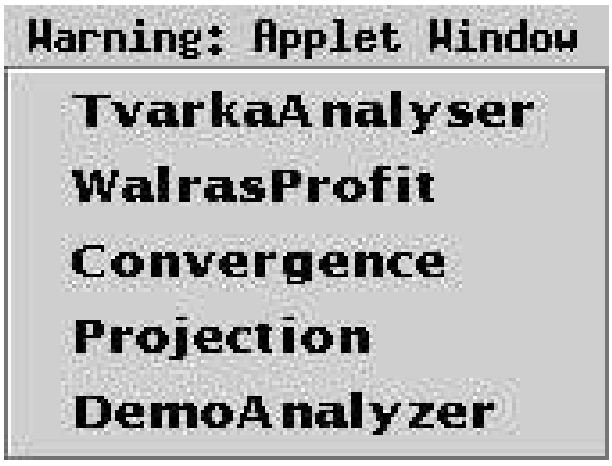


Figure 9.9. The analysis menu including the $DuelStarwars$ task.

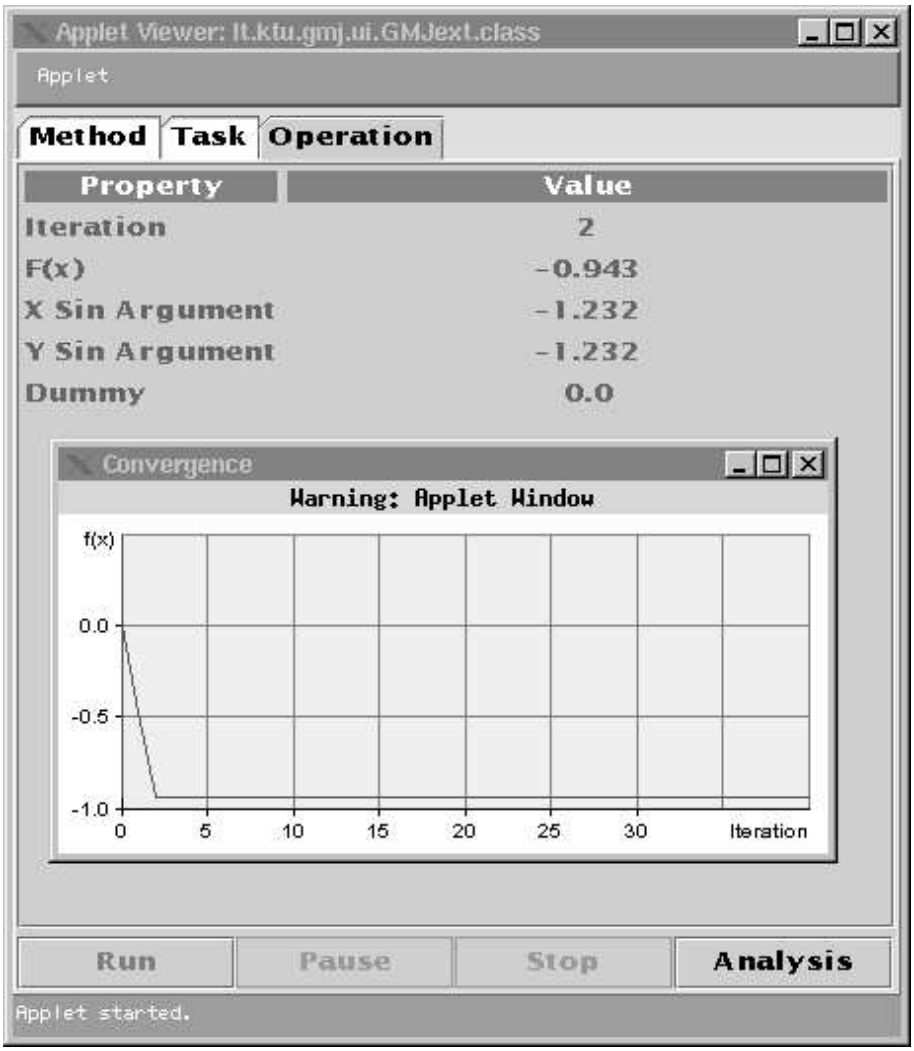


Figure 9.10. Optimization results and the convergence line.

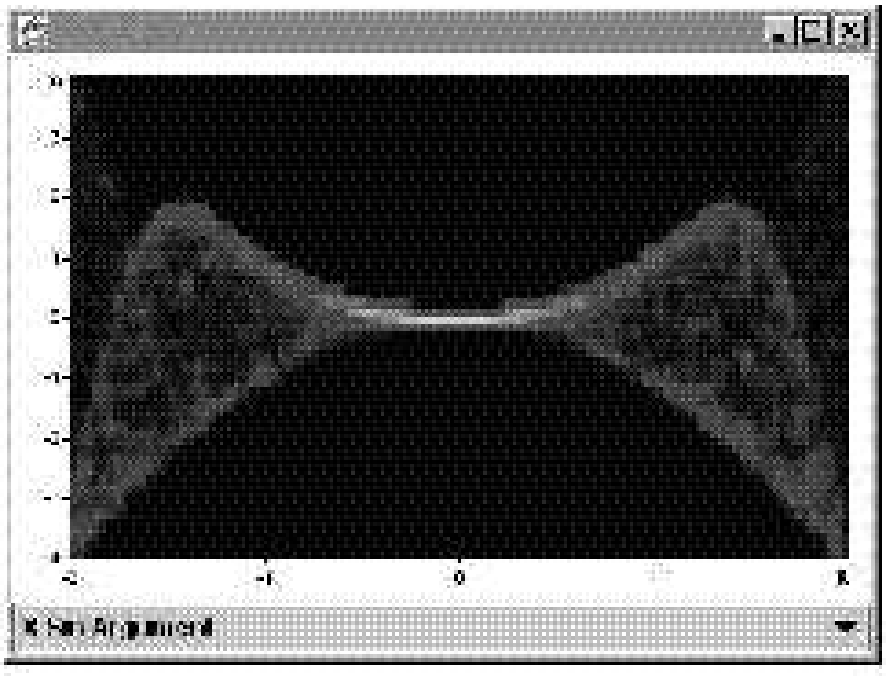


Figure 9.11. The distribution of $f(x_1, x_2)$ values with respect to x_1.

III

EXAMPLES OF MODELS

Chapter 10

COMPETITION MODEL WITH FIXED RESOURCE PRICES, NASH EQUILIBRIUM

1. Optimization Problems in Market Models

We consider two models of the theory of games and markets (Rosenmuller, 1981). The Nash equilibrium (Nash, 1950) is applied. The "market" is represented by a collection of independent servers. Each server tries to maximize its profit by setting optimal service prices and optimal server rates. The server rate is the average number of customers that would be served in nonstop operation. Thus, we can consider this rate as the server capacity.

The profits of individual servers are maximized assuming that their partners respect some agreement about the service prices and rates. We call this the "Contract-Vector." Service prices and rates obtained by maximizing individual profits of the servers transform the Contract-Vector into the "Fraud-Vector." The optimization problem is to search for such Contract-Vector that reduces the deviation of the Fraud-Vector from the Contract-Vector. That makes the fraud less relevant. The fraud is irrelevant and the Nash equilibrium is achieved, if this deviation is zero. Then servers cannot increase their profits by changing service prices and server rates agreed by the contract.

The quality of service is defined by the average time lost by customers while waiting for services. Formal expressions of other quality measures are difficult. A customer prefers the server with lesser total service cost. The total cost includes the service price plus waiting losses. A customer goes away, if the total cost exceeds a certain critical level. A flow of customers is stochastic.

Service times are stochastic, too. There is no known analytical solution for this model. The results are obtained by Monte-Carlo simulation.

The first market model considers a simple case. Here the server rate depends on a single resource that is freely obtainable at some market price. This model is the "Nash model," for short. In the second model, it is supposed that the server rates depend on several resources. Each server owns a fixed amount of single resource. Thus, the number of different resources is assumed to be equal to the number of servers. The resource limits are controlled using resource prices that are expressed as Lagrange multipliers[1]. This model is the "Walras model," for short. It is similar, but not identical, to the traditional Walras model (Walras, 1874).

In the Nash model, the servers rent their equipment at fixed price per unit of the server capacity. Therefore, in the Nash model we are looking for the equilibrium of server capacities and service prices. In the Walras model, the servers share each other resources. Therefore, here we are looking for the optimal prices charged for the shared resources. That is in addition to the search of equilibrium of server capacities and service prices.

Here the market models are to illustrate the possibilities and limitations of the optimization theory and numerical techniques in competitive environments. Both the models are developed as test functions for the Bayesian algorithms. However, these simple models may help to design more realistic ones describing the processes of competition better. Besides, the simplified competitive models are convenient tools for teaching the Operations Research and the Theory of Games and Markets.

2. Nash Model

Let us consider m servers providing the same service

$$u_i = u_i(x_1, y_1, ..., x_m, y_m) = a_i y_i - x_i,\ i = 1, ..., m, \tag{10.1}$$

where u_i is the profit, y_i is the service price, a_i is the rate of customers, x_i is the running cost, and i is the server index. Assume that a server capacity w_i is an increasing function of the running cost x_i:

$$w_i = \phi_i(x_i). \tag{10.2}$$

[1]Defined by minimizing the Lagrange function corresponding to each server.

A simple example of this function

$$w_i = k_i(1 - \exp(-k_{i0}x_i)). \tag{10.3}$$

Here k_i defines the maximal server rate and k_{i0} shows the efficiency of resource x_i. Then the total service cost

$$c_i = y_i + \gamma_i, \tag{10.4}$$

where γ_i is the average waiting cost. To simplify the expressions assume that the waiting cost is equal to an average waiting time. Suppose that arriving customers estimate the average waiting time as the relation of the number of waiting customers n_i to the server capacity w_i

$$\gamma_i = n_i/w_i \tag{10.5}$$

A customer goes to the server i, if

$$c_i \leq c_j,\ j = 1, ..., m,\ j \neq i,\ c_i \leq c_0. \tag{10.6}$$

A customer goes away, if

$$\min_i c_i > c_0, \tag{10.7}$$

where c_0 is the critical cost. The rate a of incoming consumers is fixed

$$a = \sum_{i=0}^{m} a_i, \tag{10.8}$$

where a_0 is the rate of lost customers. Conditions (10.6) and (10.7) separate the flow of incoming customers into $m + 1$ flows. This makes the problem very difficult for analytical solution. The separated flow is not simple one, even in the case when the incoming flow is Poisson (Gnedenko and Kovalenko, 1987). Thus, we need the Monte Carlo simulation, to define average rates of customers $a_i,\ i = 0, 1, ..., m$, by conditions (10.6) (10.7), and average profits $u_i,\ i = 1, ..., m$ by expression (10.1).

2.1 Customers of Different Taste

In reality, different customers value their time differently. In the simple case, one divides these customers into two categories: busy and very busy. The time coefficient of the busy customers is set to one. The time of very busy customers is denoted by $k_b > 1$. Therefore, parameters h_i are different.
For busy customers

$$h_i = y_i + \gamma_i. \tag{10.9}$$

For very busy customers

$$h_i^b = y_i + k_b.\gamma_i \tag{10.10}$$

Assume that the probability of very busy customer is p_b, then the probability of busy customer is $1 - p_b$.

3. Search for Nash Equilibrium

First we fix the initial values, the "Contract-Vector" $z^0 = (x_i^0, y_i^0,\ i = 1, ..., m)$. The transformed values, the "Fraud-Vector" $z^1 = (x_i^1, y_i^1,\ i = 1, ..., m)$, is obtained by maximizing the profits of each server i. The maximization is performed under the assumption that all partners $j \neq i$ will honor the contract $(x_j^0, y_j^0,\ j = 1, ..., m,\ j \neq i)$

$$(x_i^1, y_i^1) = arg \max_{x_i, y_i} u_i(x_i, y_i, x_j^0, y_j^0,\ j = 1, ..., m,\ j \neq i),\ i = 1, ..., m \tag{10.11}$$

Formally, condition (10.11) transforms the vector
$z^n = (x_i^n, y_i^n,\ i = 1, ..., m) \in B \subset R^{2m},\ n = 0, 1, 2, ...$ into the vector z^{n+1}. To make expressions shorter denote this transformation by T

$$z^{n+1} = T(z^n),\ n = 0, 1, 2, ... \tag{10.12}$$

One may obtain the equilibrium at the fixed point z^n, where

$$z^n = T(z^n). \tag{10.13}$$

The fixed point z^n exists, if the feasible set B is convex and all the profit functions are convex (Michael, 1976). We obtain the equilibrium directly by iterations (10.12), if the transformation T is contracting (Neuman and Morgenstern, 1953). If not, then we minimize the square deviation

$$\min_{z \in B} \parallel z - T(z) \parallel^2 . \tag{10.14}$$

The equilibrium is achieved, if the minimum (10.14) is zero. If the minimum (10.14) is positive then the equilibrium does not exist. That is a theoretical conclusion. In statistical modeling, some deviations are inevitable. Therefore, we assume that the equilibrium exists, if the minimum is not greater then modeling errors.

One can minimize deviation (10.14) by the usual stochastic approximation techniques (Ermoljev and Wets, 1988), if square deviation (10.14) is an unimodal function of z. If not, then the Bayesian techniques of global stochastic

optimization (Mockus, 1989a) should be used. The global stochastic optimization may outperform the local one in the unimodal case, too. That happens, if the noise level is great because the Bayesian global stochastic optimization methods are less sensitive to large noise levels.

The Equilibrium is stable, if transformation (10.12) is locally contracting near a fixed point (10.13). If not, then some stabilizing conditions should be introduced. The transformation $T(z)$ is referred to as locally contracting, if there exists a constant $0 \leq \alpha < 1$ such that

$$\|T(z^1) - T(z^2)\| \leq \alpha \|z^1 - z^2\| \tag{10.15}$$

for all $z^1, z^2 \in Z_\epsilon$. Here Z_ϵ is a ϵ-vicinity defined by the "natural" deviations from the equilibrium.

The same result one obtains by the following condition.

$$\min_{z \in B} \sum_i (u_i(T(z)) - u_i(z)). \tag{10.16}$$

Here the difference $u_i(T(z)) - u_i(z)$ shows the profit obtained by i-th server by breaking the contract z. We call the sum (10.16) as a fraud profit. Minimization of the fraud profit seems natural in economical terms and is convenient for computations. In these terms equilibrium means a contract where fraud is not profitable.

3.1 Existence of Nash Equilibrium

The first existence theorem is due to Nash (Nash, 1951) and dates back to 1951. Many generalizations appeared since then. Finding less and less restrictive sufficient conditions have been an active field of research (Forgo et al., 1999). The proofs of these conditions are based on the various fixed point theorems (Brouwer, 1912, Kakutani, 1941, Browder, 1968). Considering the examples of this book, we prefer simple existence conditions to the general ones. Testing the existence conditions, we express them in terms of the profit functions u instead of the operators T.

For example, the stable equilibrium exists, if the profit $u(z)$ is strictly convex function of all the components of its parameters $z \subset Z$, and Z is a convex set (Michael, 1976). In such cases, small changes of z components will not change the maximum points considerably.

The situation would be different in the non-strictly-convex cases. Here even very small change of some parameters z, may change the maximum point z^* of $u(z)$ considerably. For example, in linear cases this point may jump from minimal to maximal limits. In multi-modal cases the point z^* can jump from one local minimum to another one. These sharp changes violate the continuity of the transformation T. The continuity of T is needed in the Brouwer's fixed point theorem (Brouwer, 1912). In other theorems, such as Kakutani's (Kakutani, 1941) or Browder's (Browder, 1968), the fixed-point conditions are less restrictive. However, testing these conditions is not a trivial task.

4. Stable Coalition, Core of Games

If $m > 2$, the formation and stability of a coalition $S \subset M$ are important issues.
Denote by $S = i$ a coalition of an individual server i.
Denote by $S = (i_1, i_2)$ a coalition of two servers i_1 and i_2.
Denote by $S = (i_1, ..., i_m) = M$ a coalition of all m servers.
Assume that service prices $y_i = y_i(S)$, service rates $x_i = x_i(S)$, and shares of profit $u_i = u_i(S)$ are determined by the coalition agreement for all the coalition members $i \in S$.
Denote by $|S|$ the number of servers in the coalition S and by $u(S) = \sum_{i \in S} u_i$.
Suppose that all the remaining servers form the opposite coalition $M \setminus S$.
Assume that the profits u_i corresponds to equilibrium service prices and rates $y(S), x(S), y(S), y(M \setminus S)$. This way we consider the m-server system as a set of two-server systems, where the first server is coalition S and the second one is coalition $M \setminus S$. A coalition $S^* \subset M$ is stable if there is no dominant coalition $S \subset M$ such that

$$u_i(S) > u_i(S^*), \ for\ all\ i \in S \tag{10.17}$$

The definition of the stable coalition is related to the definition of a game core $\mathbf{C}$ (Rosenmuller, 1981).

It is well known that if

$$u(M) > \sum_{i \in M} u_i(M), \tag{10.18}$$

$$u(S) + u(M \setminus S) = u(m) \tag{10.19}$$

then the game core is empty $\mathbf{C} = \emptyset$ and there is no stable coalition S.

Inequality 10.18 means that a game is essential. Equality 10.19 defines a constant-sum game (Owen, 1968). The server system is clearly not a constant-

sum game. Thus one of two "non-existence" conditions (10.19) is not satisfied meaning that a stable coalition S^* may exist. If a stable coalition S^* exists, one may determine it by testing 10.17 for all pairs of coalitions. The game core C is empty, if for any $S' \subset M$ there is a dominant coalition S.

4.1 Example of Server Coalition

Consider a case of three servers $m = 3$.
Denote by $S_1 = i$ the coalition of an individual server i.
Denote by $S_2 = (i_1, i_2)$ the coalition of two servers i_1 and i_2.
Denote by $S_3 = (i_1, i_2, i_3)$ the monopolistic coalition of all three servers.
Subscripts $k = 1, 2, 3$ denote numbers of servers in the coalition S_k.
Consider for simplicity the symmetric case. Here profit functions u and control parameters x, y depend only on the number of servers in the coalition but not on their "names."

The symmetric case implicitly assumes equality of the control parameters and equal sharing of the coalition profit, meaning that in the case of S_3

$$u_i(S_3) = (u_i + u_j + u_l)/3, \tag{10.20}$$

$$x_i(S_3) = x_j = x_l,\ y_i(S_3) = y_i = y_j = y_l. \tag{10.21}$$

In the case of S_2

$$u_i(S_2) = (u_i + u_j)/2,\ \ x_i(S_2) = x_i = x_j,\ y_i(S_2) = y_i = y_j. \tag{10.22}$$

In the case of individual servers S_1

$$u_i(S_1) = u_i,\ x_i(S_1) = x_i,\ y_i(S_1) = y_i. \tag{10.23}$$

The monopolistic coalition S_3 is stable if $u_i(S_3) \geq u_i(S_2)$ and $u_i(S_1) \geq u_i(S_1)$.
The coalition S_2 is stable if $u_i(S_2) \geq u_i(S_3)$ and $u_i(S_2) \geq u_i(S_1)$.
The coalition S_1 is stable if $u_i(S_1) \geq u_i(S_3)$ and $u_i(S_1) \geq u_i(S_2)$.
There is no stable coalition if no of these condition are satisfied.

Chapter 11

COMPETITION MODEL WITH FREE RESOURCE PRICES, WALRAS EQUILIBRIUM

1. Walras Model

In the model called by the Nash name, the cost of a service capacity unit is supposed to be fixed. Each individual server i controls the capacity x_i and the price y_i charged for services. Therefore, the Nash model illustrates the formation of competitive service prices, if the resource prices are fixed by some large external market.

More complicated models, called by the Walras name, describe the formation of the competitive resource prices, too. In the Walras model, the capacity w_i of servers $i = 1, ..., m$ depends on the resource vector $x_i = (x_{ij},\ i, j = 1, ..., m)$ defining the consumption of different resources. Each server i owns a single resource b_i and charges a price p_i for a unit of this resource to partner-servers.

A server i controls the resource vector $x_i = x_{ij},\ j = 1, ..., m$. The component x_{ij} denotes the amount of resource b_j used by the server i. The resource balance condition means that

$$b_i = \sum_{j=1,...,m} x_{ij},\ i = 1, ..., m \tag{11.1}$$

A server i also controls the price y_i that is charged for services, as in the Nash model. Therefore we shall describe the Walras model in the terms similar to those of the Nash model (see expression (10.1)).

Assume that the server capacity w_i is an increasing function of the resource vector $x_i = (x_{ij}, j = 1, ..., m)$.

$$w_i = \phi_i(x_i). \tag{11.2}$$

A simple example of this function

$$w_i = k_i \prod_{j=1}^{m} (1 - \exp(-k_{ij} x_{ij})). \tag{11.3}$$

The resource component x_{ij} denotes the amount of resource b_j used by server i. The coefficient k_{ij} shows how useful is the resource b_j for the server i. The coefficient k_i defines the capacity limit when $x_{ij} \to \infty$. Then the profit of the i-th server:

$$u_i = u_i(x_j, y_j, p_j,\ j = 1, ..., m) = \\ a_i y_i + p_i \sum_{j \neq i} x_{ij} - \sum_{j \neq i} p_j x_{ij}. \tag{11.4}$$

Here i is the server index. a_i is the rate of customers. y_i is the service price. $x_j = (x_{jk},\ k = 1, ..., m)$ is the resource vector determining the capacity w_j of server j.
From the balance condition follows $x_{ii} = b_i - \sum_{j \neq i} x_{ji}$. Note, the profit u_i of each individual server i depends on the parameters x_j, y_j, p_j of all m servers $j = 1, ..., m$.

Here the first component $a_i y_i$ defines the income of a server i collected from customers of for its services. The second component $p_i \sum_{j \neq i} x_{ij}$ defines the sum payed by other servers using the resource b_i. The third component $\sum_{j \neq i} p_j x_{ij}$ shows the expenses for resources obtained from other servers.
In two-server cases

$$u_1 = u_1(x_{12}, y_1, p_1, x_{21}, y_2, p_2) = \\ a_1 y_1 + p_1 x_{21} - p_2 x_{12}. \tag{11.5}$$

and

$$u_2 = u_2(x_{21}, y_2, p_2, x_{12}, y_1, p_1) = \\ a_2 y_2 + p_2 x_{12} - p_1 x_{21}. \tag{11.6}$$

Here x_{11} and x_{22} are not included explicitly because they are defined by the balance conditions

$$x_{11} = b_1 - x_{21},\ x_{22} = b_2 - x_{12}. \tag{11.7}$$

By fixing the lower and upper limits
$a_{p_i}, a_{x_{ij}}, a_{y_i}, b_{p_i}, b_{x_{ij}}, b_{y_i}\ i, j = 1, ...m$ we obtain the inequalities

$$\begin{aligned} a_{p_i} \le p_i \le b_{p_i},\ a_{x_{ij}} \le x_{ij} \le b_{x_{ij}}, \\ a_{y_i} \le y_i \le b_{y_i},\ i, j = 1, ..., m \end{aligned} \tag{11.8}$$

In the Walras model the service cost, the waiting cost, and the customer behavior are similar to those of the Nash model. Namely, the service cost

$$c_i = y_i + \gamma_i, \tag{11.9}$$

where γ_i is waiting cost[1] at the server i (see expression (10.5)) . A customer goes to the server i, if

$$c_i < c_j,\ j = 1, ..., m,\ j \neq i,\ c_i \le c_0. \tag{11.10}$$

A customer goes away, if

$$\min_i c_i > c_0, \tag{11.11}$$

where c_0 is the critical cost. The rate a of incoming consumers flow is fixed:

$$a = \sum_{i=0}^{m} a_i, \tag{11.12}$$

where a_0 is the rate of lost customers.

2. Search for Nash Equilibrium

First fix a contract-vector $(x_i^0, y_i^0, p_i^0,\ i = 1, ..., m)$. Then the fraud-vector $(x_i^1, y_i^1, p_i^1,\ i = 1, ..., m)$ is obtained by maximizing the profits of each server i assuming that all other partners $j \neq i$ will respect the contract $(x_i^0, y_i^0, p_i^0,\ i = 1, ..., m)$

$$\begin{aligned} (x_i^1, y_i^1, p_i^1,) = \\ arg \max_{x_i, y_i, p_i} u_i(x_i, y_i, p_i,\ x_j^0, y_j^0, p_j^0,\ j = 1, ..., m,\ j \neq i). \end{aligned} \tag{11.13}$$

Here the profit function $u_i(x_i, y_i, p_i,\ x_j^0, y_j^0, p_j^0,\ j = 1, ..., m,\ j \neq i)$ is defined by expression (11.4). There are rectangular constraints (11.8). A server i

[1] It is. assumed for simplicity that the waiting cost is equal to an average waiting time.

optimizes only components $x_{ij},\ i \neq j$ because x_{ii} is defined by the balance condition $x_{ii} = b_i - \sum_{j \neq i} x_{ji}$.

In the two-server case

$$(x_{12}^1, y_1^1, p_1^1,) = \\ arg \max_{x_{12}, y_1, p_1} u_1(x_{12}, y_1, p_1, x_{21}^0, y_2^0, p_2^0), \tag{11.14}$$

and

$$(x_{21}^1, y_2^1, p_2^1,) = \\ arg \max_{x_{21}, y_2, p_2} u_2(x_{21}, y_2, p_2, x_{12}^0, y_1^0, p_1^0). \tag{11.15}$$

Condition (11.13) transforms vectors $z^n,\ n = 0, 1, 2, ...$ into vectors z^{n+1}, where $z^n = (x^n, y^n, p^n)$, $x^n = (x_1^n, ..., x_m^n)$, $y^n = (y_1^n, ..., y_m^n)$, and $p^n = (p_1^n, ..., p_m^n)$. Denote this transformation by T

$$z^{n+1} = T(z^n),\ n = 0, 1, 2, ... \tag{11.16}$$

Here the vector $z = (x_i, y_i, p_i, v_i\ i = 1, ..., m) \in B \subset R^{m^2+2m}$. We reach the equilibrium[2] at the fixed point z^n, where

$$z^n = T(z^n). \tag{11.17}$$

We may obtain the Nash equilibrium directly by simple iterations (11.16), if the transformation T is contracting (Neuman and Morgenstern, 1953). There are more sophisticated and efficient iterative procedures (Herings, 1994).

If the equilibrium exists but the transformation T is not contracting then one minimizes the square deviation

$$\min_{z \in B} \parallel z - T(z) \parallel^2 . \tag{11.18}$$

The equilibrium is achieved, if the minimum (11.18) is zero.

The alternative way to achieve equilibrium is by minimizing the fraud profit

$$\min_{z \in B} \sum_i (u_i(T(z)) - u_i(z)). \tag{11.19}$$

Here the difference $u_i(T(z)) - u_i(z)$ shows the profit obtained by i-th server by breaking the contract z. The advantage of (11.19) is clear economic meaning.

[2]If the equilibrium exists.

The constraints (11.8) limits the Walras model. Therefore, setting these constraints one should resolve the following contradiction. Wider limits means more computing for optimization but less restriction for the Walras model, and vice versa. A way to get around this contradiction is to start with narrow bounds (11.8). One widens them if some of these bounds obviously restrict the profit maximum.

If the equilibrium test (11.18) fails additional testing of the sufficient existence conditions is needed. It is well known, that the equilibrium exists, if the profit u is strictly convex function of parameters (y, x, p) (Michael, 1976). Therefore, the "non-strict-convexity" of this function could be a reason of the failure to obtain the equilibrium.

Expressions (11.4) and (11.6) show that profits u_i are linear functions of resource prices p_i. Linear functions are not strictly convex. In economic terms this deviation from the sufficient conditions means the server obligation to buy some fixed amount of resource x_j independently of the price p_j. In this case the owner of resource x_j maximizes the profit by increse the price p_j to infinity. Then nobody will buy it e.tc.. Thus the Nash equilibrium may not exist in this case. Besides this model is not practical because the obligation to buy certain amount of resources regardless of the price in obviosly not the realistic one.

To avoid this difficulty two versions of the "Look-Ahead" (LA) equilibrium models are considered in the following sections. In the LA models the control parameters p, y, x are devided into two groups: contract and free. For example in the model (11.13)- (11.19) all the parameters were included into the contract. The parameters that are not included into the contract are called as free. The natural example of the free parameter is resource consumptions x.

The important problem of models with free parametrs is than one must anticipate the free parameters of all competitors.

We consider estimations of free parameters based on two different assumptions: Nash (NLA) and Greedy (GLA). In the Nash (NLA) case we assume that all the servers select the free parameters at given contract parameters by conditions Nash equilibrium. In the Greedy (GLA) case servers select such free parameters that maximize their profits at fixed contract parameters. The NLA models may provide genuine Nash equilibrium if servers esimate competitors profit functuions well enough. However GLA models seems to be more realistic.

Look Ahead LA versions differs by the number of parameters to be set free. Denote by LA(p,y) the case when only the vector of resource demand x is free. Denote by LA(p) the case when both vectors x and y are free.

3. "Nash-Look-Ahead" (NLA) Equilibrium

Definition of NLA equilibrium is similar to that of Nash equilibrium. The difference is that competitors responces. are anticipated while searching for the equilibrium prices. We consider two versions of NLA.

In the first version, NLA(p) for short, both the resource consumption x_j and the service charge y_j are predicted for all servers at given resource price p. It is assumed that competing servers $j \neq i$ define parameters (x_j, y_j) by the usual Nash equilibrium conditions such as (10.14) at fixed resource prices $p = (p_i,\ i = 1, ..., m)$ [3]. For example, they increase consumption if the prices fall.

In the second version that is denoted as NLA(p,y), only the resource consumptions x_j are predicted for all servers. That means all servers j define resource demand x_j by the usual Nash equilibrium conditions at fixed both the service charges $p = (p_j,\ i = 1, ..., m)$ and service prices $y = (y_j,\ i = 1, ..., m)$. For example, they not only increase resource consumption x_j if the prices p_j fall but adapt the service charges y_j to changed resource prices as well.

Using both versions NLA while determining resource prices p one transforms linear profit functions of p_j[4] into nonlinear ones. This way one may satisfy the necessary equilibrium conditions. That is true in both the cases: NLA(p) and NLA(p,y). It seems that NLA(py) better describes the actual economical behavior of participants and demands less calculations.

3.1 First Version: NLA(p)

Search for NLA(p) Equilibrium

Defining Nash Profit. Using NLA(p) the Nash profit is defined as the profit function $U(p)$ satisfying Nash conditions at fixed resource prices p. Then the

[3] That explains the acronym NLA(p).
[4] As defined by expressions (11.4) and (11.6).

profit of the i-th server:

$$U_i(p) = u_i(x_j^* y_j^*, p_j,\ j = 1, ..., m) = \\ a_i y_i^* + p_i \sum_{j \neq i} x_{ij}^* - \sum_{j \neq i} p_j x_{ij}^*. \tag{11.20}$$

Here i is the server index. a_i is the rate of customers. $y_i^* = y_i(p)$ is the equilibrium price of services. $x_j^* = (x_{jk}^*,\ k = 1, ..., m),\ x_{jk}^* = x_{jk}(p)$ is the equilibrium resource vector that defines the capacity w_j of server j. All depend on resource prices p.
From the balance condition follows $x_{ii}^* = b_i - \sum_{j \neq i} x_{ji}^*$.
The vector (x^*, y^*) is defined by Nash conditions (11.19) for fixed resource prices p by this condition

$$\min_{z \in B} \sum_i (u_i(T(z)) - u_i(z)), \tag{11.21}$$

where $z = (x, y)$. Denote Nash equilibrium at fixed p as $z^* = z(p)$, then

$$x^* = x(p), y^* = y(p). \tag{11.22}$$

In two-server cases

$$U_1(p_1, p_2) = u_1(x_{12}^*, y_1^*, p_1, x_{21}^*, y_2^*, p_2) = \\ a_1 y_1^* + p_1 x_{21}^* - p_2 x_{12}^*. \tag{11.23}$$

and

$$U_2(p_1, p_2) = u_2(x_{21}^*, y_2^*, p_2, x_{12}^*, y_1^*, p_1) = \\ a_2 y_2^* + p_2 x_{12}^* - p_1 x_{21}^*. \tag{11.24}$$

Here x_{11}^* and x_{22}^* are not included explicitly because they are defined by the balance conditions

$$x_{11}^* = b_1 - x + 21^*,\ x_{22}^* = b_2 - x_{12}^*. \tag{11.25}$$

One solves (11.21) many times for each fixed resource price p before the equilibrium value $p = p^*$ is reached.

Defining NLA(p) Profit. First a contract-vector $(p_i^0,\ i = 1, ..., m)$ is fixed. Then the fraud-vector $(p_i^1,\ i = 1, ..., m)$ is obtained by maximizing the profits of each server i assuming that other partners $j \neq i$ keep the contract resource prices.

$$p_i^1 = arg\ \max_{p_i} U_i(p_i,\ p_j^0,\ j = 1, ..., m,\ j \neq i),\ i = 1, ..., m. \tag{11.26}$$

Here $p_j^0,\ j \neq i$ are contact resource prices of competing servers $j \neq i$.

In the two-server case

$$p_1^1 = arg \max_{p_1} U_1(p_1, p_2^0), \tag{11.27}$$

and

$$p_2^1 = arg \max_{p_2} U_2(p_2, p_1^0). \tag{11.28}$$

Condition (11.26) transforms vectors $p^n,\ n = 0, 1, 2, ...$ into vectors p^{n+1}, where $p^n = (p_i^n,\ i = 1, ..., m)$, Denote this transformation by T

$$p^{n+1} = T(p^n),\ n = 0, 1, 2, ... \tag{11.29}$$

We reach the equilibrium[5] at the fixed point p^n, where

$$p^n = T(p^n). \tag{11.30}$$

If the equilibrium exists but the transformation T is not contracting then we minimize the square deviation

$$\min_{p \in B} \parallel p - T(p) \parallel^2 . \tag{11.31}$$

The equilibrium is achieved, if the minimum (11.31) is zero.

The alternative way to achieve equilibrium is by minimizing the fraud profit

$$\min_{p \in B} \sum_i (U_i(T(p)) - U_i(p)). \tag{11.32}$$

Here the difference $U_i(T(p)) - u_i(p)$ shows the profit obtained by i-th server by deviating from the NLA equilibrium p^*.
The final equilibrium values of (x, y) are defined substituting p^* into expression (11.22)

3.2 Second Version: NLA(p,y)

Search for NLA(p,y) Equilibrium

Defining Nash Profit. Using NLA(p,y) the profit of the i-th server:

$$U_i(p, y) = u_i(x_j^* y_j, p_j,\ j = 1, ..., m) =$$

[5] If the equilibrium exists.

$$a_i y_i^* + p_i \sum_{j \neq i} x_{ij}^* - \sum_{j \neq i} p_j x_{ij}^*. \tag{11.33}$$

Here i is the server index. a_i is the rate of customers. $x_j^* = (x_{jk}^*,\ k = 1, ..., m),\ x_{jk}^* = x_{jk}(p, y)$ is the equilibrium resource vector that defines the capacity w_j of server j. All depend on prices p and y.
From the balance condition follows $x_{ii}^* = b_i - \sum_{j \neq i} x_{ji}^*$.
The vector x^* is defined at fixed prices (p, y) by this condition

$$\min_{x \in B} \sum_i (u_i(T(x)) - u_i(x)). \tag{11.34}$$

Denote Nash equilibrium at fixed price vector (p, y) as

$$x^* = x(p, y). \tag{11.35}$$

In two-server cases

$$U_1(p_1, y_1, p_2, y_2) = u_1(x_{12}^*, y_1, p_1, x_{21}^*, y_2, p_2) = \\ a_1 y_1 + p_1 x_{21}^* - p_2 x_{12}^*. \tag{11.36}$$

and

$$U_2(p_1, y_1, p_2, y_2) = u_2(x_{21}^*, y_2, p_2, x_{12}^*, y_1, p_1) = \\ a_2 y_2 + p_2 x_{12}^* - p_1 x_{21}^*. \tag{11.37}$$

Here x_{11}^* and x_{22}^* are not included explicitly because they are defined by the balance conditions

$$x_{11}^* = b_1 - x + 21^*,\ x_{22}^* = b_2 - x_{12}^*. \tag{11.38}$$

One solves (11.21) many times for each fixed price vector (p, y) before the equilibrium values $(p = p^*, y^*)$ are reached.

Defining NLA(p,y) Profit. First a contract-vector $(p_i^0, y_i^0\ i = 1, ..., m)$ is fixed. Then the fraud-vector $(p_i^1, y_i^1\ i = 1, ..., m)$ is obtained by maximizing the profits of each server i assuming that other partners $j \neq i$ keep the contract resource prices.

$$(p_i^1, y_i^1) = arg \max_{p_i, y_i} U_i(p_i, y_i\ p_j^0, y_j^0\ j = 1, ..., m,\ j \neq i),\ i = 1, ..., m \tag{11.39}$$

Here $p_j^0, y_j^0\ j \neq i$ are contact prices of competing servers $j \neq i$.

In the two-server case

$$(p_1^1, y_1^1) = arg \max_{p_1, y_1} U_1(p_1, y_1, p_2^0, y_2^0), \tag{11.40}$$

and

$$(p_2^1, y_2^1) = arg \max_{p_2, y_2} U_2(p_2, y_2 p_2^0, y_2^0). \tag{11.41}$$

Condition (11.26) transforms vectors p^n, y^n $n = 0, 1, 2, ...$ into vectors p^{n+1}, y^{n+1}, where $p^n = (p_i^n,\ i = 1, ..., m)$, and $y^n = (y_i^n,\ i = 1, ...m)$. Denote this transformation by T

$$z^{n+1} = T(z^n),\ n = 0, 1, 2, ... \tag{11.42}$$

where $z = (p, y)$. We reach the equilibrium[6] at the fixed point p^n, where

$$z^n = T(z^n). \tag{11.43}$$

If the equilibrium exists but the transformation T is not contracting then we minimize the square deviation

$$\min_{z \in B} \parallel z - T(z) \parallel^2 . \tag{11.44}$$

The equilibrium is achieved, if the minimum (11.44) is zero.

The alternative way to achieve equilibrium is by minimizing the fraud profit

$$\min_{z \in B} \sum_i (U_i(T(z)) - U_i(z)). \tag{11.45}$$

Here the difference $U_i(T(z)) - u_i(z)$ shows the profit obtained by i-th server by deviating from the NLA equilibrium z^*.
The final equilibrium values of x^*) are defined substituting (p^*, y^*) into expression (11.35)

4. "Greedy-Look-Ahead" (GLA) Equilibrium

Definition of GLA equilibrium is similar to that of NLA because in both the cases competitors responces. are anticipated while searching for the equilibrium prices. The difference is that the 'greedy' servers select the free parameters not by equilibrium conditions but by maximal profit. We consider only one version

[6] If the equilibrium exists.

GLA(py) where the free parameter is a vector of resource consumption x. The resource vector x is predicted for all servers assuming that each server defines resource demand by maximizing profit at fixed both the resource and service price vectors p and y .

Using GLA(py) one transforms linear profit functions of p_j[7] into nonlinear ones. This way one may satisfy the necessary equilibrium conditions if the profit functions are convex. That is true in both the cases: NLA and GLA. In this sense both versions are equivalent. Thus one may select a version that better describes the actual economical behavior of participants.

Both the NLA and the GLA approaches is based on the important tacit assumption that servers know profit functions of their competitors. This is not true as usuall. However, that is a price one pays for making profit functions strictly convex. This is needed to satisfy necessary equilibrium conditions. The price is not so great when servers know at least some approximation of competitor profit functions and behavior

Search for GLA(p,y) Equilibrium

Defining Nash Profit. Using GLA(p,y) the profit of the i-th server:

$$U_i(p,y) = u_i(x_j^* y_j, p_j,\ j = 1,...,m) = \\ a_i y_i^* + p_i \sum_{j \neq i} x_{ij}^* - \sum_{j \neq i} p_j x_{ij}^*. \tag{11.46}$$

Here i is the server index. a_i is the rate of customers. $x_j^* = (x_{jk}^*,\ k = 1,...,m)$, $x_{jk}^* = x_{jk}(p,y)$ is the greedy resource vector that defines the capacity w_j of server j and is obtained by maximizing the profit function U_i at given contract prices p and y.
From the balance condition follows $x_{ii}^* = b_i - \sum_{j \neq i} x_{ji}^*$.
At fixed prices (p,y) the vector x^* is defined by these conditions

$$\max_{x_{ij} \in B} U_i(p,y),\ i = 1,...,m,\ i \neq j. \tag{11.47}$$

Denote the Greedy resource consumption vector at fixed price vectors (p,y) as

$$x^* = x(p,y). \tag{11.48}$$

[7] As defined by expressions (11.4) and (11.6).

In two-server cases

$$U_1(p_1, y_1, p_2, y_2) = u_1(x^*_{12}, y_1, p_1, x^*_{21}, y_2, p_2) = a_1 y_1 + p_1 x^*_{21} - p_2 x^*_{12}. \quad (11.49)$$

and

$$U_2(p_1, y_1, p_2, y_2) = u_2(x^*_{21}, y_2, p_2, x^*_{12}, y_1, p_1) = a_2 y_2 + p_2 x^*_{12} - p_1 x^*_{21}. \quad (11.50)$$

Here x^*_{11} and x^*_{22} are not included explicitly because they are defined by the balance conditions

$$x^*_{11} = b_1 - x + 21^*, \; x^*_{22} = b_2 - x^*_{12}. \quad (11.51)$$

One solves (11.21) many times for each fixed price vector (p, y) before the equilibrium values $(p = p^*, y^*)$ are reached.

Defining GLA(p,y) Profit. First a contract-vector $(p^0_i, y^0_i \; i = 1, ..., m)$ is fixed. Then the fraud-vector $(p^1_i, y^1_i \; i = 1, ..., m)$ is obtained by maximizing the profits of each server i assuming that other partners $j \neq i$ keep the contract resource prices.

$$(p^1_i, y^1_i) = arg \max_{p_i, y_i} U_i(p_i, y_i \, p^0_j, y^0_j \; j = 1, ..., m, \; j \neq i), \; i = 1, ..., m \quad (11.52)$$

Here $p^0_j, y^0_j \; j \neq i$ are contact prices of competing servers $j \neq i$.

In the two-server case

$$(p^1_1, y^1_1) = arg \max_{p_1, y_1} U_1(p_1, y_1, p^0_2, y^0_2), \quad (11.53)$$

and

$$(p^1_2, y^1_2) = arg \max_{p_2, y_2} U_2(p_2, y_2 p^0_2, y^0_2). \quad (11.54)$$

Condition (11.26) transforms vectors $p^n, y^n \; n = 0, 1, 2, ...$ into vectors p^{n+1}, y^{n+1}, where $p^n = (p^n_i, \; i = 1, ..., m)$, and $y^n = (y^n_i, \; i = 1, ...m)$. Denote this transformation by T

$$z^{n+1} = T(z^n), \; n = 0, 1, 2, ... \quad (11.55)$$

where $z = (p, y)$. We reach the equilibrium[8] at the fixed point p^n, where

$$z^n = T(z^n). \tag{11.56}$$

If the equilibrium exists but the transformation T is not contracting then we minimize the square deviation

$$\min_{z \in B} \parallel z - T(z) \parallel^2 . \tag{11.57}$$

The equilibrium is achieved, if the minimum (11.44) is zero.

The alternative way to achieve equilibrium is by minimizing the fraud profit

$$\min_{z \in B} \sum_i (U_i(T(z)) - U_i(z)). \tag{11.58}$$

Here the difference $U_i(T(z)) - u_i(z)$ shows the profit obtained by i-th server by deviating from the NLA equilibrium z^*.
The final equilibrium values of x^*) are defined substituting (p^*, y^*) into expression (11.35)

5. Monte-Carlo Simulation

5.1 Search for Equilibrium

The analytical solution of both the described market models is not practical. Therefore, we briefly consider the basic steps of an algorithm of the statistical simulation using Monte-Carlo techniques. The algorithm implements two basic tasks:

- generates the next event time t,
- updates the state of queuing system defined by the vector of waiting customers $n(t) = (n_i(t),\ i = 1, ..., m)$,
- updates the vector $h(t) = (h_i(t),\ i = 1, ..., m)$ of the service cost including the money charged and the time lost.

There are $2m + 2$ types of event times t:

[8] If the equilibrium exists.

- the time t when a customer arrives into the system,
- the time t when a customer arrives into the i-th server, $i = 1, ..., m$,
- the time t when a customer departs from the i-th server,
- the time t when a customer abandons the service (departs from the system without being served).

Here $i = 1, ..., m$. The system state is updated at each event time t. Two vectors define the system state:

- a vector $n = n(t)$ with m components $n = (n_1, ..., n_m)$, where $n_i = n_i(t)$ shows the number of customers waiting for the service of the i-th server,
- a vector $h = h(t)$ with m components $h = (h_1, ..., h_m)$, where $h_i(t) = y_i + \gamma_i,\ \gamma_i = n_i(t)/w_i$ shows the total customer expenses, y_i is money charged for the service, and γ_i is the time lost waiting for the service of the i-th server[9].

There are no state changes between events. The basic steps of the Monte Carlo algorithm:

1 fix the zero event time $t = t^0 = 0$ when the first customer arrives,

2 define the zero state vector n^0 by the condition: $n_i^0 = 0,\ i = 1, ..., m$ and the zero state vector h^0 by the condition : $h_i^0 = y_i = 0,\ i = 1, ..., m$ because there are no customers waiting for service yet,

3 define the next arrival into the system by the expression

$$\tau_a = -1/a\ ln(1 - \eta) \tag{11.59}$$

where η is a random number uniformly distributed in the interval [0,1],

4 chose the best server i^0 for the first customer by the condition $i^0 = arg\ \min_{i=0,...,m} h_i$ where $h_i = y_i$, because $\gamma_i = 0,\ i = 1, ..., m$ since there are no customers waiting yet, $i^0 = 0$ means that the customer abandons the service,

[9]For simplicity, it is assumed that "time-is-money" and that a unit of time cost a unit of money, in the real life cases the corresponding coefficients should be included

5 define the time of event when the first customer will be served by the server i^0 using the expression

$$\tau_{i^0} = -1/x_{i^0}\ ln(1-\eta), \tag{11.60}$$

6 define the next event t^1 by comparing the arrival time τ_a and the service time τ_{i^0} :
if $\tau_a < \tau_{i^0}$ then $t^1 = \tau_a$,
if $\tau_a > \tau_{i^0}$ then $t^1 = \tau_{i^0}$,

7 define the system state at the next event t^1:
if $t^1 = \tau_a$ then $n_{i^0} = 1$ and $n_i = 0,\ i = 1, ..., m,\ i \neq i^0$,
consequently $h_{i^0} = y_{i^0} + 1/w_{i^0}$ and $h_i = y_i,\ i = 1, ..., m,\ i \neq i^0$,
if $t^1 = \tau_{i^0}$ then $n_i = 0,\ i = 1, ..., m$, and $h_i = y_i,\ i = 1, ..., m,\ i \neq i^0$.

Definition of later events and system states is longer but the main idea remains the same. For illustration, we update the fourth step for the nth customer:

- chose the best server i^0 for the nth customer by the condition
  $i^0 = arg\ \min_{i=0,...,m}\ h_i$,
  where $h_i = y_i + \gamma_i$, $\gamma_{i^0} = n_i/\omega_{i^0}$, and n_i is the number of customers waiting for server i.

The algorithm can be directly adapted to the Monte-Carlo simulation of the Nash model with two servers, too. For example, that can be done this way:
- set to unit both the resource charges $p_i = 1,\ i = 1, 2$,
- set to zero resources exchanges $x_{21} = x_{12} = 0$,
- assume that $x_{11} = x_1,\ x_{22} = x_2$, where variables $x_1,\ x_2$ are from expression (10.1).
If the number of severs $m > 2$ then some modification of the described algorithm is needed.

5.2 Modeling Customers of Different Taste

If the customers value their time differently (see Section 2.1) then the fourth step of the algorithms should be changed accordingly. For illustration, here is the fourth step for the second customer :

- if a random number $\eta < p_b$,
  regard the nth customer as very busy one,

chose the best server i^0 for this customer by the condition
$i^0 = arg\min_{i=0,\ldots,m} h_i^b$,
where $h_i^b = y_i + k_b\gamma_i$, $\gamma_{i^0} = n_i/\omega_{i^0}$, and n_i is the number of customers waiting for server i,
if a random number $\eta < p_b$,
regard the customer as busy one,
and chose the best server i^0 by the condition
$i^0 = arg\min_{i=0,\ldots,m} h_i$,
where $h_i^b = y_i + \gamma_i$, $\gamma_{i^0} = n_i/\omega_{i^0}$.

5.3 Testing Equilibrium Conditions

In the Monte-Carlo simulation, equilibrium tests (11.18) should be relaxed by accepting some simulation error ϵ:

$$\min_{z\in B} \parallel z - T(z) \parallel^2 \leq \epsilon, \tag{11.61}$$

The alternative way to achieve equilibrium is by minimizing the fraud profit (11.19). Then the test is

$$\min_{z\in B} \sum_i (u_i(T(z)) - u_i(z)) \leq \epsilon \tag{11.62}$$

To test the convexity of profit functions, some smoothing is desirable. The smoothing eliminates the random deviations due to Monte-Carlo simulation. Both the convolution and the Wiener filters are applied for smoothing the profit functions (possibly multi-modal). The convolution filter defines the function at some fixed point as an average of values in the neighborhood of this point. The more sophisticated Wiener filter is implemented, too.

5.4 Wiener Filter

If the objective function $f(x)$ is defined by Monte Carlo simulation, some noise is present. That means that one observes the sum

$$\phi(x) = f(x) + \xi, \tag{11.63}$$

where ξ is a random number called the noise.

If finding the optimum of a convex function $f(x)$ is the only goal, we can apply some stochastic optimization algorithms (Ermoljev and Wets, 1988). These

algorithms converge to the optimum of $f(x)$ by filtering the noise during the optimization process.

To test properties of $f(x)$, such as convexity, unimodality e.t.c., we need specific smoothing algorithms that eliminate false local optima. In one-dimensional cases, a convenient smoothing function is the conditional expectation of the Wiener process with noise (Kushner, 1964b, Senkiene, 1980). It is assumed that the optimization parameter $x \in [0,1]$, the Wiener parameter is a unit, and the noise ξ is Gaussian with zero mean and variance S_i at the points $x^i \in [0,1],\ i = 1,...,n$. Then the conditional expectation $\mu_k = \mu_n(x^k)$ of the objective function $y_k = f(x^k)$ at some fixed point x^k with respect to the observations results $y_i = \phi(x^i),\ i = 1,...,n$, can be expressed this way

$$\mu_k = \frac{\sum_{i=1}^{k} b_i y_i + \frac{b_k}{c_k} \sum_{i=k+1}^{n} c_i y_i}{\sum_{i=1}^{k} b_i + \frac{b_k}{c_k} \sum_{i=k+1}^{n} c_i}. \tag{11.64}$$

Here

$$b_1 = 1,\ B_1 = 1, \tag{11.65}$$

$$b_2 = \frac{S_1^2 + r_{1,2}}{S_2^2},\ B_2 = B_1 + b_2, \tag{11.66}$$

.....

$$b_k = \frac{S_{k-1}^2 b_{k-1} + r_{k,k-1} B_{k-1}}{S_k^2},\ B_k = B_{k-1} + b_k, \tag{11.67}$$

$$c_n = 0, \tag{11.68}$$

$$c_k = \frac{S_{k+1}^2 c_{k+1} + r_{k,k+1} C_{k+1}}{S_k^2}, C_{k+1} = \sum_{i=k+1}^{n} c_i, \tag{11.69}$$

$$r_{i,j} = |x^i - x^j|. \tag{11.70}$$

It is convenient to assume that

$$S_i = S,\ i = 1,...,n, \tag{11.71}$$

where S can be considered as a smoothing parameter.

If $S = 0$ then no smoothing occurs. The smoothing function (the conditional expectation) is the piece-wise line connecting the observed points (see Figure 1.1).
If S is large then one obtains a horizontal line corresponding to the average value of observed values y_i. That means a sort of "total smoothing." In modeling the yield of differential amplifiers, the best smoothing was achieved at $S = 10$ (Mockus et al., 1997).

Figure 11.1 shows how the first server profit u_1 depends on the price p_1 charged for its resources. There are two samples of the same relation. They show the differences between two samples of random arrival times of fifty customers.
The buttons 'smooth' and 'wiener' on the right side are for switching on these filters. The button 'smooth' is for the convolution filter. The button 'wiener' is for the Wiener filter.
The field denoted by 'S' at the top right corner, defines the smoothing parameter S of the Wiener filter (see expression (11.71).
One can increase the level of smoothing by pressing the 'smooth' or 'wiener' buttons repeatedly.
The 'refresh' button repeats the Monte-Carlo simulation of the same profit function.

Figures 11.2 show the third sample of the 'refreshed' unsmoothed graph and the same sample smoothed by the convolution filter.

The underlying profit function is the same in all the Figures, from 11.1 up to 11.2. The repeated simulation defines different graphs because of the simulation errors. After smoothing, the level of these errors is lower.

6. Software Example

Java j2sdk implementation (Treigys, 2003) and (Sviderskaite, 2003) of the Walras model is on web-sites (see section 4).

Figure 11.3 shows the input page. The method 'Bayes1' is set. The parameters:
the number of initial iterationslt is 5,
the number of iterationsit is 10,
the stocks of server resources b_1 and b_2 is 0.7
the "run-away" threshold $C0$ is 20
the customer rate A is 20,
the number of time units M is 5
the efficiency of all servers z_{ij} is 1.0
the accuracy is 20 %
the lower bounds min are 0.0,
the upper limits max are 30.0,
t

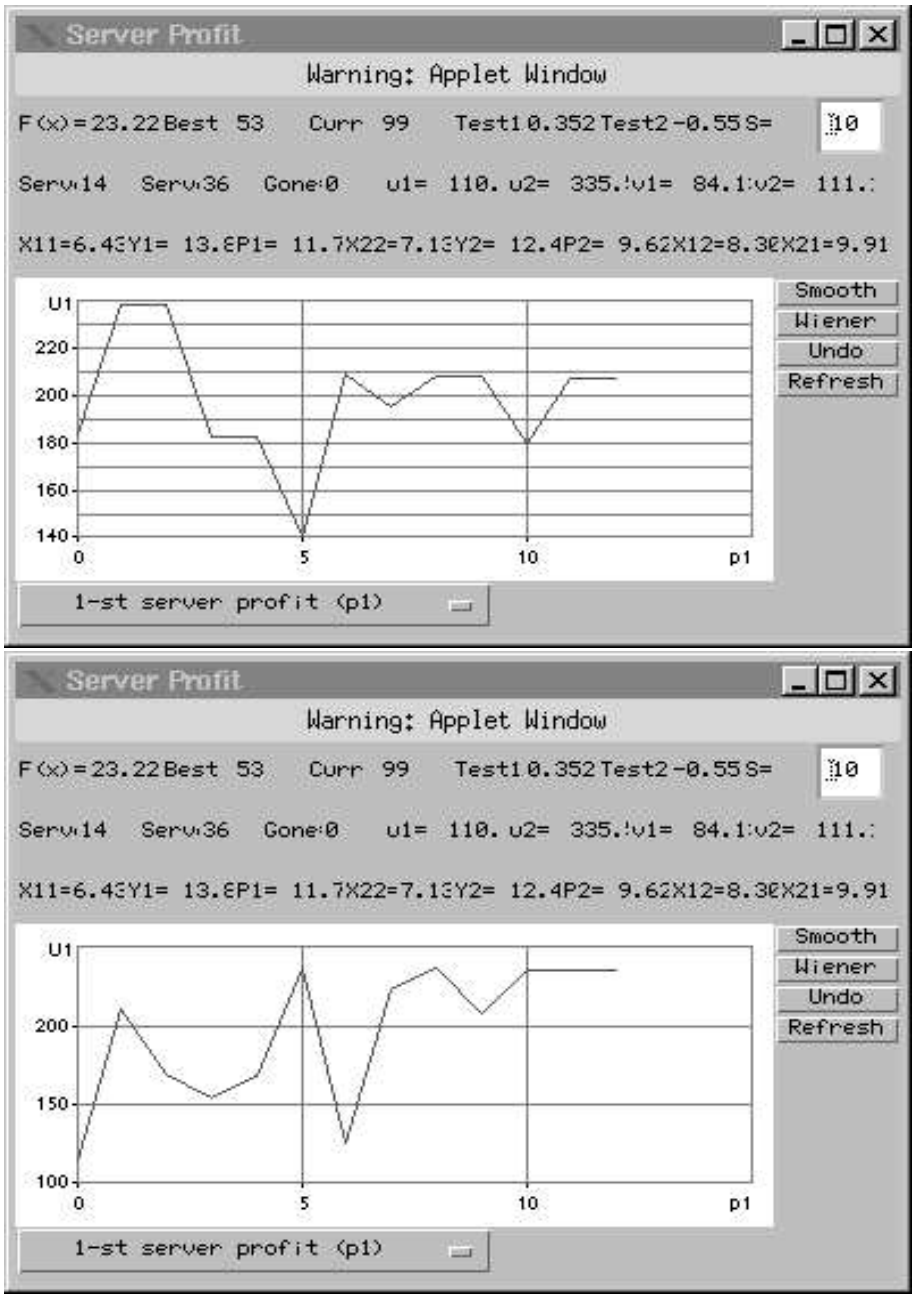


Figure 11.1. The relation of the profit u_1 on the resource price p_1 (top), the "refreshed" relation of the profit u_1 on the resource price p_1 (bottom).

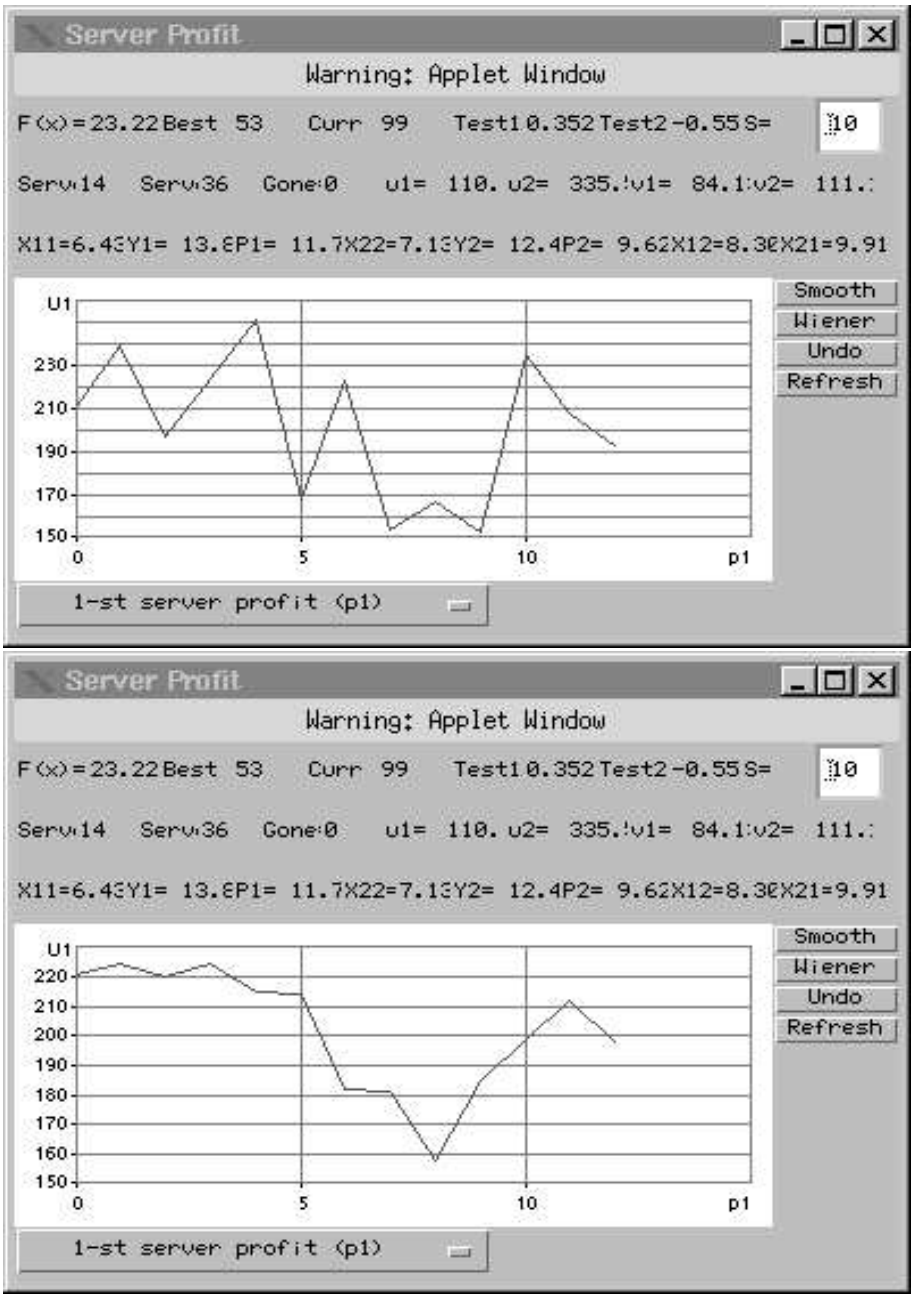


Figure 11.2. The third sample of the relation of the profit u_1 on the resource price p_1 (top), the relation of the profit u_1 on the price p_1, smoothed by the convolution filter (bottom).

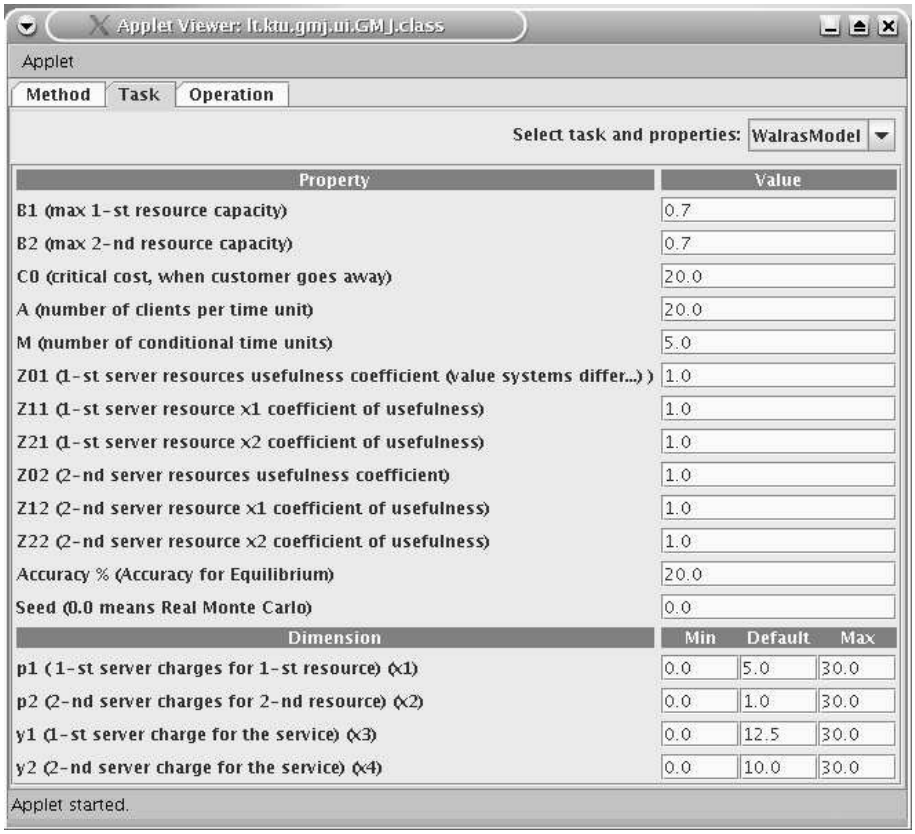


Figure 11.3. The table of initial parameters.

Figure 11.4 shows the results of optimization.
In this figure:
$iteration$ denotes the "best' iteration
$F(x)$ means the minimal deviation from the equilibrium point,
y_1, p_1, y_2, p_2 are the optimal values of the parameters.

Figure 11.5 shows how the profits of the first server depends on the service charge y_1.
The figure 11.6 shows how the profits of the first server depends on the resource price p_1.
The result is difficult to explain so the additional investigation is needed. That could be made easy by performing profit analysis by separate program at fixed esimates of equilibrium vectors p, y.

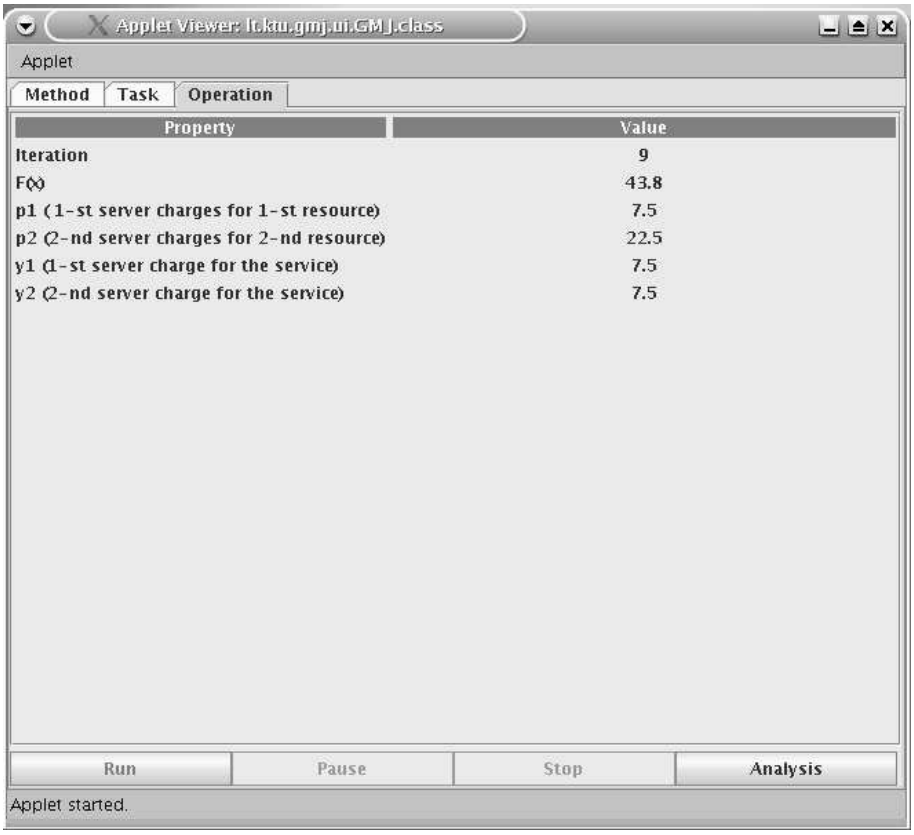


Figure 11.4. Optimization results using the method Bayes1.

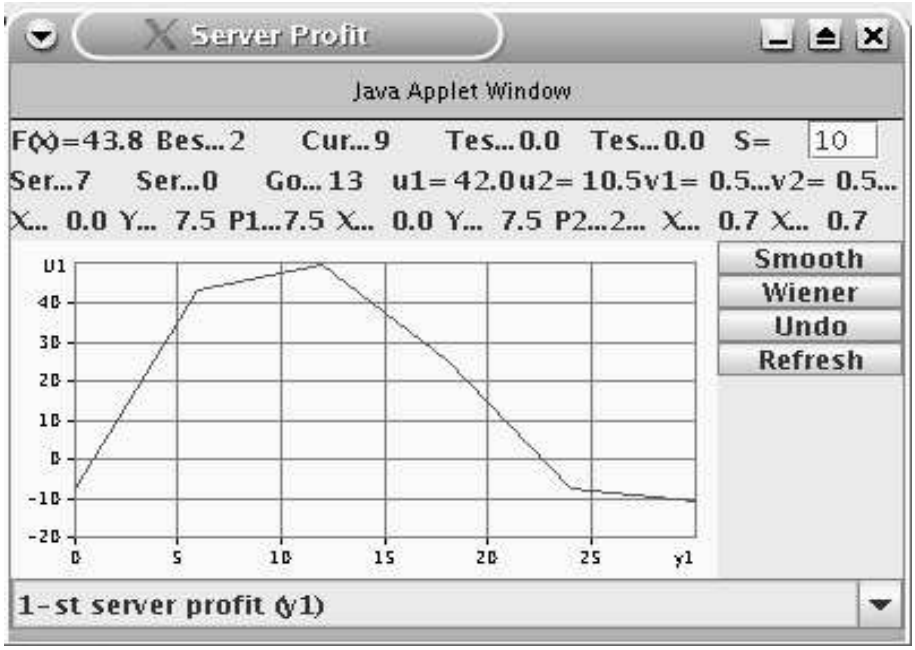
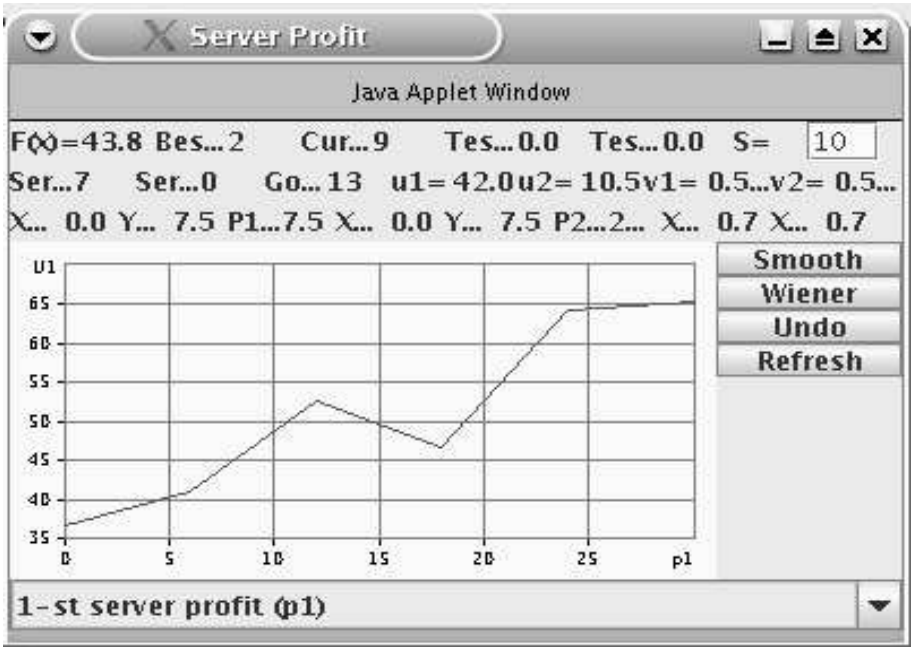


Figure 11.5. Relation of the profit u_1 on the service charge y_1.



p

Figure 11.6. Relation of the profit u_1 on the resource price p_1.

Chapter 12

"DUEL" PROBLEM, DIFFERENTIAL GAME MODEL

1. Introduction

In all three competition models, the Nash, the Walras, and the Inspector, the steady-state conditions were considered, for simplicity. In real life competition processes, dynamics are essential. Dynamical competition models may be represented as differential games. They are difficult for numerical and visual analysis, as usual (Friedman, 1971, Galperin and Zheng, 1991, Hajek, 1975, Isaacs, 1965, Krasovski, 1970, von Neumann and Morgenstern, 1953, Petrosyan, 1985, Petrosyan, 1993, Rodin, 1987). Therefore, one tries to make the dynamic competition models as simple as possible.

Consider, as a simple illustration, two objects in space trying to destroy each other. Assume, as a first approximation, that the control parameters are: initial points z_0, w_0, "climbing" rates a, b and firing times t_1, t_2. These parameters are set before the start for both objects. Denote the control parameters by vectors $x = (x_i,\ i = 1, 2, 3)$ and $y = (y_i,\ i = 1, 2, 3)$, correspondingly. Denote by (a_i^x, a_i^y) the lower bounds of these parameters. Denote by (b_i^x, b_i^y) the upper bounds. In the illustrative example $x_1 = z_0, x_2 = a, x_3 = t_1$, and $y_1 = w_0, y_2 = b, y_3 = t_2$, where $-1 \leq z_0 \leq 1, -1 \leq w_0 \leq 1, -1 \leq a \leq 1, -1 \leq b \leq 1$. Movements of the objects are described in two-dimensional "hight-time" space by the first-order differential equations

$$dz(t)/dt = az(t), \tag{12.1}$$

$$dw(\tau)/d\tau = bw(\tau),\ \tau = 2 - t. \tag{12.2}$$

The solution of these equations defines trajectories of the objects

$$z(t) = z_0 e^{at}, \tag{12.3}$$
$$w(\tau) = w_0 e^{bt}. \tag{12.4}$$

As usual, models of moving objects use homogenuos second-order differential equations, at least

$$d^2z(t)/dt^2 + gz(t)/dt = az(t), \tag{12.5}$$
$$d^2w(t)/dt^2 + hdw(\tau)/d\tau = bw(\tau),\ \tau = 2 - t. \tag{12.6}$$

The solution of these equations defines more realistic trajectories of the objects because this includes acceleration, too. These trajectories depend on the roots r_1, r_2 of the charasteristic equation

$$r^2 + gr - a = o. \tag{12.7}$$

Here

$$r_1 = -g/2 + \sqrt{(g/2)^2 + a},\ r_2 = -g/2 - \sqrt{(g/2)^2 + a}. \tag{12.8}$$

The trajectory of the first object

$$z(t) = \begin{cases} C_1 e^{r_1 t} + C_2 e^{r_2 t}, & \text{if } r_1 \neq r_2 \\ (C_1 + C_2 t)e^{-(g/2)t}, & \text{if } r_1 = r_2 \\ e^{-(g/2)t}(C_1 cos\beta t + sin\beta t), & \text{if } r_1 = -g/2 + \beta i, r_2 = -g/2 - \beta i/. \end{cases} \tag{12.9}$$

Here $\beta = \sqrt{-a - (g/2)^2}$.

The constants C_1, C_2 are defined by initial or boundary conditions. Suppose that the initial higth $z(0) = z_0$ are fixed by operator of the first flying object. Then

$$z_0 = \begin{cases} C_1 e^{r_1 t} + C_2 e^{r_2 t} = C_1 + C_2, & \text{if } r_1 \neq r_2 \\ (C_1 + C_2 t)e^{-(g/2)t} = C_1, & \text{if } r_1 = r_2 \\ e^{-(g/2)t}(C_1 cos\beta t + sin\beta t) = C_1, & \text{if } r_1 = -(g/2) + \beta i, r_2 = -(g/2) - \beta i \end{cases} \tag{12.10}$$

Considering the speed $dz(t)/dt$ one can write

$$dz(t)/dt = C_1 r_1 e^{r_1 t} + C_2 r_2 e^{r_2 t}, if\ r_1 \neq r_2 \tag{12.11}$$

$$dz(t)/dt = -(g/2)(C_1 + C_2 t)e^{-(g/2)t} + C_2 e^{-(g/2)t}, if\ r_1 = r_2 \tag{12.12}$$

(12.13)

$$dz(t)/dt =$$
$$-(g/2)e^{-g/2t}(C_1 cos\beta t + C_2 sin\beta t) +$$
$$e^{-g/2t}(-\beta C_1 sin\beta t + \beta C_2 cos\beta t)$$
$$if r_1 = -(g/2) + \beta i, r_2 = -(g/2) - \beta i \quad (12.14)$$

From here assuming that initial speed $dz(0)/dt = z_0^1$

$$z_0^1 = \begin{cases} C_1 r_1 + C_2 r_2, & \text{if } r_1 \neq r_2 \\ -(g/2)C_1 + C_2, & \text{if } r_1 = r_2 \\ -(g/2)C_1 + \beta C_2, & \text{if } r_1 = \alpha + \beta i, r_2 = \alpha - \beta i \end{cases} \quad (12.15)$$

This way we obtained three pairs of linear equations corresponding to three different pairs of roots $r_1, 2$ defining two unknown constants C_1, C_2.
The trajectory of the second object $w(t)$ is defined in the same way.

Later we shall apply first-order linear differential equations (12.1) and (12.2) as a first approximation to test algorithms of optimization and modeling. Denote by $d(t)$ a distance between objects at the moment t

$$d(t) = ||(w(t), \tau) - (z(t), t)||. \quad (12.16)$$

Denote by $p(t)$ the hitting probability

$$p(t) = 1 - (d(t)/D)^{\alpha}. \quad (12.17)$$

Here $D \geq \max_t d(t)$ and $\alpha > 0$. The expected utility function of the first object

$$U(x,y) = \begin{cases} u_1 p(x_3) - u_2(1 - p(x_3))p(y_3), & \text{if } x_3 < y_3, \\ -u_2 p(y_3) + u_1(1 - p(y_3))p(x_3), & \text{if } x_3 > y_3, \\ u_1 p(x_3) - u_2 p(y_3), & \text{if } x_3 = y_3, \end{cases} \quad (12.18)$$

The expected utility function of the second object

$$V(x,y) = \begin{cases} v_1 p(y_3) - v_2(1 - p(y_3))p(x_3), & \text{if } x_3 > y_3, \\ -v_2 p(x_3) + v_1(1 - p(x_3))p(y_3), & \text{if } x_3 < y_3, \\ v_1 p(y_3) - v_2 p(x_3), & \text{if } x_3 = y_3. \end{cases} \quad (12.19)$$

Expressions (12.18) and (12.19) of expected utilities follows from expression (12.17) of hitting probabilities. It is assumed that utilities to hit the hostile object are plus u_1 and v_1. Utilities to be hit are minus u_2 and v_2. Here u_1, u_2 denote utilities of the first object and v_1, v_2 define utilities of the second one.

2. Convex Version

Expected utilities (12.18) and (12.19) are not convex functions of variables $p(x_3)$ and $p(y_3)$. The convex version may be obtained assuming that the object

destroys the enemy, if it is not hit itself. Suppose that $u_1 = u_2 = v_1 = v_2 = 1$. Then it follows from expressions (12.18) and (12.19) that the expected utility function of the first object

$$U(x,y) = \begin{cases} p(x_3) - (1 - p(x_3)), & \text{if } x_3 < y_3, \\ -p(y_3) + (1 - p(y_3)), & \text{if } x_3 > y_3, \\ p(x_3) - p(y_3), & \text{if } x_3 = y_3, \end{cases} \tag{12.20}$$

The expected utility function of the second object

$$V(x,y) = \begin{cases} p(y_3) - (1 - p(y_3)), & \text{if } x_3 > y_3, \\ -p(x_3) + (1 - p(x_3)), & \text{if } x_3 < y_3, \\ p(y_3) - p(x_3), & \text{if } x_3 = y_3. \end{cases} \tag{12.21}$$

Expected utilities (12.18) and (12.19) are convex functions of probabilities $p(x_3)$ and $p(y_3)$ at the equilibrium point

$$\begin{aligned} p(x_3) &= 0.5, \\ p(y_3) &= 0.5. \end{aligned} \tag{12.22}$$

The convexity of expected utilities (12.18) and (12.19) as functions of probabilities $p(x_3)$ and $p(y_3)$ does not provide their convexity as functions of firing times x_3 and y_3 and other parameters. The convexity is a part of sufficient equilibrium conditions. Therefore, the equilibrium may not exist.

Expected utilities reach the maximal values at the ends of intervals of parameters z_0, a, w_0, b, as usual. For the first object there are four pure strategies corresponding to various arrangements of these ends
$(z_0 = z_{min}, a = a_{min}), (z_0 = z_{min}, a = a_{max}), (z_0 = z_{max}, a = a_{min})$,
$(z_0 = z_{max}, a = a_{max})$.
The pure strategies for the second object:
$(w_0 = w_{min}, b = b_{min}), (w_0 = w_{min}, b = b_{max}), (w_0 = w_{max}, b = b_{min})$,
$(w_0 = w_{max}, b = b_{max})$.
We may search for the pure equilibrium of bimatrix games (12.20)(12.21) corresponding to each fixed pair $(x_3.y_3)$ of firing times using simple discrete algorithm (13.14). However, the probability to obtain the pure equilibrium is high only for large bimatrix games (Forgo et al., 1999). Therefore, we consider mixed strategies, first.

3. Mixed Strategies

If expected utilities reach the maximal values at the ends of intervals of parameters z_0, a, w_0, b then mixed strategies could be helpful. The mixed strategies mean that each object chooses at random a combination of lower and upper

bounds. Define those bounds as zero-ones for simplicity. There are four events: $(z_0 = 0, a = 0), (z_0 = 0, a = 1), (z_0 = 1, a = 0), (z_0 = 1, a = 1)$. Probabilities of these events are defined by first four components $x(1), ..., x(4),\ x(i) \geq 0,\ \sum_{i=1}^{4} x(i) = 1$ of a vector $x = (x(i),\ i = 1, ..., 5)$[1].
The fifth component $x(5) = t_1$ defines the firing time. A vector $y = (y(j),\ j = 1, ..., 5)$ has a similar meaning for the second object.

Applying mixed strategies $x = (x_i,\ i = 1, ..., 5)$ and $y = (y_j,\ j = 1, ..., 5)$, the utility functions of the first and the second objects are

$$U_a(x, y) = \sum_{r \in R} \sum_{s \in S} x(r)u(r, s)y(s), \tag{12.23}$$

$$V_a(x, y) = \sum_{r \in R} \sum_{s \in S} x(r)u(r, s)y(s). \tag{12.24}$$

Here $R = \{1, 2, 3, 4\},\ S = \{1, 2, 3, 4\}$, where $1, 2, 3, 4$ denotes the four events $(z_0 = 0, a = 0), (z_0 = 0, a = 1), (z_0 = 1, a = 0), (z_0 = 1, a = 1)$. The matrix

$$u(r, s) = u_{t_1,t_2}(r, s) \tag{12.25}$$

is the winning of the first object, if the pure strategy denoted by the quadruple (t_1, t_2, r, s) is used. The matrix

$$v(r, s) = v_{t_1,t_2}(r, s) \tag{12.26}$$

is the winning of the second object, if the pure strategy (t_1, t_2, r, s) is used. Expanding expressions (12.23)

$$\begin{aligned} U_a(x, y) = {} & x(1)u(1, 1)y(1) + x(2)u(2, 1)y(1) + \\ & x(3)u(3, 1)y(1) + x(4)u(4, 1)y(1) + \\ & x(1)u(1, 2)y(2) + x(2)u(2, 2)y(2) + \\ & x(3)u(3, 2)y(2) + x(4)u(4, 2)y(2) + \\ & x(1)u(1, 3)y(3) + x(2)u(2, 3)y(3) + \\ & x(3)u(3, 3)y(3) + x(4)u(4, 3)y(3) + \\ & x(1)u(1, 4)y(4) + x(2)u(2, 4)y(4) + \\ & x(3)u(3, 4)y(4) + x(4)u(4, 4)y(4). \end{aligned} \tag{12.27}$$

[1]The variables x, y obtain different meanings, if one uses mixed strategies. That reduces the number of symbols. To avoid confusion, the utilities $U_a(x, y), V_a(x, y)$ are supplied with the subscript a. If the subscript a is absent then x, y defines the parameters z_0, w_0, a, b, t_1, t_2. Otherwise, x, y denotes mixed strategies.

Here winnings $u(r,s)$ are defined by expression (12.25). Expression (12.24) illustrates how the function $U_a(x,y)$ is defined. The function $V_a(x,y)$ is defined similarly.

One defines bimatrix games (12.23) and (12.24) for each pair of firing times $(t_1,\ t_2)$. Therefore, for fixed pair $(t_1,\ t_2)$, the same methods of search for the equilibrium points may be used as in the inspector model (13.14). Here winnings $u(r,s)$ and $v(r,s)$ depend on firing times t_1, t_2. This is the important difference from the inspector game. One updates bimatrix games while changing firing times.

Therefore, there are only two parameters t_1 and t_2 to be found by the fraud minimization (see section 4.1). One can do that by applying methods, such as the $Bayes$, to search for the pair (t_1, t_2) that minimizes the fraud.

4. Search for Equilibrium

4.1 Bimatrix Game Algorithm

Consider the bimatrix game (12.23) and (12.24) defined by fixing a pair $(t_1,\ t_2)$ of firing times. We search for the equilibrium of this game using conditions (13.14) and (13.19)- (13.23). Here the matrix is small thus the proportion of pure strategy equilibrium points will not be considerable (Forgo et al., 1999). Therefore we start by looking for such a mixed strategy that makes all the partner strategies equally profitable for him. That makes the partner fraud irrelevant. At fixed pair $(t_1,\ t_2)$ a mixed strategy $x(r)$ of the first object is "fraud irrelevant" if

$$\sum_{r\in R} x(r)v(r,s) = V,\ s\in S, \tag{12.28}$$
$$\sum_{r\in R} x(r) = 1,\ x(r)\geq 0.$$

There are four linear inequalities, five linear equations and five variables: $V,\ x(r),\ r\in R$, because the set R has four elements.

A mixed strategy $y(s)$ of the second object is fraud irrelevant if

$$\sum_{s\ inS} u(r,s)y(s) = U,\ r\in R, \tag{12.29}$$
$$\sum_{s\in S} y(s) = 1.y(s)\geq 0.$$

These expressions include inequalities what is usual in the linear programming (LP). Applying LP, the variables U and V should be expesed as differences of two non-negative variables $U = u_1 - u_2$ and $V = v_1 - v_2$. Then from the expressions (12.28), (12.29) one obtains the LP problem:

$$\max_{x,y,u,v}(u_1 - u_2 + v_1 - v_2) \quad (12.30)$$

$$\sum_{j=1}^{m} u(i,j)y_j = u_1 - u_2,\ i = 1, ..., m, \quad (12.31)$$

$$\sum_{i=1}^{m} v(i,j)x_i = v_1 - v_2,\ j = 1, ..., m, \quad (12.32)$$

$$\sum_{j=1}^{m} y_j = 1, \quad (12.33)$$

$$\sum_{i=1}^{m} x_i = 1 \quad (12.34)$$

$$x_i \geq 0,\ y_j \geq 0,\ i, j = 1, ..., m, u_1 \geq 0,\ u_2 \geq 0, v_1 \geq 0, v_2 \geq 0.. \quad (12.35)$$

Here $x = (x_1, ..., x_m),\ y = (y_1, ..., y_m),\ u = (u_1, u_2),\ v = (v_1, v_2)$.

Similar linear fraud irrelevant equations were considered in the inspection problem (see expressions (13.42) and (13.43)). Here the fraud irrelevant equations (12.31) and (12.32) depend on two unknown parameters t_1 and t_2. Therefore, linear problems (12.31) and (12.32) are just a part of general non-linear problem. We use longer notations, such as $U = U(t_1, t_2)$ and $V = V(t_1, t_2)$, to show that U, V depend on t_1, t_2.

If there is no solution of fraud irrelevant equations, we search for the pure equilibrium strategies using condition (13.14). The Strategy Elimination Algorithm (SE) is a good alternative to obtain equilibrium. A general but difficult way to find the equilibrium is to solve the bilinear problem (13.18) defining the necessary and sufficient equilibrium conditions.

Obtaining equilibrium in mixed strategies $x(r), y(s),\ r \in R,\ s \in S$ or in pure strategies r, s, one reduces the six-dimensional optimization problem to the two-dimensional one.

$$\min_{z \in B} \parallel z - T_t(z) \parallel^2 . \quad (12.36)$$

Here $z = (t_1^0, t_2^0)$, where (t_1^0, t_2^0) are the "contract" firing times. The operator T_t is defined by the fraud condition

$$t_1^1 = arg \max_{t_1} U(t_1, t_2^0),$$

$$t_2^1 = arg \max_{t_2} V(t_1^0, t_2). \tag{12.37}$$

Here $U(t_1, t_2^0)$ is the expected winning of the first object. It is the solution of the corresponding bimatrix game at fixed firing times t_1 and t_2^0. $V(t_1^0, t_2)$ is the expected winning of the second object defined as the solution of the bimatrix game at given firing times t_1^0 and t_2. For example, winnings $U(t_1, t_2^0)$ and $V(t_1^0, t_2)$ can be defined by LP problems (12.31)(12.32), or by the Direct Search Algorithm (DSA) (see Section 2.1), or by Strategy Elimination Algorithm (SE) (see Section 2.4).

The equilibrium is achieved, if the minimum (12.36) is zero. The minimization of (12.36) is the task for global optimization algorithms, including the Bayesian ones.

4.2 Matrix Game Algorithm

The bimatrix game algorithm can be applied to both the symmetric and the non-symmetric cases. In the symmetric case the utilities satisfy the zero-sum game condition $U = -V$. Here one obtains the equilibrium by solving the following linear programming problems

$$\max_x \ U \tag{12.38}$$

$$\sum_{i=1}^{m} x_i u_{ij} \geq U, \ j = 1, ..., m,$$

$$\sum_{i=1}^{m} x_i = 1, \ x_i \geq 0$$

$$U = u_1 - u_2, \ u_1 \geq 0, \ u_2 \geq 0,$$

and

$$\min_y \ V, \tag{12.39}$$

$$-\sum_{j=1}^{m} y_j u_{ij} \leq V, \ i = 1, ..., m,$$

$$\sum_{i=1}^{m} y_i = 1, \ y_i \geq 0$$

$$V = v_1 - v_2, \ v_1 \geq 0, \ v_2 \geq 0.$$

Here mini-max condition (Owen, 1968)

$$\min_y \ V = \max_x \ U \tag{12.40}$$

holds for all firing times t_1, t_2. Therefore, a search for the equilibrium is reduced to the condition

$$\min_{t_1,t_2}(U^2 + V^2). \tag{12.41}$$

The zero-sum equilibrium is reached when the hitting probabilities are equal to 0.5 for both players and their winnings are zero (see the one-dimensional example in the next section).

5. One-dimensional Example

Applying expressions (12.1)-(12.19) in one-dimensional space the trajectories are

$$z(t) = t, \tag{12.42}$$
$$w(\tau) = 2 - \tau, \tag{12.43}$$

and the hitting probability is

$$p(t) = 1 - d(t)/D. \tag{12.44}$$

Here $D \geq \max_t d(t)$. The control parameters are the firing times $x = t_1$ and $y = t_2$. Assuming that $D = 2$ and applying expression (12.20), the expected utility function of the first object is

$$U(x,y) = \begin{cases} x - (1-x), & \text{if } x < y, \\ -y + (1-y), & \text{if } x > y, \\ x - y, & \text{if } x = y. \end{cases} \tag{12.45}$$

The expected utility function of the second object follows from expression (12.21)

$$V(x,y) = \begin{cases} y - (1-y), & \text{if } x > y, \\ -x + (1-x), & \text{if } x < y, \\ y - x, & \text{if } y = x, \end{cases} . \tag{12.46}$$

The equilibrium is reached at the firing moment $x = y = 0.5$. The one-dimensional example helps to understand and to test the results of the two-dimensional case.

6. Economic Duel. Nash Model

By the "Economic Duel" we mean a dynamic case of the Nash model. In Chapter 10, the static case was described.

There are m servers providing the same service

$$u_i = u_i(x_1, y_1, ..., x_m, y_m) = a_i y_i - x_i,\ i = 1, ..., m,$$

where u_i is the profit, y_i is the service price, a_i is the rate of customers, x_i is the running cost, and i is the server index

Suppose that arriving customers estimate the average waiting time as the relation of the number of waiting customers n_i to the server capacity x_i that is assumed to be equal to the running cost

$$\gamma_i = n_i/x_i$$

A customer goes to the server i, if

$$c_i \leq c_j,\ j = 1, ..., m,\ j \neq i,\ c_i \leq c_0.$$

A customer goes away, if

$$\min_i c_i > c_0,$$

where c_0 is the critical cost. The rate a of incoming consumers is fixed

$$a = \sum_{i=0}^{m} a_i,$$

To illustrate general ideas of the dynamic Nash model, we simplify the static one, first.

6.1 Simplified Nash Model. Static Case

We express the customer estimate of the expected waiting time as

$$\gamma_i = a_i/x_i,\ i = 1, 2. \tag{12.47}$$

Assume the customer balance condition.

$$a_i/x_i + y_i = q. \tag{12.48}$$

Here q is the customer balance factor. The balance condition follows from (10.4),(10.5), (10.6), if the number of customers is large. The condition is based on the observation that arriving customers prefer servers with lesser sum $y_i + \gamma_i$. From expression (10.7)

$$q \leq c. \tag{12.49}$$

From (12.48)

$$a_i = (q - y_i)x_i, \tag{12.50}$$

and

$$u_i = (q - y_i)x_i y_i - x_i. \tag{12.51}$$

Consider a monopolistic situation when only the ith server is there. In this case

$$a_i = a, \; if \;\; q \leq c \tag{12.52}$$

and

$$y_i + a/x_i \leq c. \tag{12.53}$$

Othervise, customers abandon services $a_i = 0$. The maximal monopolistic profit

$$\max_{x_i, y_i}(ay_i - x_i), \tag{12.54}$$

$$y_i + a/x_i \leq c. \tag{12.55}$$

Replacing inequality (12.55) by equality, we eliminate (12.55) and maximize the expression

$$max_{y_i}(ay_i - a/(c - y_i)). \tag{12.56}$$

Now one defines the optimal service price

$$y_i^* = c - 1, \tag{12.57}$$

the optimal service capacity,

$$x_i^* = a, \tag{12.58}$$

and the maximal monopolistic profit

$$u_i^* = a(c - 2). \tag{12.59}$$

6.2 Dynamic Nash Model

In the previous section, considering the static model, it was tacitly assumed that a server can operate steadily while making loss instead of profit. Now we consider a case of insolvency, too. Suppose that a server gets broken, if the accumulated losses exceed some credit threshold. Here the bankrupt server is eliminated and the remaining one assumes a monopolistic position.

Denote by $U_i(t)$ a profit accumulated by the ith server at a time t. Assume that

$$U_i(t) = \int_{t_0}^{t} u_i(t)dt, \tag{12.60}$$

where $u_i(t)$ defines a profit at a moment t

$$u_i(t) = a_i(t)y_i(t) - x_i(t). \tag{12.61}$$

Here $a_i(t)$ is a rate of customers of the ith server at a moment t. This rate is formally defined as a limit of the fraction

$$a_i(t) = \lim_{\delta \to 0} \frac{A_i(t+\delta) - A_i(t)}{\delta}, \tag{12.62}$$

where $A_i(t)$ is a total number of customers arrived during an interval $(0, t),\ t \leq T$. The zero denotes the starting time of the system. T is the end of the operation period, the "horizon". In real life $A_i(t)$ is defined at fixed moments, for example, minutes, hours, days e.t.c.. We consider it as a continuous parameter because we consider a continuous model.

Denote the bankruptcy threshold of an ith server by U_i^*. Denote a bankruptcy moment (if it happens) of the ith server by t_i^*. Then

$$a_i(t) \quad = \quad \begin{cases} a, & \text{if } t \geq t_j^*,\ t \leq t_i^*, i, j = 1, 2,\ i \neq j, \\ 0, & \text{if } t \leq t_j^*,\ t \geq t_i^*\ \ i, j = 1, 2,\ i \neq j. \end{cases} \tag{12.63}$$

Otherwise, customer rates $a_i(t)$ are defined by the customer balance condition (12.50)[2] .

From (12.59) the profit

$$u_i(t) \quad = \quad \begin{cases} u_i^*, & \text{if } t \geq t_j^*,\ t \leq t_i^*, i, j = 1, 2,\ i \neq j, \\ 0, & \text{if } t \leq t_j^*,\ t \geq t_i^*\ \ i, j = 1, 2,\ i \neq j. \end{cases} \tag{12.64}$$

Otherwise, the profit $u_i(t)$ is defined by (12.61).

Assume that both servers are planning their dynamic competition in advance. Each server i defines trajectories of the service price $y_i(t)$ and the service capacity $x_i(t)$. These trajectories show how the service prices and capacities depend on the time $t \in (0, T)$. Objectives are to maximize profits

$$U_i = U_i(T),\ i = 1, 2. \tag{12.65}$$

[2]This means that the customer balance follows the changes of parameters $y_i(t)$ and $x_i(t)$ almost without delay.

Here $U_i(T)$ defines the accumulated profit (12.60) of ith server at the end of period $t = T$.

We illustrate the idea by a simple example. Assume that servers control initial values $y_i(0),\ x_i(0)$ and rates of change $b_{y_i},\ b_{x_i}$ of parameters $y_i(t)\ x_i(t)$. Then the trajectories are defined by differential equations

$$dy_i(t)/dt = b_{y_i} y_i(t), \tag{12.66}$$

and

$$dx_i(t)/dt = b_{x_i} x_i(t). \tag{12.67}$$

The corresponding solutions are

$$y_i(t) = y^i(0) \exp(b_{y_i} y_i(t)), \tag{12.68}$$

and

$$x_i(t) = x^i(0) \exp(b_{x_i} x_i(t)). \tag{12.69}$$

Search for Nash Equilibrium

The search is similar to that in the Nash model (see Section 10). We fix the the initial values, the "Contract-Vector"
$z^0 = (x_i(0)^0, b^0_{x_i}, y_i(0)^0, b^0_{y_i},\ i = 1, ..., m)$. The transformed values, the "Fraud-Vector"
$z^1 = (x_i(0)^1, b^1_{x_i}, y_i(0)^1, b^1_{y_i},\ i = 1, 2)$, is obtained by maximizing the profits of each server i. The maximization is performed assuming that all partners $j \neq i$ honors the contract
$(x_i(0)^0, b^0_{x_j}, y_i(0)^0, b^0_{y_j},\ j = 1, 2\ j \neq i)$

$$(x_i(0)^1, b^1_{x_i}, y_i(0), b^1_{y_i}) =$$
$$arg \max_{x_i(0), b_{x_i}, y_i(0), b_{y_i}} U_i(x_i(0), b_{x_i}, y_i(0), b_{y_i}, x_j(0)^0, b^0_{x_j}, y_i(0)^0, b^0_{y_j},$$
$$j = 1, 2\ j \neq i),\ i = 1, 2. \tag{12.70}$$

Here the symbol U_i denotes the accumulated profit (12.65). Formally, condition (12.70) transforms the vector
$z^n = ((x_i(0)^n, b^n_{x_i}, y_i(0)^n, b^n_{y_i},\ i = 1, 2) \in B \subset R^8,\ n = 0, 1, 2, ...$ into the vector z^{n+1}. To make expressions shorter denote this transformation by T

$$z^{n+1} = T(z^n),\ n = 0, 1, 2, ... \tag{12.71}$$

One may obtain the equilibrium at the fixed point z^n, where

$$z^n = T(z^n). \tag{12.72}$$

The fixed point z^n exists, if the feasible set B is convex and all the profit functions (10.1) are convex (Michael, 1976). We obtain the equilibrium directly by iterations (12.71) , if the transformation T is contracting. If not, then we minimize the square deviation

$$\min_{z \in B} \parallel z - T(z) \parallel^2 . \tag{12.73}$$

The equilibrium is achieved, if the minimum (12.73) is zero.

One makes the model more flexible by introducing second derivatives of parameters $y_i(t)$ and $x_i(t)$. The second derivatives represent accelerations. Assume that servers control the starting values $y_i(0),\ x_i(0)$, the rates $b_{y_i},\ b_{x_i}$ and accelerations $g_{y_i},\ g_{x_i}$ of parameters $y_i(t)\ x_i(t)$. Then trajectories are defined by differential equations

$$d^2 y_i(t)/dt^2 = g_{y_i} dy_i(t)/dt + b_{y_i} y_i(t), \tag{12.74}$$

and

$$d^2 x_i(t)/dt^2 = g_{x_i} dx_i(t)/dt + b_{x_i} x_i(t). \tag{12.75}$$

Here the vector z^n has twelve components.
$z^n = (x_i(0)^n, b^n_{x_i}, g^n_{x_i}, y_i(0)^n, b^n_{y_i}, g^n_{y_i}\ i = 1, 2) \in B \subset R^{12},\ n = 0, 1, 2, ...$
That means high dimension of the fraud minimization problem (12.73). In addition, one needs to test the convexity of accumulated profits U_i defined by expressions (12.65) as functions of parameters
$x_i(0), b_{x_i}, y_i(0), b_{y_i},\ i = 1, 2$. If they are convex then one obtains equilibrium by condition (12.73). Otherwise, some specific algorithms should be designed. An example is the inspector problem.

7. Software Example

An interactive duel model is implemented using Java JDK1.2 (Matulevicius, 1999). To run the Java applet go to web-sites (see section 4), start $GMJ2$ and select the task $DuelStarwars$ (see Figure 9.6).
The task window, in Figure 12.1, shows the domain of firing times $0 \leq t_1 \leq 1,\ 0 \leq t_2 \leq 1$.

One selects the optimization method from the $Methods$ menu (see Figure 9.7). The method $Bayes$ is selected (see Figure 12.2). The total number of

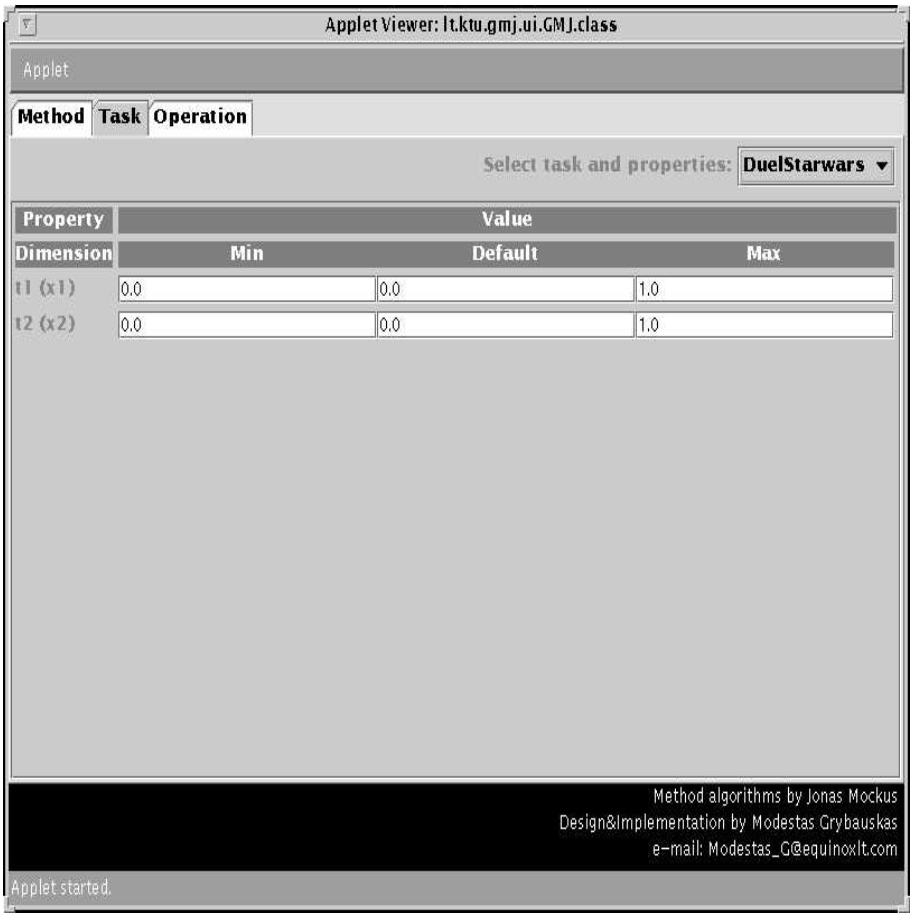


Figure 12.1. The duel task window.

iterations is set to forty. The number of initial iterations is set to five.

The optimization results are in Figure 12.3.
To start the analysis of results one touches the $Analysis$ button (see Figure 9.9).
The $Convergence$ mode shows how the deviation from the equilibrium depends on the iteration number (see Figure 12.4).

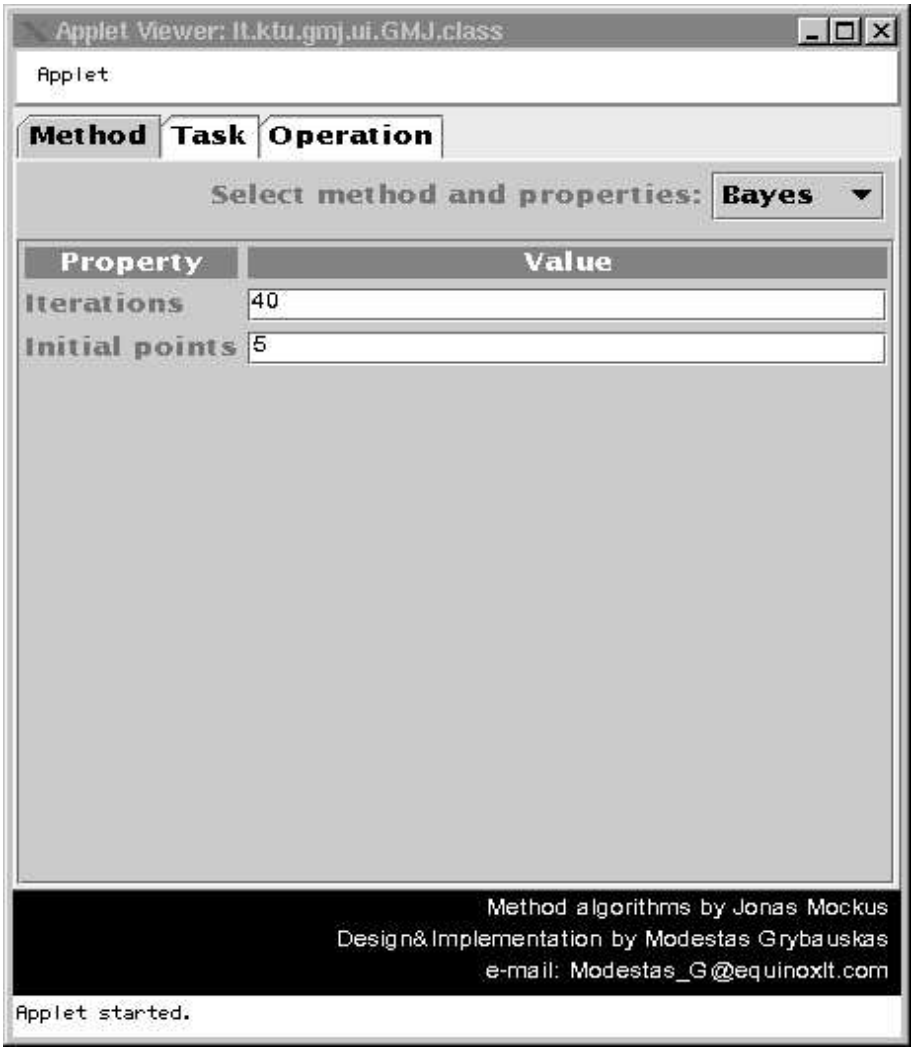


Figure 12.2. The Bayes method selection.

7.1 Operating Duel

The duel starts by selecting $DuelStarwars$ mode in the $Analysis$ menu (see Figure 9.8).
To run duel press the $Start$ button (see Figure 12.5).

One selects the operation mode in the operation window (see Figure 12.6). There are three operation modes:

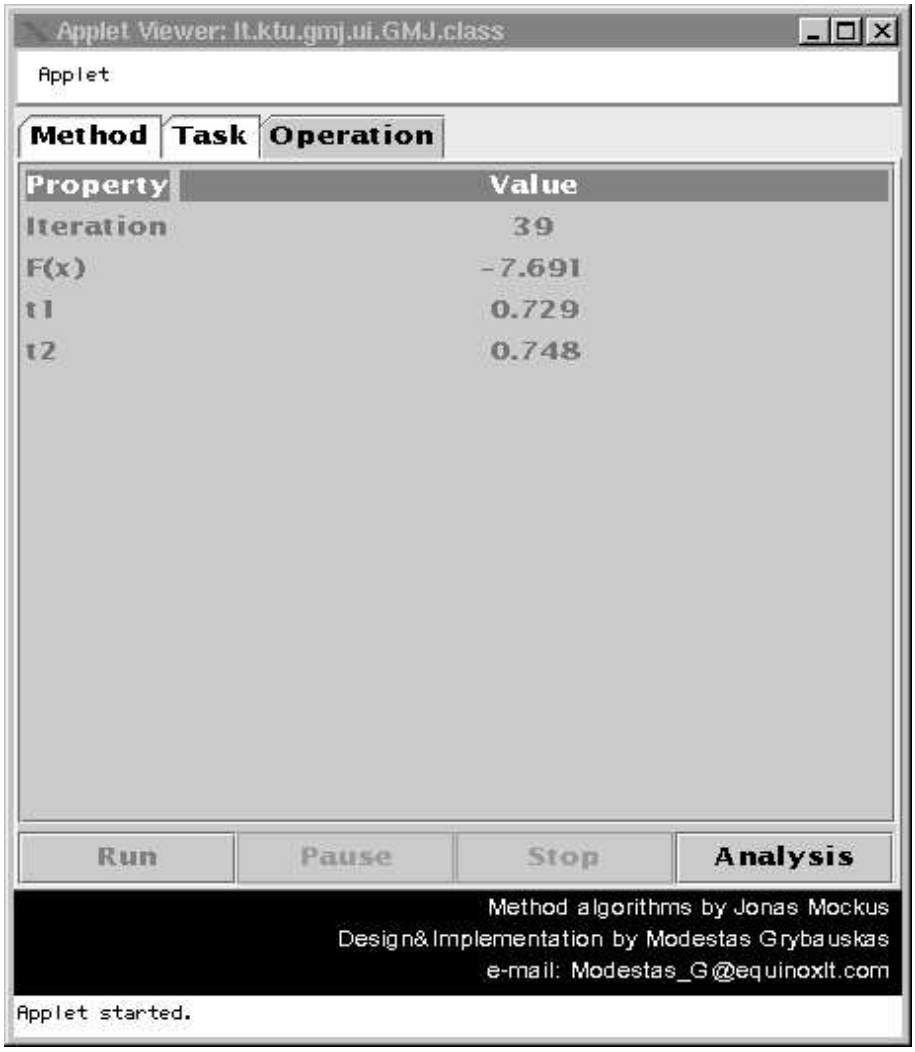


Figure 12.3. The results.

- Person vs Person,
- Person vs Computer,
- Computer vs Computer.

Figure 12.6 shows the $Computer\ vs\ Computer$ mode. The $Current\ situation$ field shows parameters z_0, w_0, a, b generated at random

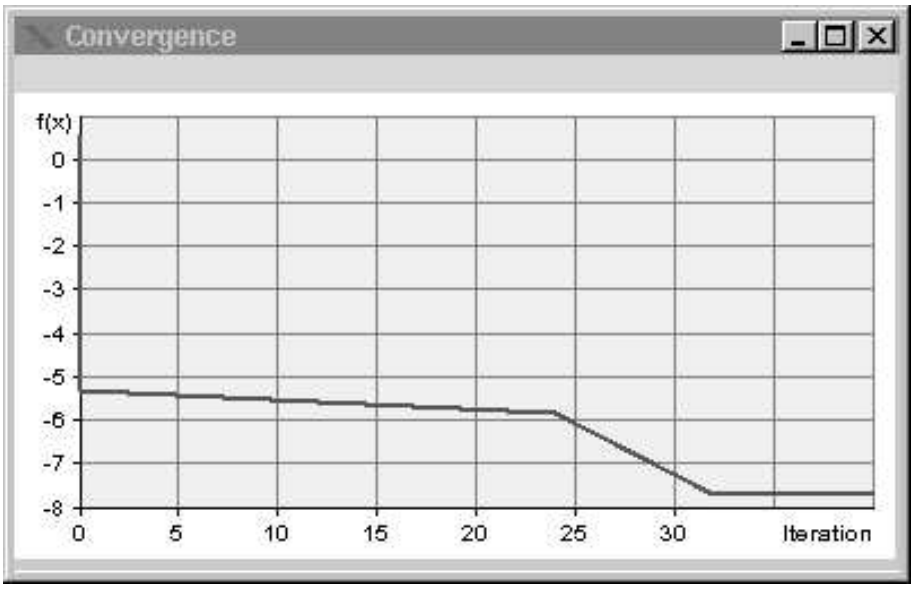


Figure 12.4. The convergence window.

using probabilities defined by equilibrium conditions. The firing times t_1, t_2 are decided by these conditions, directly.

In the $Person\ vs\ Computer$ game the person fix parameters z_0, a, t_1 of the first object. A window to fix them is in the center of the top Figure 12.7.

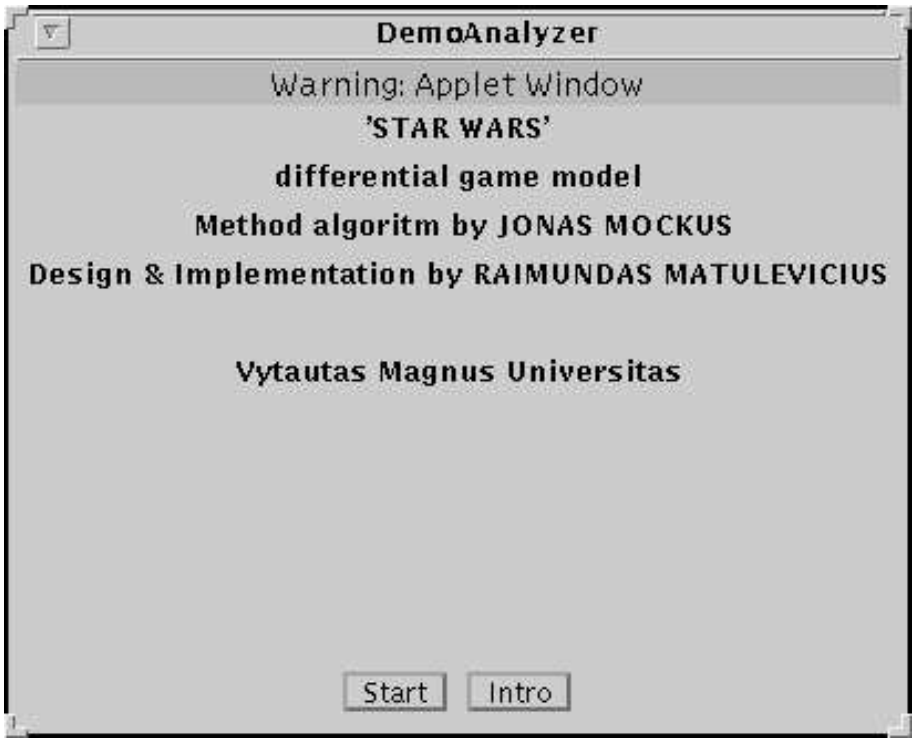


Figure 12.5. The starting window.

Probabilities of parameters w_0, b of the second object and its firing time t_2 are defined by equilibrium conditions (12.33),(12.32), and (12.36). Parameters w_0, b of the second object, and the actual hitting, are defined by probabilities. Therefore, the outcome is random.

Figure 12.8 shows the input of the $Person\ vs\ Person$ game. In this game parameters of both objects are fixed by persons using the input window. The outcome of this game is shown in a form similar to the $Person\ vs\ Computer$ game (see the bottom Figure 12.7).
One sets the speed of moving objects in the $Speed$ window (see Figureduelout1).
Figures 12.10 show $help$ windows.

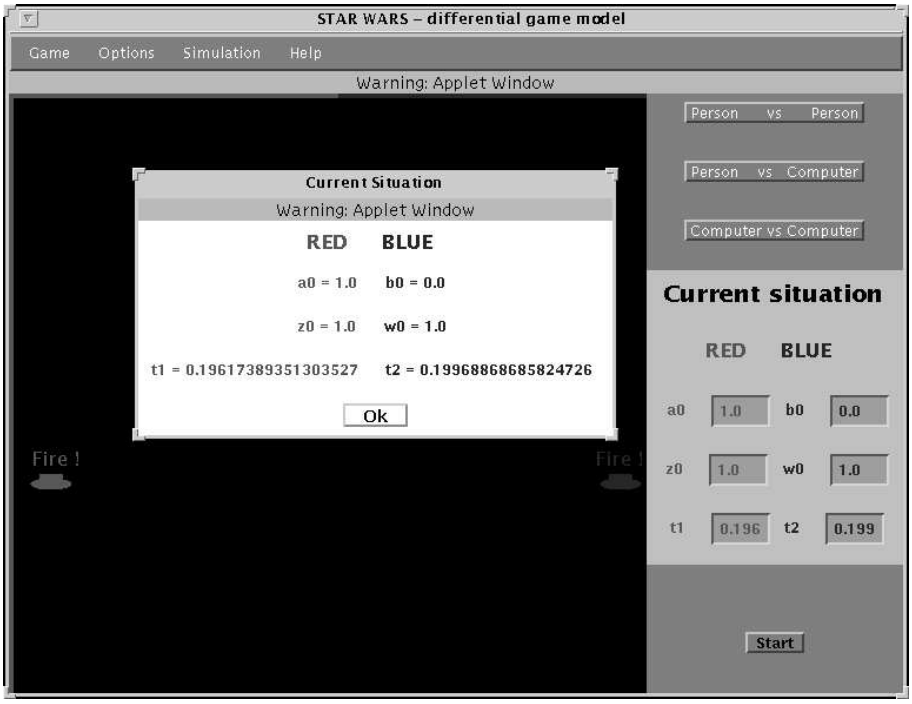


Figure 12.6. The operation window in the $Computer\ vs\ Computer$ game.

7.2 Future Developments

In the example, one searches for equilibrium in mixed strategies using the Irrelevant Fraud Algorithm (IF) which is described in section 2.3. This algorithm solves linear programming problems (12.31) and (12.32). If no solution is found, one searches for pure equilibrium strategies using the Direct Search Algorithm (DS) (13.14). The Strategy Elimination Algorithm (SE) (see section 2.4) helps, if both IF and DS fail.

The main future task is to implement flexible economic models. Another task is the implementation of second-order economic duel models. These tasks need additional theoretical investigations and software developments. Creation of entertaining graphical interfaces is important, too.

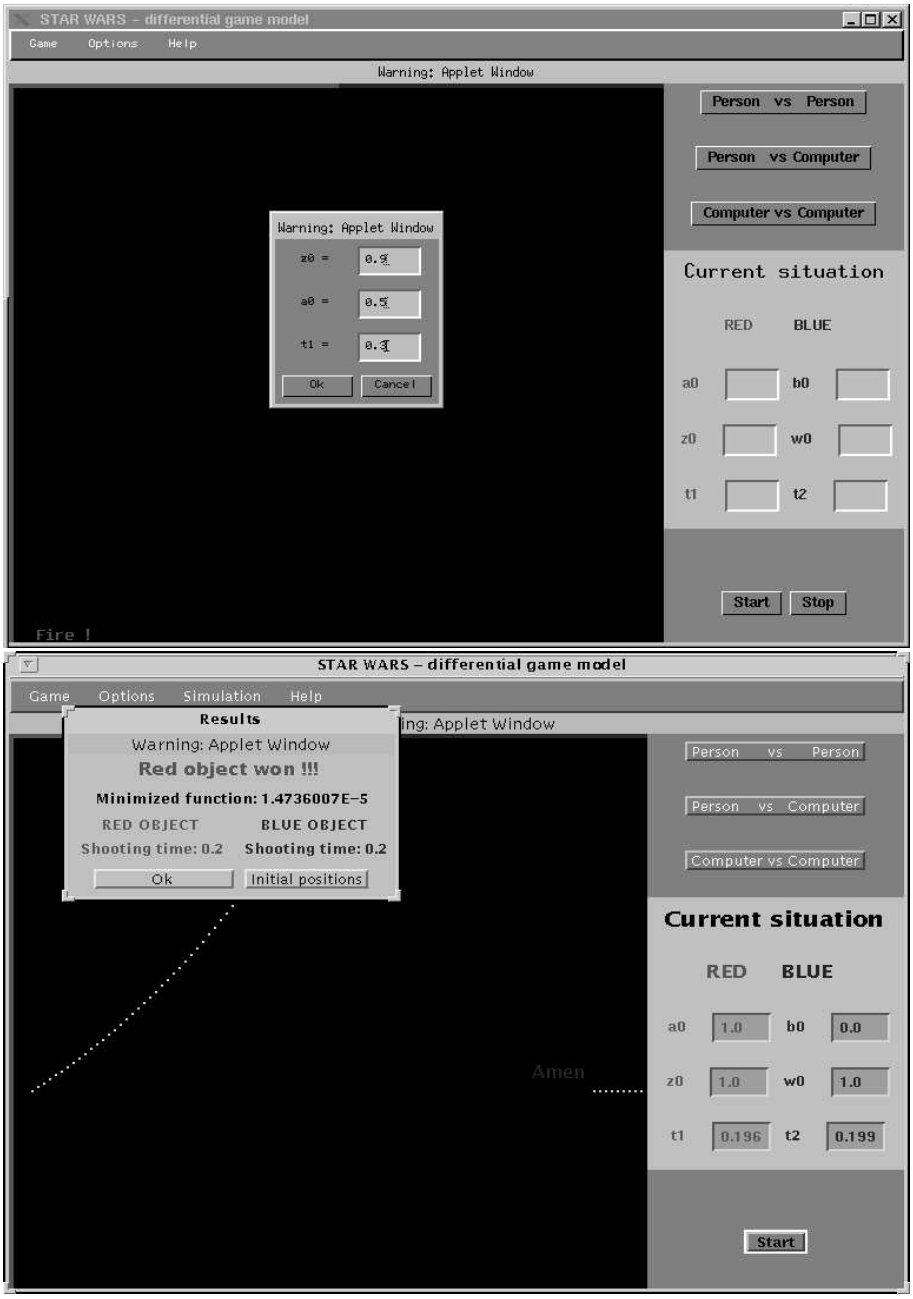


Figure 12.7. The input (top) and the outcome (bottom) of the $Person\ vs\ Computer$ game.

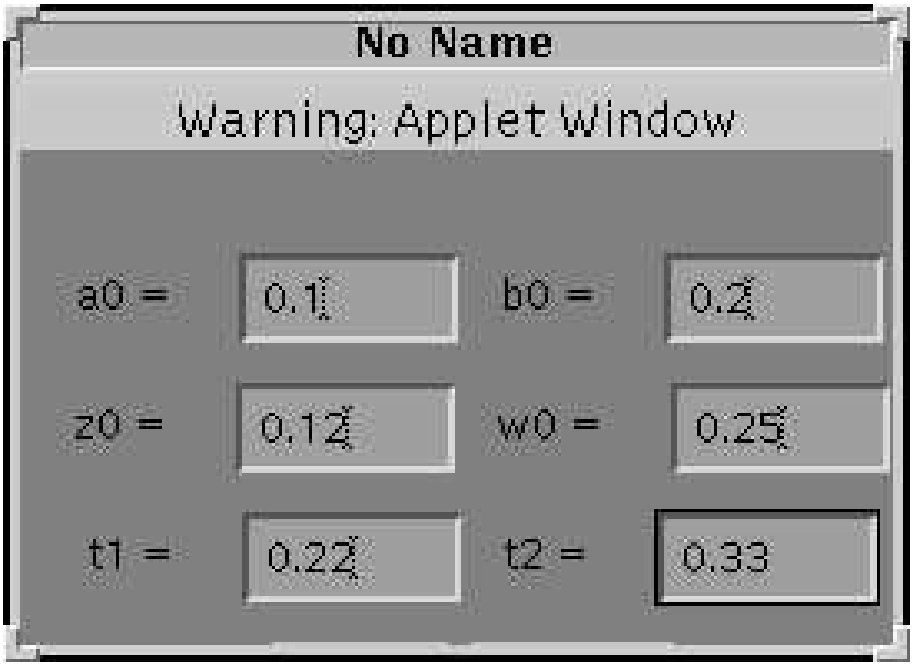


Figure 12.8. The input of the $Person\ vs\ Person$ game.

Figure 12.9. The speed control window.

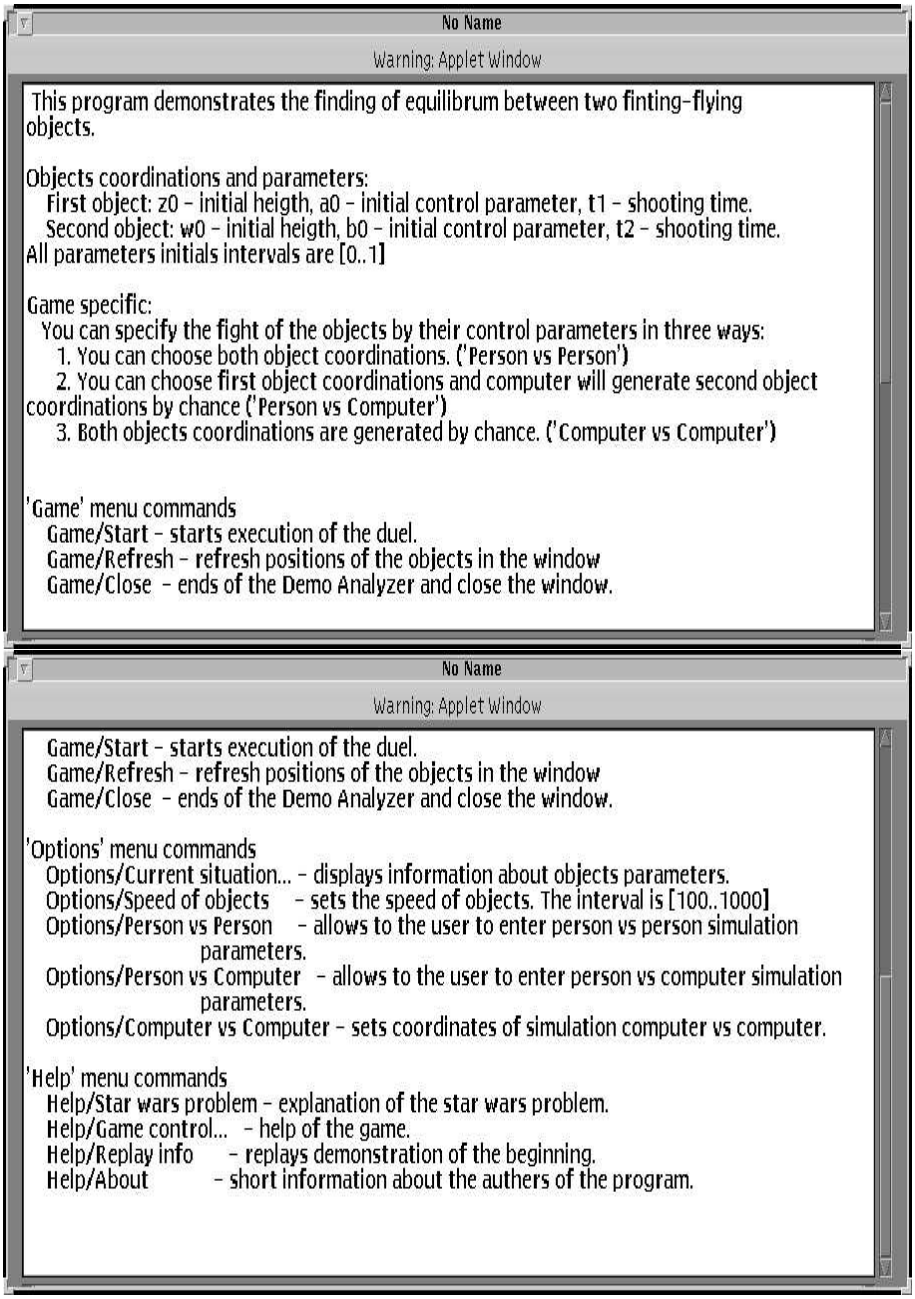


Figure 12.10. The *help* pages.

Chapter 13

INSPECTION MODEL

The Nash and Walras models illustrate the competition in economics. Both models define such prices and other production parameters that satisfy the Nash equilibrium. However, the market competition represents only a part of economical and social activities. Another part is the inspection that one needs to provide tax collection, health and environment protection, e.t.c.. We consider a simple model that illustrates the competition between inspector and violator.

1. Bimatrix Game (BG)

1.1 Quadratic Bimatrix Game (QBG)

There are two players. The payoff function of the first player is expressed by a matrix $u(i,j)$ where $i = 1, ..., m$ denote moves (pure strategies) of the first player and $j = 1, ..., n$ represent moves of the second one. The payoff of the second player is $v(i,j)$. The utility functions U and V of the first and second players are defined as expected values of the payoffs $u(i,j)$ and $v(i,j)$.

$$U(x,y) = \sum_{i,j} x_i u(i,j) y_j, \tag{13.1}$$

and

$$V(x,y) = \sum_{i,j} x_i v(i,j) y_j. \tag{13.2}$$

Here $x = (x_i,\ i = 1, ..., m)$ and $y = (y_j,\ j = 1, ..., n)$ are probabilities (mixed strategies) of moves i and j.

In the Bimatrix Game (BG) $u(i,j) \neq v(i,j)$. In the Quadratic Bimatrix Game (QBG) $m = n$. We shall consider mainly QBM because they are simpler and represent many real games.

The QBG is Well-defined Quadratic Bimatrix Game (WBG) if it cannot be reduced to a "lesser" game by elimination of a raw or a column. Details are explained by examples.

This game is not well-defined

$$u(i,j) = \begin{vmatrix} 0.1 & 0.0 & 0.0 \\ 0.0 & 0.2 & 0.0 \\ 0.0 & 0 & 0.03 \end{vmatrix} \quad v(i,j) = \begin{vmatrix} 0.8 & 1.0 & 0.1 \\ 1.0 & 0.6 & 0.1 \\ 1.0 & 1.0 & 0.04 \end{vmatrix} \tag{13.3}$$

because it can be reduced to a lesser game by elimination of the third column. Here is the reduced BG:

$$u_1(i,j) = \begin{vmatrix} 0.1 & 0.0 \\ 0.0 & 0.2 \\ 0.0 & 0 \end{vmatrix} \quad v_1(i,j) = \begin{vmatrix} 0.8 & 1.0 \\ 1.0 & 0.6 \\ 1.0 & 1.0 \end{vmatrix}$$

The reduced game is equivalent to the original one because the third column of the matrix $v(i,j)$ is dominated by the first two columns and will not be used by the second player.

However this reduced game can be reduced further by eliminating the third raw, too:

$$u_2(i,j) = \begin{vmatrix} 0.1 & 0.0 \\ 0.0 & 0.2 \end{vmatrix} \quad v_2(i,j)_r = \begin{vmatrix} 0.8 & 1.0 \\ 1.0 & 0.6 \end{vmatrix}$$

This QBG is equivalent to the previous BG because the third raw of the matrix $u_1(i,j)$ is dominated by the first two. That is convenient because solution of QBG is simpler. Therefore we later call such games as Almost Quadratic Bimatrix Games (ABG)

Here is another example of not well-defined game

$$u(i,j) = \begin{vmatrix} 0.1 & 0.0 & 0.0 \\ 0.0 & 0.1 & 0.0 \\ 0.0 & 0 & 0.29 \end{vmatrix} \quad v(i,j) = \begin{vmatrix} 0.09 & 0.1 & 0.01 \\ 0.1 & 0.08 & 0.01 \\ 0.1 & 0.1 & 0.007 \end{vmatrix} \tag{13.4}$$

In this example the third column of $v(i,j)$ is dominated by the first two but the third raw of $u(i,j)$ is not. Thus the original quadratic game is reduced to

non-quadratic one by eliminating the third column of $v(i,j)$.

$$u(i,j) = \begin{vmatrix} 0.1 & 0.0 & 0.0 \\ 0.0 & 0.1 & 0.0 \\ 0.0 & 0 & 0.29 \end{vmatrix} \quad v(i,j) = \begin{vmatrix} 0.09 & 0.1 \\ 0.1 & 0.08 \\ 0.1 & 0.1 \end{vmatrix}$$

We denote such games as NBG

Now we shall describe two simple game models.

1.2 Poaching Game

Consider the ranger-poacher game. Denote by $x = (x_1, ..., x_m),\ x_i \geq 0,\ \sum_i x_i = 1$ the inspection vector and by $y = (y_1, .., y_m),\ y_j \geq 0,\ \sum_j y_j = 1$ the violation vector. Here x_i denotes the inspection probability of the area i. y_j means the violation probability in the area j. Denote by $u(i,j)$ the inspection payoff function when the object i is inspected and the object j is violated. Denote by $v(i,j)$ the violation payoff function when the object i is inspected and the object j is violated. Functions $U(x,y)$ and $V(x,y)$ denote the inspection and violation utility functions using inspection and violation vectors x, y. These vectors define probabilities of inspection and violation.

For example,

$$u(i,j) = \begin{cases} p_i g_i q_i, & \text{if } i = j \\ 0, & \text{otherwise,} \end{cases} \tag{13.5}$$

and

$$v(i,j) = \begin{cases} -q_j p_i g_j + (1 - p_i) q_j g_j, & \text{if } i = j \\ q_j g_j, & \text{otherwise.} \end{cases} \tag{13.6}$$

Here p_i is the probability of detecting the violation if it happens in the area i. A number q_i is the probability of completing the violation[1] if violation occurs in the area i and g_i is the payoff of the completed violation in the area i.

Expression (13.5) means that if the violation is completed and detected[2] then the inspector premium is equal to the payoff of violation[3]. Expression (13.6) shows that if the violation is completed and detected, the violator payoff is negative. The payoff is positive, if it is completed and not detected.

[1] For example, killing prey.
[2] The prey is killed and a poacher is caught.
[3] The value of the killed prey.

The utility functions at given inspection and violation vectors x and y

$$U(x,y) = \sum_{i,j} x_i u(i,j) y_j, \tag{13.7}$$

and

$$V(x,y) = \sum_{i,j} x_i v(i,j) y_j. \tag{13.8}$$

Here $u(x,y) \neq v(x,y)$, so the inspection model is a bimatrix game (Forgo et al., 1999). The matrices (13.3) were defined by (13.5)(13.6) when $p_1 = 0.1,\ p_2 = 0.2,\ p_3 = 0.3$, $g_i = 1,\ i = 1,,.3$, and $q_1 = q_2 = 1.0,\ q_3 = 0.1$.

1.3 Security Game

Consider a security system protecting some object. Denote by $x = (x_1, ..., x_m)$, $x_i \geq 0,\ \sum_i x_i = 1$ the protection vector and by $y = (y_1, .., y_m),\ y_j \geq 0,\ \sum_j y_j = 1$ the violation vector. Here x_i denotes the part of security funds designated to protect the area i and y_j means the violation probability of the area j. Denote by $u(i,j)$ the security payoff function when the area i is protected and the area j is violated. Denote by $v(i,j)$ the violation payoff function when the area i is protected and the area j is violated. Utility functions $U(x,y)$ and $V(x,y)$ denote expected values of security and violation payoff functions using protection and violation vectors x, y. These vectors define the distribution of protection funds and violation probabilities.

Consider examples of simplified payoff functions. We start by defining security payoff function

$$u(i,j) \quad = \quad \begin{cases} p_i g_i - (1-p_i) g_i q_i, & \text{if } j = i \\ 0, & \text{if } j \neq i, \end{cases} \tag{13.9}$$

Here $p_i g_i$ is the expected reward of security team which protects the area i for preventing the violation and arresting the violators. Denote by p_i the probability of this event. A number g_i is the value of protected goods. Denote by q_i the probability of completing the violation[4] in the area i. Suppose that the expected penalty for bad protection is defined as the product $(1-p_i) g_i q_i$. The payoff function of security team protecting area i is zero, is no violation happens in this area $i \neq j$.

[4]For example, stealing goods.

The violation payoff function

$$v(i,j) \quad = \quad \begin{cases} -p_i b + (1-p_i) q_i g_i, & \text{if } j = i \\ q_j g_j, & \text{if } j \neq i. \end{cases} \tag{13.10}$$

Here $-p_i b$ is the penalty of attempted and detected violation where b is a large number. The expected profit of successful violation of the protected area i is defined as the product $(1-p_i)q_j g_j$ and the expected profit of successful violation of the unprotected area j is $q_j g_j$.

Assume for simplicity that probability of protection p_i is equal to the part of security funds x_i. This is the linearizion of the non-linear security function $p_i = p_i(x_i)$.

Expression (13.9) means that if the violation is completed and detected[5] in the area i then the reward of security is equal to the value of the protected goods g_i. Expression (13.10) shows that if the violation is detected, the violator penalty is a large number b. The violetors payoff is equal to the value g_i of stolen goods if the crime in the area i is successfully completed and not detected.

The utility functions at given inspection and violation vectors x and y

$$U(x,y) = \sum_{i,j} x_i u(i,j) y_j, \tag{13.11}$$

and

$$V(x,y) = \sum_{i,j} x_i v(i,j) y_j. \tag{13.12}$$

Here $u(x,y) \neq v(x,y)$.

The matrices (13.4) were defined by (13.9)(13.10) when
$p_1 = 0.1,\ p_2 = 0.2,\ p_3 = 0.3$, $g_i = 1,\ i = 1,,,3$, and $q_1 = q_2 = 0.1,\ q_3 = 0.01$

2. Search for Equilibrium

It is well known (Forgo et al., 1999) that, in randomly generated large bimatrix games, the proportion of games having k pure strategy equilibriums is defined by the Poisson distribution

$$e^{-1}/(k!) \tag{13.13}$$

[5]Goods are not stolen and the thief is caught.

That means that roughly two-thirds of randomly generated bimatrix games have at least one equilibrium point in pure strategies. This result holds for general randomly generated bimatrix games [6].

2.1 Direct Search Algorithm (DS)

A simple way to obtain equilibrium in pure strategies is the Direct Search Algorithm (DS). The set IJ of equilibrium points in pure strategies is the intersection of two sets I and J defined by the following conditions

$$IJ = I \cap J, \tag{13.14}$$

$$I = \cup_j I(j),\ J = \cup_i J(i) \tag{13.15}$$

$$I(j), \subseteq arg\ max_i\ u(i,j) \tag{13.16}$$

$$J(i), \subseteq arg\ max_j\ v(i,j). \tag{13.17}$$

Here $I(j)$ is a set of maximal elements at each column of the matrix $u(i,j)$. $J(i)$ is a set of maximal elements at each row of the matrix $v(i,j)$. If equilibrium exists and the intersection IJ is empty, one needs mixed strategies to obtain the equilibrium (Owen, 1968).

2.2 Necessary and Sufficient Conditions of BG

For a pair (x^0, y^0) to be an equilibrium point of the bimatrix game (A, B), it is necessary and sufficient (Forgo et al., 1999) that there exist real numbers α^0, β^0 such that $(x^0, y^0, \alpha^0, \beta^0)$ satisfies the system

$$\begin{aligned} xAy - \alpha &= 0, \\ xBy - \beta &= 0, \\ Ay - \alpha I &\le 0, \\ xB - \beta I &\le 0, \\ x \ge 0,\ y \ge 0,\ Ix = 1, Iy &= 1. \end{aligned} \tag{13.18}$$

Here I denotes the unit vector. The condition (13.18) represents the bilinear problem because it involves products of different variables. The solution of bilinear problems is difficult. Therefore, specific algorithms are developed to search for mixed strategies of the equilibrium. Complementary pivoting is most frequently used (Lemke, 1965) one. The Lemke-Howson algorithm finds

[6] In the inspector game, there is no equilibrium in pure strategies if $g_i = g$ and $q_i = q$, see (13.68).

a sample equilibrium point of bimatrix game.
Later we shall investigate a simple "Irrelevant Fraud" (IF) algorithm (Mockus, 2000) that follows directly from the Nash equilibrium conditions and is implemented using standard software of linear problems. The conditions when IF provides the Nash equilibrium will be defined. The software for QBG will be described.

2.3 Irrelevant Fraud Algorithm (IF)

It follows from definition that the mixed strategies x and y satisfy equilibrium if one cannot obtain greater profit by changing them. From conditions

$$\sum_{j=1}^{m} u(i,j)y_j = U,\ i = 1,...,m, \tag{13.19}$$

$$\sum_{i=1}^{m} v(i,j)x_i = V,\ j = 1,...,m, \tag{13.20}$$

$$\sum_{j=1}^{m} y_j = 1, \tag{13.21}$$

$$\sum_{i=1}^{m} x_i = 1, \tag{13.22}$$

$$x_i \geq 0,\ y_j \geq 0,\ i,j = 1,...,m. \tag{13.23}$$

it follows that

$$\sum_{i=1}^{m}\sum_{j=1}^{m} x_i u(i,j)y_j = U,\ i = 1,...,m, \tag{13.24}$$

$$\sum_{j=1}^{m}\sum_{i=1}^{m} y_j v(i,j)x_i = V,\ j = 1,...,m, \tag{13.25}$$

$$\sum_{j=1}^{m} y_j^0 = 1, \tag{13.26}$$

$$\sum_{i=1}^{m} x_i^0 = 1, \tag{13.27}$$

$$x_i \geq 0,\ y_j \geq 0,\ i,j = 1,...,m. \tag{13.28}$$

because

$$\sum_{i=1}^{m} x_i \sum_{j=1}^{m} u(i,j)y_j = U,\ i = 1,...,m, \tag{13.29}$$

$$\sum_{j=1}^{m} y_j \sum_{i=1}^{m} v(i,j)x_i = V,\ j = 1, ..., m, \tag{13.30}$$

$$\sum_{j=1}^{m} y_j = 1, \tag{13.31}$$

$$\sum_{i=1}^{m} x_i = 1, \tag{13.32}$$

$$x_i \geq 0,\ y_j \geq 0,\ i, j = 1, ..., m. \tag{13.33}$$

and

$$\sum_{i=1}^{m} x_i U = U,\ i = 1, ..., m, \tag{13.34}$$

$$\sum_{j=1}^{m} y_j V = V,\ j = 1, ..., m, - \tag{13.35}$$

$$\sum_{j=1}^{m} y_j = 1, \tag{13.36}$$

$$\sum_{i=1}^{m} x_i = 1, \tag{13.37}$$

$$x_i \geq 0,\ y_j \geq 0,\ i, j = 1, ..., m. \tag{13.38}$$

The idea is to define such mixed strategies that makes the partner fraud irrelevant[7]. Therefore, if the solution x, y of (13.19), (13.21, (13.23) exists one obtains the equilibrium just by setting

$$x^0 = x, \tag{13.39}$$

and

$$y^0 = y. \tag{13.40}$$

Expressions (13.19) - (13.23) include inequalities. Therefore, the linear programming (LP) is a convenient method of solution. In LP all the variables are supposed to be nonnegative. Thus, variables U and V should be expressed as differences of two nonnegative variables $U = u_1 - u_2$ and $V = v_1 - v_2$. Then from expressions (13.19) (13.21 (13.23) one obtains this LP problem

$$\max_{x,y,u,v} (u_1 - u_2 + v_1 - v_2), \tag{13.41}$$

[7]The fraud is irrelevant, if average winnings of the partner does not depend on his pure strategy $i = 1, ..., m$, and vice versa

$$\sum_{j=1}^{m} u(i,j)y_j = u_1 - u_2,\ i = 1, ..., m, \quad (13.42)$$

$$\sum_{i=1}^{m} v(i,j)x_i = v_1 - v_2,\ j = 1, ..., m, \quad (13.43)$$

$$\sum_{j=1}^{m} y_j = 1, \quad (13.44)$$

$$\sum_{i=1}^{m} x_i = 1, \quad (13.45)$$

$$x_i \geq 0,\ y_j \geq 0,\ i, j = 1, ..., m, u_1 \geq 0,\ u_2 \geq 0, v_1 \geq 0, v_2 \geq 0. \quad (13.46)$$

Here $x = (x_1, ..., x_m),\ y = (y_1, ..., y_m),\ u = (u_1, u_2),\ v = (v_1, v_2)$.

Note, that in WBG the expressions (13.19) - (13.23) and (13.24) - (13.28) satisfy the equilibrium condition (13.18). That remains true in the almost well-defined quadratic bimatrix games (ABG), too.

The IF conditions shows some interesting aspects of Nash equilibrium of quadratic bimatrix games and can be implemented using standard software for linear equations or linear programming that is simple, efficient and widely available.

Example-1 (explicit solution of poaching game)

Suppose that $q_i = g_i = 1$. Then from (13.5)(13.6)

$$u(i,j) \quad = \quad \begin{cases} p_j, & \text{if } i = j \\ 0, & \text{otherwise,} \end{cases} \quad (13.47)$$

and

$$v(i,j) \quad = \quad \begin{cases} -p_i + (1 - p_i), & \text{if } i = j \\ 1, & \text{otherwise.} \end{cases} \quad (13.48)$$

From here and the Irrelevant Fraud expressions(13.19)(13.23)

$$\sum_{j=1}^{m} u(i,j)y_j = U,\ i = 1, ..., m, \quad (13.49)$$

$$\sum_{i=1}^{m} v(i,j)x_i = V,\ j = 1, ..., m, \quad (13.50)$$

$$\sum_{j=1}^{m} y_j = 1, \tag{13.51}$$

$$\sum_{i=1}^{m} x_i = 1, \tag{13.52}$$

$$x_i \geq 0,\ y_j \geq 0,\ i,j = 1,...,m. \tag{13.53}$$

From here and (13.47)(13.48) follows

$$p_j y_j = U,\ j = 1,...,m \tag{13.54}$$

$$\sum_{i \neq j} x_i + (1 - 2p_j)x_j = V,\ j = 1,...,m \tag{13.55}$$

$$\sum_j y_j = 1,\ \sum_i x_i = 1,\ y_j \geq 0,\ x_i \geq 0$$

In the case $m = 2$ from (13.54)

$$p_1 y_1 = U, \tag{13.56}$$

$$p_2 y_2 = U. \tag{13.57}$$

From here

$$y_1 = p_2/(p_1 + p_2), \tag{13.58}$$

$$y_2 = p_1/(p_1 + p_2), \tag{13.59}$$

because $p_1 + p_2 = 1$.
From (13.55)

$$(1 - 2p_1)x_1 + x_2 = V, \tag{13.60}$$

$$x_1 + (1 - 2p_2)x_2 = V. \tag{13.61}$$

From here

$$x_1 = p_2/(p_1 + p_2), \tag{13.62}$$

$$x_2 = p_1/(p_1 + p_2). \tag{13.63}$$

If $m = 3$ then, in the same way, one obtains

$$y_1 = x_1 = p_2p_3/(p_1p_2 + p_1p_3 + p_2p_3), \tag{13.64}$$

$$y_2 = x_2 = p_1p_3/(p_1p_2 + p_1p_3 + p_2p_3), \tag{13.65}$$

$$y_3 = x_3 = p_1p_2/(p_1p_2 + p_1p_3 + p_2p_3). \tag{13.66}$$

In the general case

$$y_i = x_i = \prod_{k \neq i} p_k/(\sum_i \prod_{k \neq i} p_k), \tag{13.67}$$

where $\prod_{k \neq i} p_k$ means a product of all p_k except p_i. Here the solutions are positive

$$x_i > 0,\ y_j > 0,\ i, j = 1, ..., m. \tag{13.68}$$

That shows, that the inspector problem (13.5)(13.6) can be solved by the Irrelevant Fraud (IF) algorithm.

However, the solution of linear inequalities (13.19) - (13.23) and the corresponding LP (13.43) - (13.46) may not exist if the game is not WBG.

Then one searches for the equilibrium using the Direct Search (DS) algorithm (13.14). That complements the IF algorithm (13.44) - (13.46) and defines the equilibrium in pure strategies, if it exists.

2.4 Quadratic Strategy Elimination Algorithm (QSE)

The quadratic strategy elimination algorithm includes both IF and DS algorithms. In addition it eliminates irrelevant raws and columns:

1 obtain a solution (x^*, y^*) of linear equations (13.19) - (13.22), including inequalities (13.23),
if a solution of the system is found, go to step 7

2 search for equilibrium in pure strategies by the Direct Search algorithm (13.14)
if a solution is found, go to step 7

3 obtain a solution (x^*, y^*) of linear equations (13.19) - (13.37), ignoring inequalities (13.23)

4 reduce the system of linear equations by eliminating the column $j = k$ and the raw $i = k$ if at least one of the corresponding variables is negative $y_k \leq 0$ or/and $x_k \leq 0$, set to zero both the variables $y_k = 0$ and $x_k = 0$

5 if a solution of the reduced system is found, go to step 7

6 if the reduced system is not empty, go to step 3

7 estimate maximal wins U_{max}, V_{max} obtainable assuming that partners keep solutions (x^*, y^*)

$$U_{max} = \max_x \sum_{i,j} x_i u(i,j) y_j^*, \tag{13.69}$$

$$V_{max} = \max_{y} \sum_{i,j} x_i^* v(i,j) y_j. \tag{13.70}$$

8 compare obtained wins U^*, V^* with the maximal ones U_{max}, V_{max} ,
if $U^* \geq U_{max}$ and $V^* \geq V_{max}$ stop, and record the components $x_i^* > 0,\ y_j^* * > 0$ as equilibrium strategies,
otherwise print the note that the equilibrium tests (13.69) and (13.70) failed.

QSE and Equilibrium conditions

QSE algorithm stops if DS or IF solution is obtained. In the DS case an equilibrium is found by definition. The same is true for IF solution of WBG. That remains true for IF solutions of ABG because here original QBG are reduced to lesser QBG.

Often the equilibrium is found in NBG, too, but there is no guarantee. However QSE is fast and each solution is tested by conditions (13.69) and (13.70). Therefore one may safely use QSE for NBG as well.

Example-2 (numerical solution of poaching game)

Suppose that $g_1 = g_2 = g_3 = 1$,
$q_1 = q_2 = 1,\ q_3 = 0.1$ and $p_1 = 0.1,\ p_2 = 0.2,\ p_3 = 0.3$.
Then from (13.5)(13.6)

$$u(i,j) = \begin{vmatrix} 0.1 & 0.0 & 0.0 \\ 0.0 & 0.2 & 0.0 \\ 0.0 & 0 & 0.03 \end{vmatrix}. \quad v(i,j) = \begin{vmatrix} 0.8 & 1.0 & 0.1 \\ 1.0 & 0.6 & 0.1 \\ 1.0 & 1.0 & 0.04 \end{vmatrix}. \tag{13.71}$$

There is no IF solution. Ignoring inequalities (13.53) one obtains

$$x_1 = 3.31 x_2 = 1.66 x_3 = -3.97 \tag{13.72}$$

$$y_1 = 0.21 y_2 = 0.10 y_3 = 0.69 \tag{13.73}$$

There is no DS solution, too. Therefore, following the fourth step of QSE algorithm, the game is reduced setting to zero variables $x_3 = y_3 = 0$. Then

$$u(i,j) = \begin{vmatrix} 0.1 & 0.0 \\ 0.0 & 0.2 \end{vmatrix}. \quad v(i,j) = \begin{vmatrix} 0.8 & 1.0 \\ 1.0 & 0.6 \end{vmatrix}. \tag{13.74}$$

Here the IF solution $x_1 = 2/3,\ x_2 = 1/3,\ y_1 = 2/3,\ y_2 = 1/3$ and the expected values $U = 0.067,\ V = 0.867$ satisfy the equilibrium conditions as expected because this is ABG.

Example-3 (numerical solution of security game)

Suppose that $g_1 = g_2 = g_3 = 1$,
$q_1 = q_2 = 0.1,\ q_3 = 0.01$ and $p_1 = 0.1,\ p_2 = 0.2,\ p_3 = 0.3$.
Then from (13.5)(13.6)

$$u(i,j) = \begin{vmatrix} 0.0 & 0.0 & 0.0 \\ 0.0 & 0.1 & 0.0 \\ 0.0 & 0.0 & 0.29 \end{vmatrix}. \quad v(i,j) = \begin{vmatrix} 0.09 & 0.1 & 0.01 \\ 0.1 & 0.08 & 0.01 \\ 0.1 & 0.1 & 0.07 \end{vmatrix}. \tag{13.75}$$

There is no IF solution. Ignoring inequalities (13.53) one obtains

$$x_1 = 6.414x_2 = 3.207x_3 = -8.621 \tag{13.76}$$

$$y_1 = 1.0y_2 = 0.0y_3 = 0.0 \tag{13.77}$$

There is no DS solution, too. Therefore, following the fourth step of QSE algorithm, the game is reduced setting to zero variables $x_3 = y_3 = 0$. Then

$$u(i,j) = \begin{vmatrix} 0.0 & 0.0 \\ 0.0 & 0.1 \end{vmatrix}. \quad v(i,j) = \begin{vmatrix} 0.09 & 0.1 \\ 0.1 & 0.08 \end{vmatrix}. \tag{13.78}$$

Here the QSE solution

$$x_1 = 1/3\ x_2 = 2/3\ x_3 = 0 \tag{13.79}$$

$$y_1 = 1.0\ y_2 = 0.0\ y_3 = 0.0 \tag{13.80}$$

satisfy the equilibrium conditions. Note that this game is NBG so the solution is not guaranteed in general. Therefore the statistical investigation of QSE algorithm in the NBG games would be useful to estimate the expected failure numbers when QSE is applied in NBG.

2.5 Preventing Corruption

The game will not be played by rules if the players may win more by breaking them. Unauthorized deals (for example, bribes) are usual tools breaking the rules of non-zero-sum games where $v(i,j) \neq -u(i,j)$. An example of such deal is sharing the killed pray between the inspector and violator. Values of the optimal deal $(\bar{u}, \bar{v})$ are defined by the Nash bargaining conditions (Owen, 1968)

The optimal deal $(\bar{u}, \bar{v})$ depends on the set of feasible deals D,
and on max-min values defining what the players will get if the deal fails

$$u^* = max_x\ min_y U(x,y),$$
$$v^* = max_x\ min_y V(x,y). \tag{13.81}$$

Here
u^* is the maximal expected guarantee win of the first player,
v^* is that of the second player.
In the deal the first player gets $\bar{u}$, the second one obtains $\bar{v}$.
The optimal deal $(\bar{u}, \bar{v}) = \phi(D, u^*, v^*)$ is uniquely defined by the following assumptions.

1 $(\bar{u}, \bar{v}) \geq (u^*, v^*)$,

2 $(\bar{u}, \bar{v}) \in D$,

3 if $(u, v) \in D$ and $(u, v) \geq (\bar{u}, \bar{v})$,
then $(u, v) = (\bar{u}, \bar{v})$,

4 if $(\bar{u}, \bar{v}) \in T \subset D$ and $(\bar{u}, \bar{v}) = \phi(D, u^*, v^*)$, then $(\bar{u}, \bar{v}) = \phi(T, u^*, v^*)$,

5 suppose that a set D' is obtained from D by this transformation
$u' = a_1 u + b_1,\ v' = a_2 v + b_2$, then from $\phi(D, u^*, v^*) = (\bar{u}, \bar{v})$ follows that $\phi(D', a_1 u + b_1,\ a_2 v + b_2) = (a_1 u + b,\ a_2 v + b_2)$,

6 if $(u, v) \in D \Leftrightarrow (v, u) \in D$, $u^* = v^*$ and $\phi(D, u^*, v^*) = (\bar{u}, \bar{v})$, then $\bar{u} = \bar{v}$.

If there is a pair $(u, v) \in D$, such that $u > u^*,\ v > v^*$, then the optimal deal

$$(\bar{u}, \bar{v}) = arg\ max_{u,v}(u - u^*)(v - v^*), \tag{13.82}$$

where

$$(u, v) \in D,\ u \geq u^*,\ v \geq v^* \tag{13.83}$$

If the sum of expected wins of both players is restricted by c then

$$D = \{(u, v) : u + v \leq c\}, \tag{13.84}$$

and the optimal deal

$$(\bar{u}, \bar{v}) = \\ ((c + u^* - v^*)/2,\ (c + v^* - u^*)/2) \tag{13.85}$$

An obvious way to prevent unauthorized deals is by introducing penalties (w_1, w_2) that makes the deal unprofitable

$$(w_1, w_2) \geq (\bar{u} - u^*, \bar{v} - v^*) \tag{13.86}$$

From (13.85)

$$w_i \geq c - u^* - v^*,\ i = 1, 2. \tag{13.87}$$

Assuming, that the deal is arranged before the game

$$c = \max_i\ g_i q_i \tag{13.88}$$

Here c is the maximal expected utility to be divided between the bargaining players.

Example-4 (corrupt inspector)

Suppose that in the case of inspector game (2.4) the sum of expected wins is restricted by $c = \max_i\ g_i q_i = 1$.
Assume the max-min values to be equal to nearly-equilibrium solution $u^8 = U = 0.067, v^* = V = 0.867$.
Then, from (13.85) the optimal deal
$(\bar{u}, \bar{v}) = (0.1, 0.9)$. From (13.87) expected penalties preventing the deal
$w_i \geq 0.0467,\ i = 1, 2$

3. Matrix Game

In some special cases, the Inspector problem is reduced to the Matrix Game. It is well known (Owen, 1968) that equilibrium is defined as two linear programming problems, if $v_{ij} = -u_{ij}$. One reduces utilities (13.5) and (13.6) to the zero-sum case by subtracting the penalty terms $(1 - p_i)q_j g_j$ and $q_i g_i$ from the inspectors utilities. In such a case

$$u(i,j) \quad = \quad \begin{cases} p_i g_i q_i - (1 - p_i) q_j g_j, & \text{if } i = j \\ -q_i g_i, & \text{otherwise,} \end{cases} \tag{13.89}$$

and

$$v(i,j) \quad = \quad \begin{cases} -q_j p_i g_j + (1 - p_i) q_j g_j, & \text{if } i = j \\ q_j g_j, & \text{otherwise.} \end{cases} \tag{13.90}$$

It follows form expressions (13.89) and (13.90) that $v_{ij} = -u_{Ij}$. Here one obtains the equilibrium by solving two linear programming problems (Owen, 1968)

$$\max_x\ U, \tag{13.91}$$

$$\sum_{i=1}^{m} x_i u_{ij} \geq U,\ j = 1, ..., m,$$

$$\sum_{i=1}^{m} x_i = 1,\ x_i \geq 0,$$

$$U = u_1 - u_2,\ u_1 \geq 0,\ u_2 \geq 0,$$

and

$$\min_y\ V, \tag{13.92}$$

$$-\sum_{j=1}^{m} y_j u_{ij} \leq V,\ i = 1, ..., m,$$

$$\sum_{i=1}^{m} y_i = 1,\ y_i \geq 0,$$

$$V = v_1 - v_2,\ v_1 \geq 0,\ v_2 \geq 0.$$

The important feature the zero-sum games is the mini-max condition

$$\min_y\ V = \max_x\ U. \tag{13.93}$$

4. Software Example

An interactive QBG model is implemented using Java JDK1.2 (Vaisnora, 2001). To start the program , one goes to one of web-sites (see Section 4) and selects "Discrete Optimization
Bimatrix Game Problem
Bimatrix Game, an example of bimatrix game optimization using LP, Java 1.3 and DS and QSE algorithms"

The applet starts by clicking 'index.html'. That is not secure mode thus all the data is introduced just by screen writing. Starting by the command line:
'appletviewer -J-Djava.security.policy=java.policy index.html'
one works in secure mode and the data can be introduced by files, for example:
'environment.txt'

```
#Wed Feb 18 18:01:29 EET 2004
V_EQUTION=-Pi*10+(1-Pi)*Qj*Gj;i\=j;\nQj*Gj;
U_EQUTION=Pi*Gi-(1-Pi)Gi*Qi;i\=j;\n0;\\
DATA_SAVE_URL=file\:/home/jonas/bimatrix/data1.txt
```

```
DATA_LOAD_URL=file\:/home/jonas/bimatrix/data-1.txt
VARIABLES=P G Q
RESULTS_SAVE_URL=
```

and
'data.txt'

```
P   G    Q
0.1 1.0  0.1
0.2 1.0  0.1
0.3 1.0  0.01
```

Figure 13.1 shows the input window i 2.4 and 2.4.

Figure 13.2 shows the input window defining the payoff functions of security game (13.9) (13.10)

Figure 13.3 shows the input window including data from the security game examples examples 2.4 and 2.4.

Figure 13.4 shows results obtained by solving IF equations (13.19 - (13.22) ignoring constrains (13.23

Figure 13.5 shows the input data for the reduced QBG obtained by eliminating the last raw and column.

Figure 13.6 shows the IF solution of the reduced QBG

Figure 13.7 shows the snapshot of the calculations[8]

Data is recorded by adding forests and by entering their parameters P, Q, G. To prevent unauthorized deals one enters 'Fine' and 'Fine's probability' defining the average penalties. The optimization starts by clicking the button $Calculate$.

The output window shows average wins in the form of matrices u[i,j] and v[i,j].
The results of DSA are presented as pairs of pure strategies, ignoring zero ones,

[8]One can see that the calculations do not correspond exactly to the QSE algorithm because this is an experimental software. The updated software is under development.

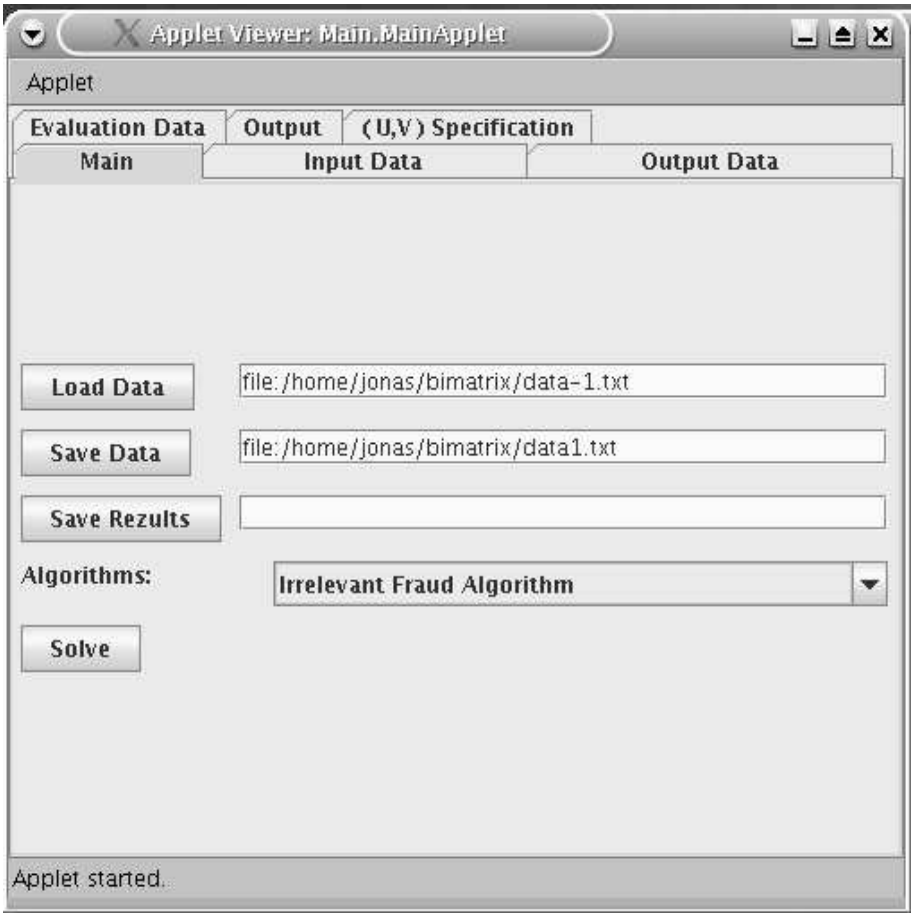


Figure 13.1. The input window.

no DSA solution is found,
The results of IFA shows that there is no IFA solution, since there are negative parameters, namely $x_3 = -8.621$‘.
Following QSE the third raw and the third column are eliminated and the QSE solution of the reduced QBG $x_1 = 0.667,\ x_2 = 0.333,\ y_1 = 1.0,\ y_2 = 0$ is found.

5. Future Developments

The first task is to update the experimental software providing the exact implementation of the QSE algorithm.

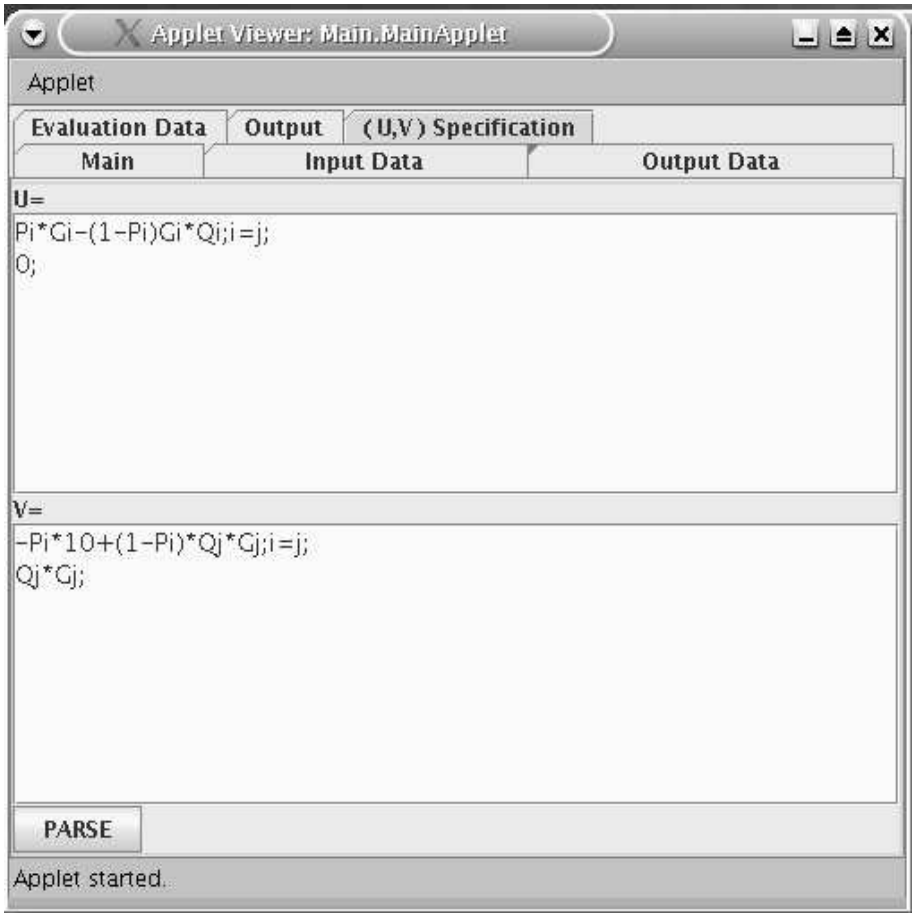


Figure 13.2. Specification of payoff functions.

The utility functions (13.5)(13.6) of both examples are selected to make models easy understandable without any special knowledge. That is important for studies. Most real life applications are described by more complicated utility functions. Therefore, the natural next step is to implement and to investigate these functions.

Statistical estimation of failures applying QSE algorithm for equilibrium search in the not well-defined cases could be useful because QSE is simple and the test by conditions (13.69) (13.70) is easy.

Optimization of resources preventing unauthorized deals (see section 2.5). is a challenging theoretical and practical problem,too.

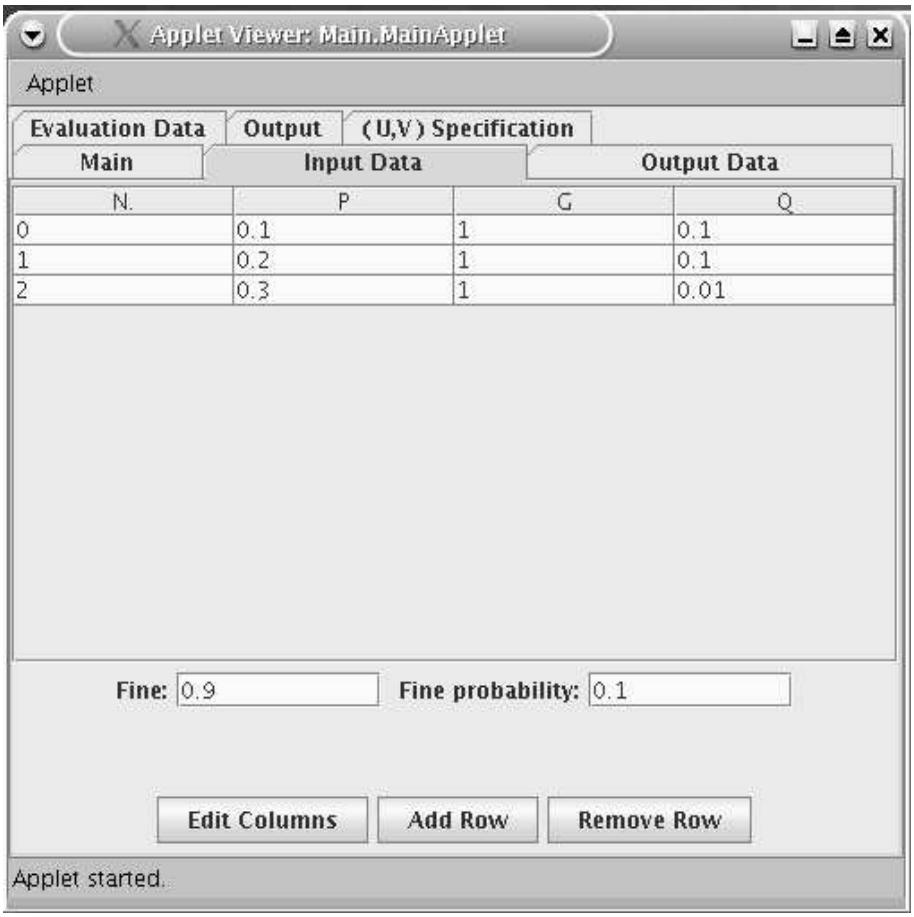


Figure 13.3. The input data of the original QBG.

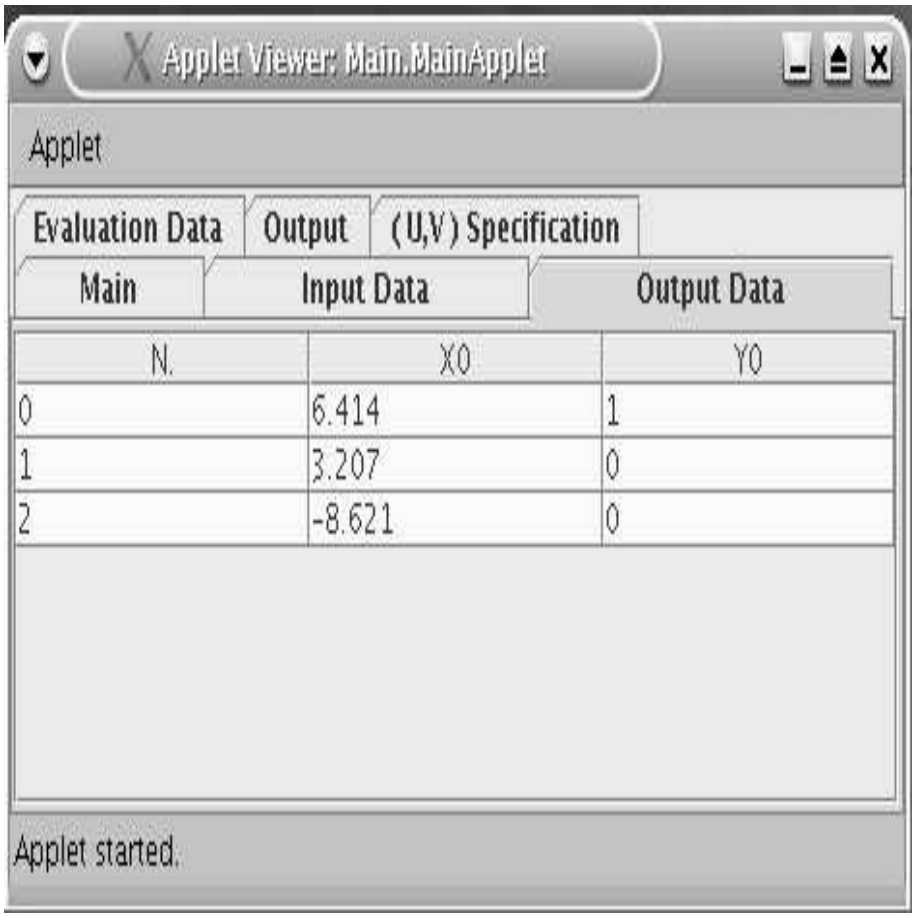


Figure 13.4. The IF results of the original QBG ignoring constrains.

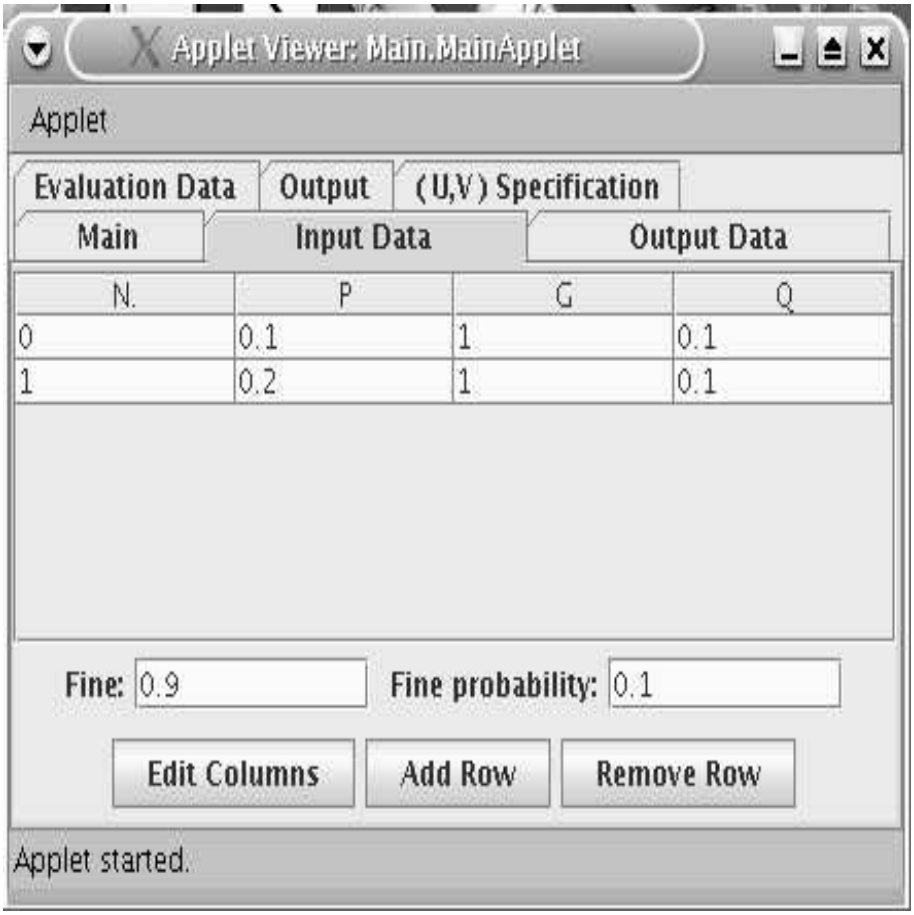


Figure 13.5. The input data for the reduced QBG.

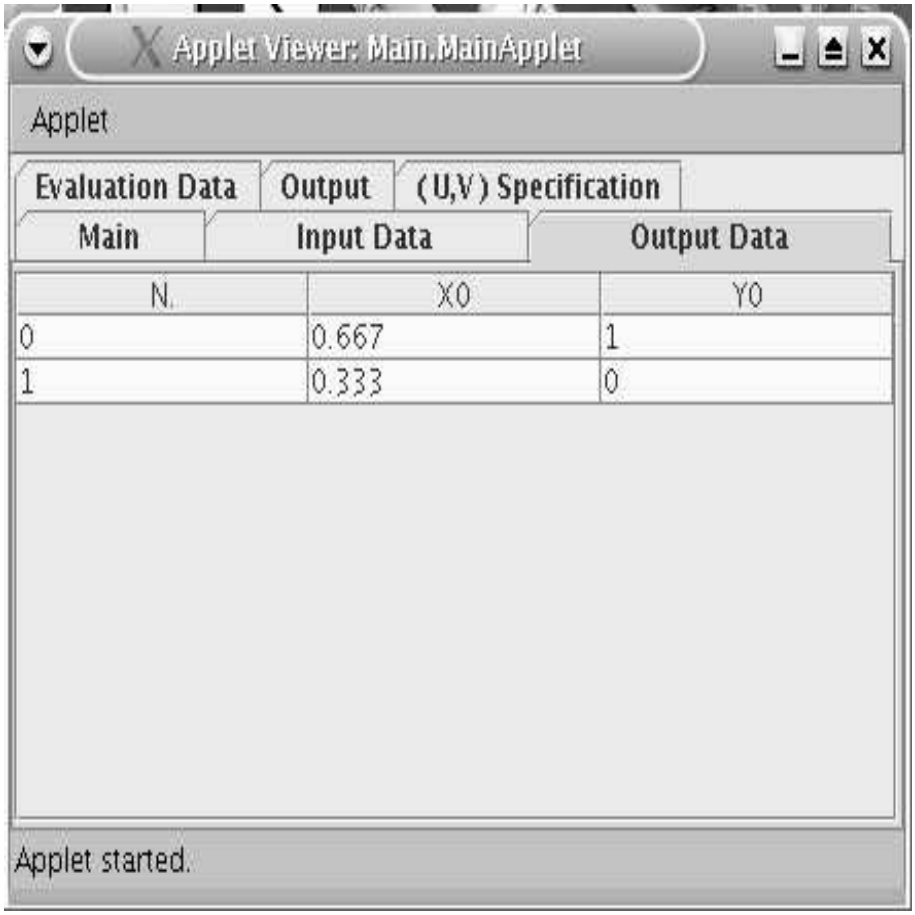


Figure 13.6. The IF solution of the reduced QBG.

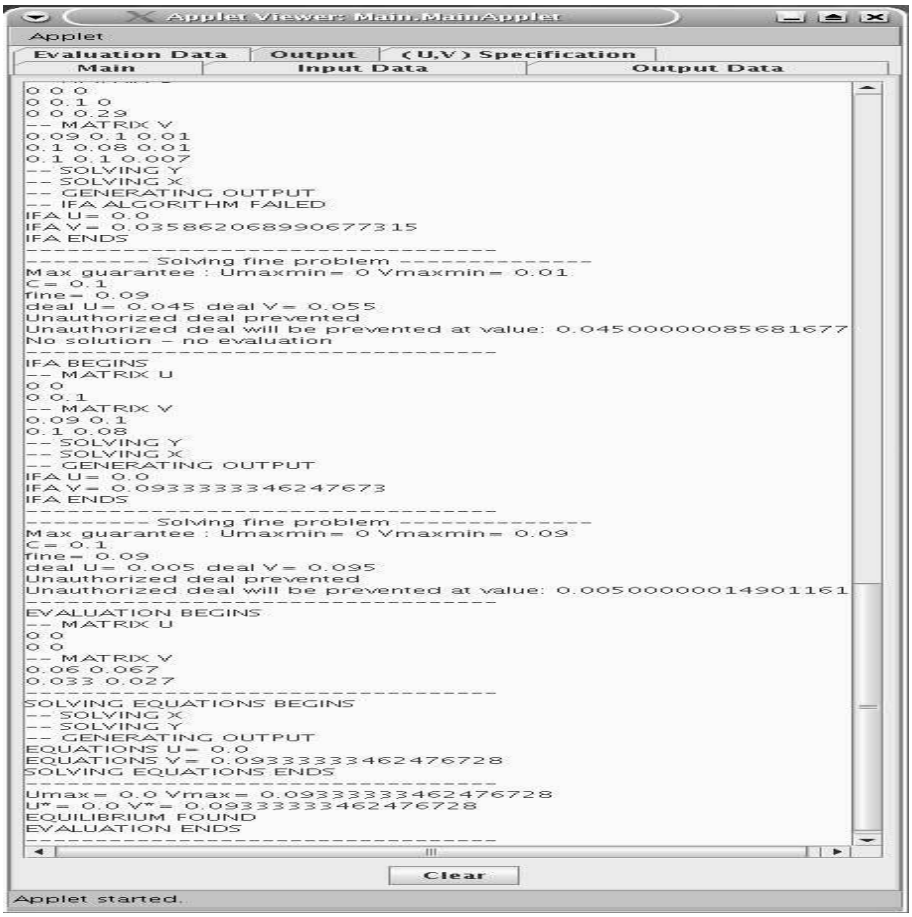


Figure 13.7. A snapshot of IF calculations.

Chapter 14

"PORTFOLIO" PROBLEM, OPTIMAL INVESTMENT OF RESOURCES

1. Introduction

The previous examples illustrated competition and inspection processes in economical, social and ecological problems. Here the optimal investment of available resources is considered. Investment problems depend on the nature of resources to be invested. An important part of any investment problem is a proper definition of utility functions that determine the profit-to-risk relation. Here we consider an illustrative example how to invest some fixed capital in Certificates of Deposit (CD) and Stocks.

The portfolio problem is to maximize the average utility of a wealth. That is obtained by optimal distribution of available capital between different objects with uncertain parameters (Mockus et al., 1997). Denote by x_i the part of the capital invested into an object i. The returned wealth is $y_i = c_i x_i$. Here $c_i = 1 + \alpha_i$ and $\alpha_i > 0$ is an interest rate. Denote by $p_i = 1 - q_i$ the reliability of investment. Here q_i is the insolvency probability. $u(y)$ is the utility the wealth y. Denote by $U(x)$ the expected utility function. $U(x)$ depends on the capital distribution $x = (x_i, ..., x_n),\ \sum_i = 1,\ x_i \geq 0$. If y is continuous, the expected utility function

$$U(x) = \mathbf{E}u(y) = \int_0^\infty u(y)p(y)dy. \tag{14.1}$$

Here $p(y)$ is probability density of wealth y. If the wealth is discrete $y = y^k,\ k = 1, ..., M$, the expected utility function

$$U(x) = \sum_{k=1}^{M} u(y^k)p(y^k). \tag{14.2}$$

Here M is the number of discrete values of wealth y^k. $p_x(y^k)$ is the probability that the wealth y^k will be returned, if the capital distribution is x. We search for such capital distribution x which provides the greatest expected utility of the returned wealth:

$$\max_x U(x), \tag{14.3}$$

$$\sum_{i=1}^{n} x_i = 1, \tag{14.4}$$

$$x_i \geq 0.$$

2. Expected Utility

2.1 Investment in CD

One may define probabilities $p(y^j)$ of discrete values of wealth $y^j,\ j = 1, 2, ...$ by exact expressions. For example,

$$\begin{aligned}
p(y^0) &= \prod_i q_i, \\
p(y^1) &= p_1 \prod_{i \neq 1} q_i, \\
p(y^2) &= p_2 \prod_{i \neq 2} q_i, \\
\ldots\ldots\ldots\ldots & \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
p(y^n) &= p_n \prod_{i \neq n} q_i, \\
p(y^{n+1}) &= p_1 p_2 \prod_{i \neq 1, i \neq 2} q_i, \\
p(y^{n+2}) &= p_1 p_3 \prod_{i \neq 1, i \neq 3} q_i \\
\ldots\ldots\ldots\ldots & \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots
\end{aligned} \tag{14.5}$$

Here
$y^0 = 0,\ y^1 = a_1 x_1,\ y^2 = a_2 x_2,\ y^n = a_n x_n,$

$y^{n+1} = a_1 x_1 + a_2 x_2,\ y^{n+2} = a_1 x_1 + a_3 x_3$. From expression (14.5)

$$U(x) = \sum_{k=1}^{M} u(y^k) p(y^k). \tag{14.6}$$

Here M is the number of different values of wealth y.

One determines $U(x)$ approximately by the Monte Carlo approach:

$$U_K(x) = 1/K \sum_{k=1}^{K} u(y^k). \tag{14.7}$$

Here

$$y^k = \sum_{i=1}^{n} y_i^k, \tag{14.8}$$

where

$$y_i^k \quad = \quad \begin{cases} c_i x_i, & \text{if } \eta_i^k \in [0, p_i] \\ 0, & \text{otherwise.} \end{cases} \tag{14.9}$$

Here K is the number of Monte Carlo samples. η_i^k is a random number uniformly distributed on the unit interval. In this case

$$U(x) = \lim_{K \to \infty} U_K(x).$$

2.2 Investment in CD and Stocks

Investing in CD, the interests α_i are defined by contracts. Only the reliabilities $p_i,\ i = 1, ..., n$ of banks are uncertain. Investing in stocks, in addition to reliabilities $p_i,\ i = n + j,\ j = 1, ..., m$ of companies, their future stock rates are uncertain, too. The predicted stock rates are defined by a coefficient a_i that shows the relation between the present and the predicted stock rates. The prediction "horizon" is supposed to be the same as the maturity time of CD.

To simplify the model suppose that one predicts L different values of relative stock rates $a_i^l,\ l = 1, ..., L$ with corresponding estimated probabilities $p_i^l,\ \sum_{l=1}^{L} p_i^l = 1,\ p_i^l \geq 0$.
In this case, one may define probabilities $p(y^i)$ of discrete values of wealth $y^i,\ i = 1, ..., n + m$ by exact expressions. The expressions for CD remains

the same. Therefore, we shall consider only stocks assuming that $n = 0$ and $L = 2$. Then

$$\begin{aligned}
p(y^0) &= \prod_i q_i, \\
p(y^1) &= p_1 p_1^1 \prod_{i\neq 1} q_i, \\
p(y^2) &= p_1 p_1^2 \prod_{i\neq 1} q_i, \\
p(y^3) &= p_2 p_2^1 \prod_{i\neq 2} q_i, \\
p(y^4) &= p_2 p_2^2 \prod_{i\neq 2} q_i, \\
\ldots\ldots\ldots\ldots & \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
p(y^{2n-1}) &= p_n p_n^1 \prod_{i\neq n} q_i, \\
p(y^{2n}) &= p_n p_n^2 \prod_{i\neq n} q_i, \\
p(y^{2n+1}) &= p_1 p_1^1 p_2 p_2^1 \prod_{i\neq 1, i\neq 2} q_i, \\
p(y^{2n+2}) &= p_1 p_1^2 p_2^p 2^2 \prod_{i\neq 1, i\neq 2} q_i, \qquad (14.10) \\
\ldots\ldots\ldots\ldots & \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots
\end{aligned}$$

Here
$y^0 = 0,\ y^1 = a_1^1 x_1,\ y^2 = a_1^2 x_1,\ y^3 = a_2^1 x_2, y^4 = a_2^2 x_2,$
$y^{2n-1} = a_n^1 x_n,\ y^{2n} = a_n^2 x_n,\ y^{2n+1} = a_1^1 x_1 + a_2^1 x_2,\ y^{2n+2} = a_1^2 x_1 + a_2^2 x_2.$
The reliabilities p_i, the stock rate predictions a_i^l and their estimated probabilities p_i^l are defined by experts, possibly, with the help of time series models such as ARMA (see Section 15). For example, maximal values of multi-step prediction are considered as "optimistic" estimates. The minimal values- as "pessimistic" ones. The average values of multi-step prediction are regarded as "realistic" estimates.

Here is a simplest illustration were $n = m = 1$ and $L = 2$. In this case from (14.5) (14.10) the probabilities $p(y^k)$ of wealth returns $y^k,\ k = 0, ..., 5$ are

$$\begin{aligned}
p(y^0) &= q_1 q_2, \\
p(y^1) &= p_1 q_2, \\
p(y^2) &= p_2 p_2^1 q_1, \\
p(y^3) &= p_2 p_2^2 q_1,
\end{aligned}$$

$$\begin{aligned} p(y^4) &= p_2 p_2^1 p_1, \\ p(y^5) &= p_2 p_2^2 p_1. \end{aligned}$$

Here
$y^0 = 0,\ y_1 = a_1 x_1\ y^2 = a_2^1 x_2,\ y^3 = a_2^2 x_2,$
$y^4 = a_1 x_1 + a_2^1 x_2,\ y^5 = a_1 x_1 + a_2^2 x_2.$

3. Optimal Portfolio, Special Cases

The optimal portfolio depends on the utility function $u(y)$. Consider, for example, the optimal portfolio for three different utility functions.

The first utility function is linear

$$u(y) = cy. \tag{14.11}$$

This function is for "rich" persons. Rich persons want to maximize the average wealth. They are not emotional about accidental losses or gains. In the linear case (14.11), the optimal portfolio is to invest all the capital in an object with the highest product $p_i c_i$.

The second utility function is for "prudent" persons which averse risk

$$u(y) = \begin{cases} 0, & \text{if } 0 \le y < a \\ 1, & \text{if } a \le y \le c, \end{cases}. \tag{14.12}$$

Here a is a risk threshold. $c = \max_i c_i x_i$ denotes the maximal return of invested capital (see expression (14.4)). If $a = 1/m\ min_i\ c_i x_i$ then, in the risk-averse case (14.12), an optimal decision is $x_i^* = 1/m,\ i = 1, ..., m$. Here one divides the capital equally between all the objects[1].

The third utility function is for "risky" persons. Risky persons are ready to risk for the great win c.

$$u(y) = \begin{cases} 0, & \text{if } 0 \le y < c, \\ 1, & \text{if } y = c. \end{cases} \tag{14.13}$$

Here one invests all the capital in the object with highest wealth return. Therefore, $x_i = 1$, if $c_i = \max_j c_j = c$.

[1]The optimal decision $x = 1/m$ is not unique, any decision satisfying the inequality $c_i\, x_i \ge a,\ i = 1, ..., m$ minimizes the expected utility function $U(y)$.

These examples are abstract. An average person behaves "risky," if only a small part of his resources is involved. The same person behaves prudently, if all his wealth is at stake. There is a point r between areas of risky and prudent behavior. At this point an average person behaves like the "rich" one. Here is an example

$$\begin{aligned} u(y) < y,\ if\ 0 \le y < r, \\ u(y) = y,\ if\ y = r, \\ u(y) > y,\ if\ r < y \le c. \end{aligned} \tag{14.14}$$

Here r is a boundary point between risky and prudent areas.

4. Optimal Insurance

In this section the optimal insurance is regarded as a special case of optimal investment. Then the expected utility at the end of the next year

$$U(x) = \sum_{k=1}^{M} u(y^k)p(y^k), \tag{14.15}$$

where $p(y^k)$ is a probability to own next year the wealth y^k,
$u(y^k)$ is an utility function of the wealth y^k.
Supose that
$y^k = \sum_{i=1}^{m} c_i(x_i)$
and

$$c_i(x_i) \quad = \quad \begin{cases} z_i - a_i x_i, & \text{jei } \delta_i = 1 \\ (1 - a_i)x_i, & \text{jei } \delta_i = 0, \end{cases}. \tag{14.16}$$

Here z_i is the value of the object i ,
the product $a_i x_i$ denotes insurance charge of the object i.
$x_i \le z_i$ is insurance policy of the object i.
$\delta_i = 1$, if the object i survives,
$\delta_i = 0$, if not.
$p_i = P\{\delta_i = 1\}$ is a survival probability of the object i .
For example:
$p(y^1) = p_1 \prod_{i \ne 1}(1 - p_i)$,
$y^1 = c_1(x_1) + \sum_{i=2}^{m} c_i(x_i)$,
where
$c_1(x_1) = z_1 - a_1 x_1$,
$c_i(x_i) = (1 - a_i)x_i,\ i = 2, ..., m.$

5. Utility Functions

Utility functions $u(y)$ are different for different persons and organizations. An individual utility function is defined by a lottery
$L(A, B, p) = \{pA + (1-p)B\}$. Here p is the probability to win the best event A.
$(1-p)$ is the probability to get the worst one B. Denote by C the "ticket price" of this lottery. There are two possible decisions:

- keep the ticket money C,
- bay a ticket and risk losing this money while hoping to win a greater wealth A with probability p.

Denote by $p(C)$ a "hesitation" probability, when one cannot decide which decision to prefer. One defines the "hesitation" probability $p(C)$ by this condition

$$L(A, B, C, p(C)) = [C \approx \{p(C)A + (1 - p(C))B\}]. \tag{14.17}$$

Here the symbol $\approx$ denotes the "hesitation." If utilities $u(A) = 1$ and $u(B) = 0$, the utility of the "ticket" C is equal to the hesitation probability $u(C) = p(C)$ (Fishburn, 1964).

Suppose, for example, that event C is to keep all the investment capital, $y = 1$, in a safe; no risk, no profit. Assume that the event A means doubling the capital, $y = 2$. The event B means losing all the capital, $y = 0$.

Denote by $p(1)$ the hesitation probability. Then
$u(1) = u(0) + p(1)(u(2) - u(0))$. If $u(0) = 0$ and $u(2) = 1$ then the utility of the capital $u(1) = p(1)$. Here one obtained capital utilities at three points: $y = 0,\ y = 1$, and $y = 2$.

To define a reasonable approximation of the utility function $u(y)$, we need at least two additional points. For example, points $y = 0.5$ and $y = 1.5$. One defines the corresponding utilities by the hesitation probabilities $p(0.5)$ and $p(1.5)$. These are obtained by two hesitation lotteries

$$L(1.0, 0.0, 0.5, p(0.5)) = \tag{14.18}$$
$$[(y = 0.5) \approx \{p(0.5)(y = 1) + (1 - p(0.5))(y = 0)\}]$$

and

$$L(2.0, 1.0, 1.5, p(1.5)) = \tag{14.19}$$
$$[(y = 1.0) \approx \{p(1.5)(y = 2.0) + (1 - p(1.5))(y = 1)\}].$$

Here one obtains utility values
$u(0) = 0,\ u(0.5) = p(0.5),\ u(1) = p(1),\ u(1.5) = u(1) + p(1.5)(u(2) - u(1))\ u(2) = 1$. The remaining capital utility values are defined by the linear interpolation

$$u(y) = u(y_i) + p(y_i)(u(y_{i+1}) - u(y_i)),\ \ y_i \le y < y_{i+1}, \qquad i = 0, 1, ..., 4. \tag{14.20}$$

In consulting offices, the "psychological tests" defining capital utilities are not always convenient. Then one of the four "typical" utility functions can be selected. The typical utility functions represent the risky, the average, the rich and the prudent persons. The selection depends on observable personal traits.

The same capital utility function (14.14) could be used for all customers. Then one defines customer differences by different border points, namely: $r_{prudent} < r_{average} < r_{rich} < r_{risky}$ (see Figure 14.2).

6. Software Example

6.1 Running the Program

The software (Petrikis and Juodgalvyte, 1997) is on the web-site (see Section 4) and is run by Internet

1 open the line 'GMJ1' on the 'Software systems' page (see Figure 3.1),

2 start the applet,

3 open the 'Methods' menu, choose the method and its parameters (see Figure 14.1),

4 open the 'Tasks' menu, chose the task 'Portfolio',

5 select the character : prudent, rich, risky, average (see Figure 14.2),

6 select the method: Monte Carlo, Deterministic (see Figure 14.2),

7 fix the predicted stock rates by updating or keeping the default values determined using ARMA model (see Figure 14.2),

8 click at the 'Operation' label at the top of page,

9 click at the 'Run' button at the bottom of page,

10 read results of the optimal investment, where 'Iteration' means the number of iterations to reach the optimal investment, 'F(x)' shows the expected utility with the minus sign, the following lines show the relative investments[2] (see Figure 14.3),

11 open the 'convergence' window (see Figure 14.4) which show how the best utility depends on the iteration number,

12 open the 'projection' window (see Figure 14.5) showing how the utility depends on the first variable,

13 open the 'spectrum' window showing how a frequency of utilities depends on the iteration number[3].

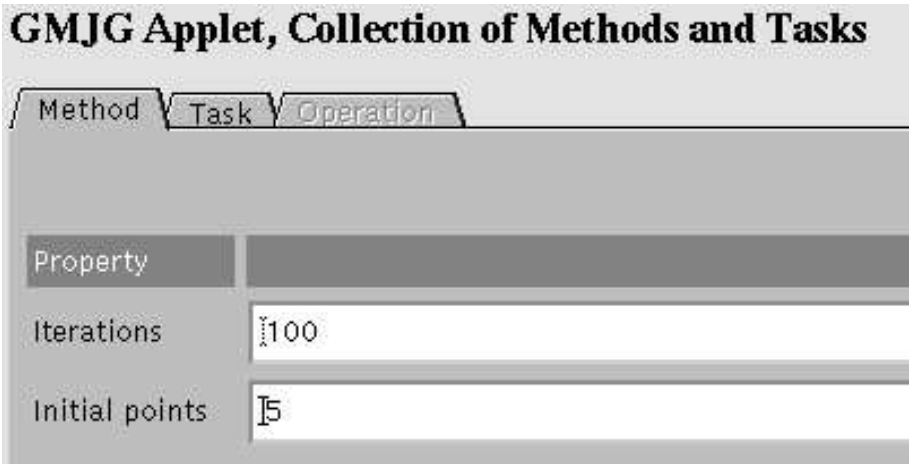


Figure 14.1. The control window.

The example of predicted stock rate of Vilnius Bankas is presented in the file 'onestep?VB.txt'.

6.2 A Set of Utility Functions

Nine approximation points are used: $yk[i] = 0.25 * i,\ i = 0, ..., 8$. Four utility functions are available. They differ at the approximation points. They represent different persons.

[2]The actual investment is the product of the total investment and the relative investment, divided by the sum of all the relative investments.
[3]Different frequencies are is denoted by different colors.

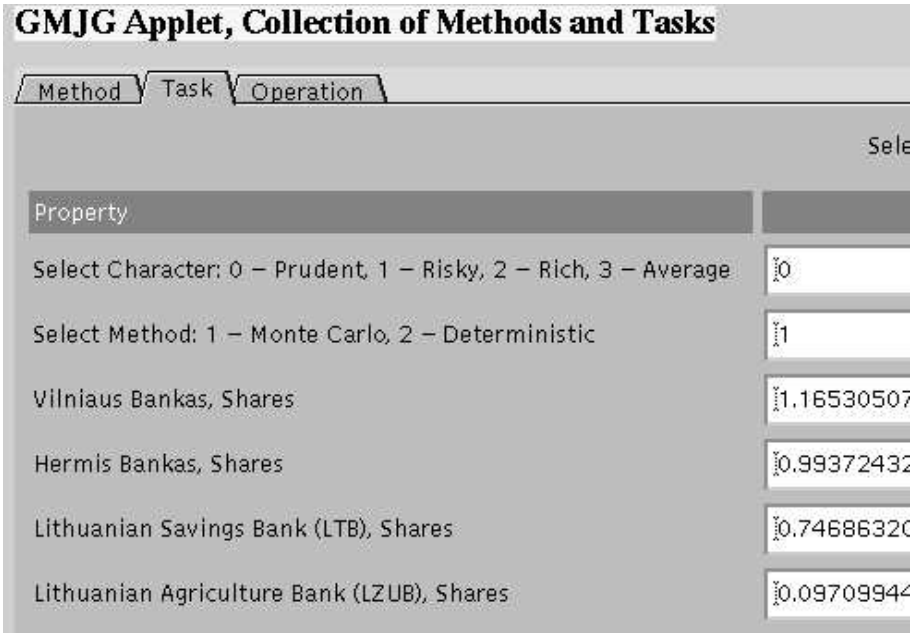


Figure 14.2. The input fields.

prudent person : $f[0] = 0.0,\ f[1] = 0.3,\ f[2] = 0.5,\ f[3] = 0.7,\ f[4] = 0.8,\ f[5] = 0.85,\ f[6] = 0.9,\ f[7] = 0.95,\ f[8] = 1.0,$

risky person : $f[0] = 0.0,\ f[1] = 0.1,\ f[2] = 0.2,\ f[3] = 0.25,\ f[4] = 0.3,\ f[5] = 0.4,\ f[6] = 0.7,\ f[7] = 0.9,\ f[8] = 1.0,$

rich person : $f[0] = 0.0,\ f[1] = 0.125,\ f[2] = 0.25,\ f[3] = 0.325,\ f[4] = 0.5,\ f[5] = 0.625,\ f[6] = 0.75,\ f[7] = 0.875,\ f[8] = 1.0,$

average person $f[0] = 0.0,\ f[1] = 0.1,\ f[2] = 0.2,\ f[3] = 0.25,\ f[4] = 0.5,\ f[5] = 0.7,\ f[6] = 0.85,\ f[7] = 0.95,\ f[8] = 1.0.$

Users select one of them.

6.3 Predicting Investment Results

Two investment possibilities are considered: Certificates of Deposit (CD) and shares. Considering CD, the two parameters should be fixed: the interest and the reliability (bank's survival probability). These parameters are defined in the file $Portfolio.java$. Considering shares, only one parameter is used, the

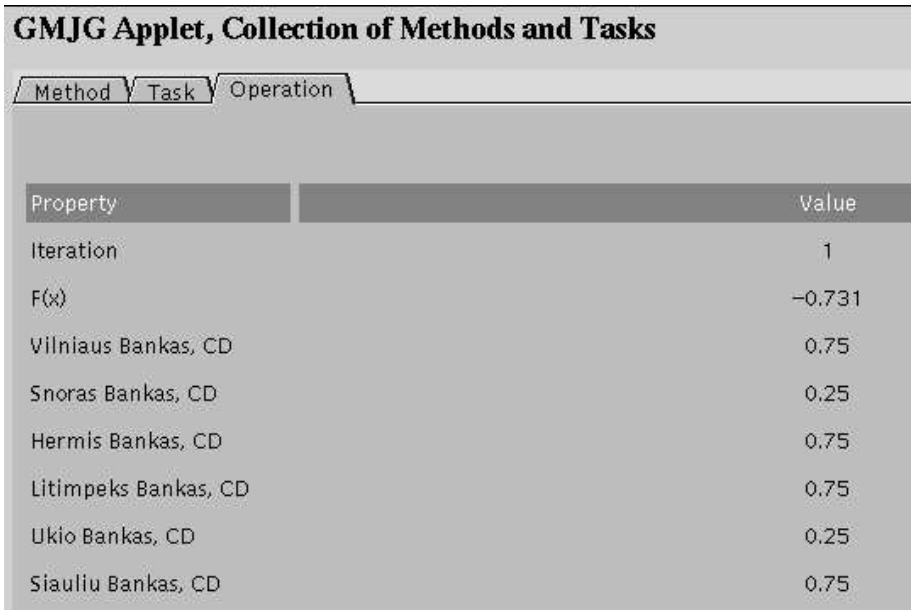


Figure 14.3. The output field.

predicted stock rate at the time of CD maturity. That is correct, if the investor is confident[4] about the predicted stock rates at the maturity time. The default values of expected changes of stock rates are zero. The predicted stock rates are given in the files like this one: $onestep?VB.txt$.

```
Input file = VB.txt
Pradzia = 26.000000 Prognoze = 30.297932 </BODY></HTML>
```

Here 'VB' denotes the Vilnius Bank stock rates. 'Pradzia' denotes the rates now. 'Prognoze' denotes the predicted rates. The difference between predicted and present stock rates, divided by the present stock rate and expressed in percents, is considered as a "stock interest." That is an important simplification. However, it helps to consider both shares and CD in the same way. That is convenient for calculations.

Predicted rates in $VB.txt$ files are obtained by the ARMA model using the Vilnius Bank data. The predicted values can be changed manually by clicking and writing on the corresponding fields of the 'Tasks' page.

[4]Without the "confidence" assumption one should consider stock rates predictions errors, too.

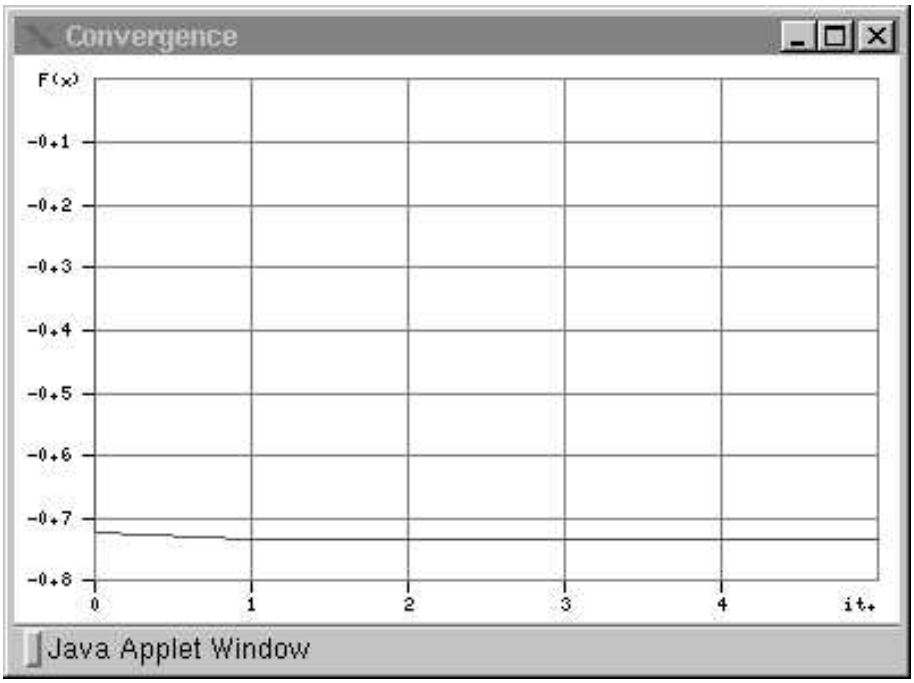


Figure 14.4. The convergence window.

6.4 Data

The stock rates of the following major Lithuanian joint-stock companies, including the banks, are considered.

Bank "Vilniaus Bankas" : data file VB.txt,

Bank "Snoras" : data file Snoras.txt,

Bank "Hermis" : data file Hermis.txt,

Savings bank "LTB" : data file LTB.txt,

Agricultural bank "LZUB" : data file LZUB.txt,

Bank "Siauliu Bankas" : data file Siauliu.txt,

Economic bank "Ukio Bankas" data file Ukio.txt,

Dairy "Rokiskio Suris" : data file Rokiskio.txt,

Brewery "Kalnapilis" data file Kalnapilis.txt,

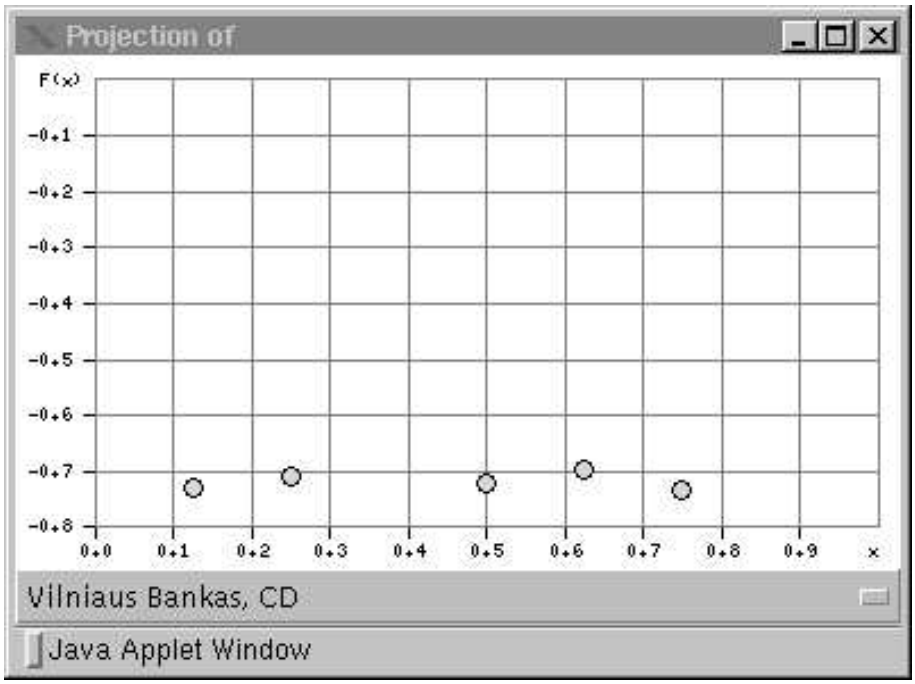


Figure 14.5. The projection window.

Dairy "Birzu Pienas" : data file BirzuAB.txt.

The first record of the data files[5] shows closing rates on January 01, 1998. The last record defines closing rates on November 14, 1998. Two hundred twenty nine sessions of Lithuanian stock exchange are recorded in the data. Using this data ARMA model makes ninty day predictions what corresponds to three month CD.

6.5 Results

Optimization results are in Tables 14.1, 14.2, and 14.3. Tables 14.1 and 14.2 both are for a prudent person. Table 14.1 is obtained by the Monte Carlo method. Table 14.2 shows results of the deterministic method. Table 14.3 is obtained by the deterministic method for a risky person.

[5]The data file is on the web-sites (see Section 4).

The "Optimal investment" column is just a copy of the output field (see Figure 14.3). The output field represents optimal values of internal variables. The internal variables are proportional to the optimal parts of the capital. Actual optimal parts are obtained by normalization [6] and are shown in the "Optimal part" column.

Company	Optimal investment	Optimal part
Vilniaus Bankas, CD	0.157	0.0251
Snoras, CD	0.354	0.0566
Hermis, CD	0.402	0.0643
Litimpex, CD	0.724	0,1159
Ukio Bankas, CD	0.334	0.0535
Siauliu Bankas, CD	0.842	0.1348
Medicinos Bankas, CD	0.096	0.0153
State bonds	0.318	0.0509
Vilniaus Bankas, shares	0.626	0.1002
Snoras, shares	0.677	0.1083
Hermis, shares	0.293	0.0469
Ukio Bankas, shares	0.466	0.0746
Siauliu Bankas, shares	0.461	0.0738
Rokiskio Suris, shares	0.108	0.0169
Birzu Pienas, shares	0.061	0.0097
Kalnapilis, shares	0.146	0.02337

Table 14.1. Monte Carlo method. Prudent person. Optimal portfolio obtained after 892 iterations. The utility is 0.773.

Tables 14.1 and 14.2 show that the optimal distribution of capital obtained by the approximate method differ from that defined by the exact one. This is important disadvantage of the approximate Monte Carlo method. Note, that the best values of utility functions are close for both methods. They are 0.773, for the Monte Carlo method, and 0.776, for the deterministic one. This means that the optimal values of utility functions are not very sensitive to moderate changes in the capital distribution. The reason is that here investment alternatives are not too different.

Comparing Table 14.2 representing the prudent person and Table 14.3 representing the risky one, we see differences. The best value of utility function is equal to 0.776, for the "prudent." It is equal to 0.291, for the "risky." This difference is explained by assumptions made to normalize utility functions. The

[6] Users make the normalization in this software version.

Company	Optimal investment	Optimal part
Vilniaus Bankas, CD	0.699	0.0755
Snoras, CD	0.453	0.0489
Hermis, CD	0.836	0.904
Litimpex, CD	0.795	0,0859
Ukio Bankas, CD	0.366	0.0395
Siauliu Bankas, CD	0.783	0.0846
Medicinos Bankas, CD	0.926	0.1001
State bonds	0.589	0.0636
Vilniaus Bankas, shares	0.937	0.1001
Snoras, shares	0.361	0.0390
Hermis, shares	0.343	0.0370
Ukio Bankas, shares	0.002	0.0002
Siauliu Bankas, shares	0.944	0.1020
Rokiskio Suris, shares	0.760	0.0822
Birzu Pienas, shares	0.07	0.00075
Kalnapilis, shares	0.052	0.0056

Table 14.2. Exact o method. Prudent person. Optimal portfolio obtained after 892 iterations. The utility is 0.776.

Company	Optimal investment	Optimal part
Vilniaus Bankas, CD	0.907	0.107
Snoras, CD	0.083	0.0098
Hermis, CD	0.857	0.101
Litimpex, CD	0.438	0,0517
Ukio Bankas, CD	0.024	0.00283
Siauliu Bankas, CD	0.737	0.0870
Medicinos Bankas, CD	0.739	0.872
State bonds	0.656	0.0774
Vilniaus Bankas, shares	0.762	0.0899
Snoras, shares	0.179	0.0211
Hermis, shares	0.894	0.1055
Ukio Bankas, shares	0.451	0.0532
Siauliu Bankas, shares	0.098	0.011
Rokiskio Suris, shares	0.946	0.112
Birzu Pienas, shares	0.019	0.00224
Kalnapilis, shares	0.076	0.0089

Table 14.3. Exact method. Risky person. Optimal investment obtained after 159 iterations. The utility is 0.291.

normalization assumes that the maximal utility is a unit and the minimal one is zero. Therefore, the best value of utility function of the risky person happens to

be less then that of the prudent one. However, lower utilities does not mean that risky persons are getting less satisfaction. One needs more sophisticated normalization techniques to compare correctly the satisfaction scales of different persons by their utility functions.

6.6 Future Developments

The first step is to drop the "confidence" assumption that one predicts the stock rates exactly. To do this, the variance of the predicted stock rates should be included into the updated Portfolio Model.

The important future task is the sensibility analysis. For example, by considering three scenarios: pessimistic, optimistic and realistic. In the pessimistic scenario lower probabilities p_i for CD's and lower interest rates for stocks are considered. In the optimistic case one sets higher values for these parameters. The realistic case is in the middle. Important task is to represent visually the "stability" of securities [7]. If the optimal values of some securities are not too different in all three scenarios, they are considered as stable.

[7] CD and Stocks.

7. Minimization of Road Accident Damages

Here we consider an illustrative example how to invest some fixed capital to minimize the damages of the road accidents. The problem is split in two stages. In the first stage, the resources are distributed between groups of accidents defined by their causes.
In the second stage, the geographical distribution of funds inside a single group is regarded. The four causes of road accidents are considered:

- intoxication,
- speeding,
- ignoring safety belts,
- bad roads- "black spots."

Denote by x_i a part of the funds spent to reduce accidents related to the cause i. In this section, we shall assume for simplicity that the total amount of funds is unit. Here index $i = 1$ denotes intoxication, $i = 2$ means speeding, $i = 3$ denotes safety belts, and $i = 4$ denotes black spots.

Denote by $y_i = y_i(x_i),\ i = 1, .., 4$ the expected number of accidents in relation to the part of funds x_i. Suppose that

$$y_i = y_i(x_i) = c_i e^{-a_i x_i}, \tag{14.21}$$

where

$$(c_i, a_i) = arg \min_{(c,a)} \sum_{k=1}^{K} (ce^{-ax_i^k} - N_i^k)^2. \tag{14.22}$$

Here N_i^k is the actual number of accidents when the part of funds was x_i^k.

7.1 Distribution Between Groups

Denote the estimate of expected losses from all four groups of accidents when the distribution of funds between causes are defined by the vector $x = (x_i,\ i = 1, ..., 4)$

$$u(x) = \sum_{i=1}^{4} h_i y_i(x_i). \tag{14.23}$$

Here h_i is the estimate of expected damages of an accident of the group i. If data is available, then h_i is just average value. If not, then h_i is some expert estimate.

7.2 Geographical Distribution

The distribution between black spots will be regarded. Denote the distribution vextor by $x = (x_i,\ i = 1, ..., m)$, where m is a number of black spots. x_i is the part of funds invested to improve the black spot i. Denote by $y_i = y_i(x_i),\ i = 1, .., m$ the expected number of accidents in relation to the part of funds x_i.

$$y_i = c_i e^{-a_i x_i}, \tag{14.24}$$

where

$$(c_i, a_i) = arg \min_{(c,a)} \sum_{k=1}^{K} (ce^{-ax_i^k} - N_i^k)^2. \tag{14.25}$$

Here N_i^k is the actual number of accidents at the black spot i when the part of funds was x_i^k.

7.3 Definition of Optimal Investment

The formal definition is

$$x^* = arg \min_x u(x) \tag{14.26}$$

$$u(x) = \sum_{i=1}^{m} h_i y_i, \tag{14.27}$$

$$\sum_{i=1}^{m} x_i = 1, \tag{14.28}$$

$$x = (x_i,\ i = 1, ..., m),\ , x_i \geq 0. \tag{14.29}$$

Here h_i is the estimate of expected damages of an accident at the black spot i. If data is available, then h_i is just average value. If not, then h_i is some expert estimate.

The direct minimization of the m-dimensional utility function (14.27) is difficult if m is large. In the case (14.27) the dynamic programming is an appropriate optimization technique.

7.4 Dynamic Programming

The dynamic programming is a conventional technique to optimize sequential decisions (Bellman, 1957). Applying the dynamic programming to specific problems, one develops specific algorithms, as usual.

Bellman's Equations

One starts from the last black spot m defining by v a part of funds left for the spot m. Then

$$u_m(v) = h_m y_m(v) \tag{14.30}$$

$$v_m(v) = v \tag{14.31}$$

$$u_{m-i}(v) = \min_{0 \le v \le 1} (h_{m-i} y_{m-i}(v) + u_{m-i+1}(v)) \tag{14.32}$$

$$v_{m-i} = arg \min_{0 \le v \le 1} (h_{m-i} y_{m-i}(v) + u_{m-i+1}(v)), \tag{14.33}$$

$$i = 1, ..., m-1 \tag{14.34}$$

Here the relation $u_{m-i}(v)$ shows how the maximal utility that can be obtained if the part of funds v designated for the set of black spots starting from $m-i$ is distributed in the best possible way. The relation $v_{m-i}(v)$ shows how the optimal part of funds for the $m-i$ black spot depends on the part of funds designated for the set of black spots starting from $m-i$. Here $i = 1, ..., m-1$.

Reurrent equations (14.32) and (14.33) define the optimal parts of funds $v_{m-i}(v)$ as a function of v. The actual optimal values of these funds x^* are obtained by substitutions assuming that the total amount of funds is unit.

$$x_1^* = v_1(1) \tag{14.35}$$

$$x_2^* = v_2(1 - x_1^*) \tag{14.36}$$

$$x_3^* = v_3(1 - x_1^* - x_2^*) \tag{14.37}$$

.....

$$x_m^* = v_m(1 - x_1^* - x_2^* - ... - x_{m-1}^*). \tag{14.38}$$

The minimization of the road accidents is just a good ilustration. Similar procedures can be used in many other problems of optimal distribution of funds.

Chapter 15

EXCHANGE RATE PREDICTION, TIME SERIES MODEL

1. Introduction

Solutions of optimal investment problems, considered in the previous chapter, depend on predicted stock rates. Predictions are needed considering most of the investment problems, because the result of an investment depends on future values of various parameters. Predictions are an important part of scheduling problems, too. The optimal schedules depend on the predicted demand, and supply. Time series models are common prediction tools. Here we investigate a well-known time series model. The so called, Auto Regressive Moving Average (ARMA) model. We briefly consider the Artificial Neural Network (ANN) and the Bilinear models, too.

Mainly, the financial data is predicted. The prediction of call rates, using the same models, are briefly mentioned, just for comparison. The call rate prediction is described in detail considering call center optimization problems in Chapter 16. There the traditional ARMA is supplemented by an expert model.

Modeling economic and financial time series by ARMA has attracted the attention of many researchers in recent years (Diebold and Rudebusch, 1989, Cheung, 1993, Yin-Wong and Lai., 1993, Cheung and Lai, 1993, Koop et al., 1994, Mockus and Soofi, 1995). Three approaches have been used to estimate the parameters of the ARMA models, : Maximum Likelihood (ML) (Sowel, 1992), approximate ML (Li and McLeod, 1986, Fox and Taqqu, 1986, Hosking, 1981, Hosking, 1984), and two-step procedures (Geweke and Porter-Hudak,

1983, Janacek, 1982). In all the cases local optimization techniques were applied. Here the optimization results depend on the initial values, what implies that one cannot be sure if the global maximum is found.

The global optimization is very difficult in many cases[1]. The reason is a high complexity of multi-modal optimization problems. It is well known (Ko, 1991) that optimization of polynomial-time computable real functions cannot be done in polynomial-time, unless $P = NP$[2]. In practice, this means that we need an algorithm of exponential time to obtain the ϵ-exact solution. The number of operations in exponential algorithms grows exponentially with the accuracy m of solution and dimension n of the optimization problem. The accuracy m means that $\epsilon \leq 2^{-m}$. The dimension n means that one optimizes a function $f(x)$ where $x = (x_1, ..., x_n)$.

The Least Squares (LS) method is a popular approach to estimate parameters of ARMA models. Using LS one minimizes the log-sum of square residuals using ARMA models and their extensions (Mockus and Soofi, 1995). In this chapter, the multi-modality problems are considered using different data. The data include:
daily exchange rates of \$/\$ and DM/\$,
closing rates of stocks of AT&T , Intel Co, and some Lithuanian banks,
the London stock exchange index (Raudys and Mockus, 1999),
call-rates of a commercial call-center.
The graphical images and comparison of the average prediction results of ARMA and the Random Walk (RW) models are presented.

2. Auto Regressive Moving-Average Models (ARMA)

2.1 Definitions

We define ARMA model as

$$w_t = \sum_{i=1}^{p} a_i w_{t-i} + \sum_{i=1}^{q} b_i \epsilon_{t-i} + \epsilon_t. \tag{15.1}$$

[1]By the term 'difficult' we mean the time measure of computational complexity, that is, the length of time would be needed for a standard universal computer to do a task.
[2]The notation $P = NP$ means the existence a polynomial-time algorithm P for solving NP-complete problems. That is merely a theoretical possibility.

We assume that

$$z_{t-i} = 0,\ w_{t-i} = 0,\ \epsilon_{t-i} = 0,\ \ if\ \ t \leq i. \tag{15.2}$$

2.2 Definition of Residuals

An advantage of residual minimization is that one may see directly how the objective depends on unknown parameters. Using equalities (15.1) we define residuals by recurrent expressions

$$\begin{aligned} \epsilon_1 &= w_1, \qquad (15.3)\\ \epsilon_2 &= w_2 - a_1 w_1 - b_1 \epsilon_1, \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \epsilon_t &= w_t - a_1 w_{t-1} - ... - a_p w_{t-p} - b_1 \epsilon_{t-1} - ... - b_q \epsilon_{t-q}. \qquad (15.4) \end{aligned}$$

Next the sum

$$f(x) = \log f_m(x),\ \ f_m(x) = \sum_{t=1}^{T} \epsilon_t^2 \tag{15.5}$$

is minimized. The logarithm is there to decrease the objective variation by improving the scales.

3. Minimization of Residuals of ARMA Models

Here we consider an algorithm for optimization of parameters of the ARMA model. This model is simple and may be considered as a good first approximation. Denote by y_t the value of y at the time t. Denote by $a = (a_1, ..., a_q)$ a vector of auto-regression (AR) parameters, and by $b = (b_1, ..., b_q)$ a vector of moving-average (MA) parameters.

$$y_t - \sum_{i=1}^{p} a_i y_{t-i} = \epsilon_t - \sum_{j=1}^{q} b_j \epsilon_{t-j},\ t = 1, ..., T. \tag{15.6}$$

The residual

$$\epsilon_t = y_t - \sum_{i=1}^{p} a_i y_{t-i} + \sum_{j=1}^{q} b_j \epsilon_{t-j}, \tag{15.7}$$

or

$$\epsilon_t = B_t + \sum_{i=1}^{p} a_i A_t(i). \tag{15.8}$$

Here

$$B_t = y_t + \sum_{j=1}^{q} b_j B_{t-j-1}, \tag{15.9}$$

and

$$A_t(i) = -y_{t-i-1} + \sum_{j=1}^{q} b_j A_{t-j-1}, \tag{15.10}$$

where $t - i > 0$ and $t - j > 0$.

3.1 Optimization of AR Parameters

Denote

$$S(a, b) = \sum_{t=1}^{T} \epsilon^2, \tag{15.11}$$

where $a = (a_1, ..., a_p)$ and $b = (b_1, ..., b_q)$. From expressions (15.11) and (15.8) the minimum condition is

$$\frac{\partial S(a, b)}{\partial a_j} = 2 \sum_{t=1}^{T} \epsilon_t A_t(j) = 0, \ j = 1, ..., p, \tag{15.12}$$

or

$$\sum_{i=1}^{p} A(i, j) a_i = -B(j), \ j = 1, ..., p. \tag{15.13}$$

Here

$$A(i, j) = \sum_{t=1}^{T} A_t(i) A_t(j), \tag{15.14}$$

and

$$B(j) = \sum_{t=1}^{T} A_t(j) B_t. \tag{15.15}$$

The minimum of expression (15.11) at fixed parameters b is defined by the system of linear equations

$$a(b) = A^{-1}B. \tag{15.16}$$

Here matrix $A = (A(i,j),\ i,j = 1,...p)$. Vector $B = (B(j),\ j = 1,...p)$, where elements $A(i,j)$ are from (15.14), components $B(j)$ are from (15.15), and A^{-1} is an inverse matrix A. This way one define the vector $a(b) = (a_i(b),\ i = 1,...,p)$ that minimize sum (15.11) at fixed parameters b.

3.2 Optimization of MA Parameters

The sum of squared residuals (15.11) is a non-linear non-convex function of parameters b. Thus, we have to consider the global optimization algorithms. Denote

$$f(x) = \log S(a(x), x). \tag{15.17}$$

Here $x = b$ and $S(a,b)$ is from (15.11) at optimal parameter $a = a(b)$. Denote

$$b^0 = x^0 = arg\min_x f(x). \tag{15.18}$$

3.3 Predicting "Next-Day" Rate

Predicting "Next-Day" Rate, one minimizes the expected "next-day" squared deviation ϵ_{t+1} using the data available at the moment t

$$y^0_{t+1} = arg\min_{y_{t+1}} \mathbf{E}\ \epsilon^2_{t+1}. \tag{15.19}$$

Here

$$\mathbf{E}\epsilon^2_{t+1} = \mathbf{E}\ (B_{t+1} + \sum_{i=1}^{p} A_{t+1}(i)a_i(b^0))^2, \tag{15.20}$$

where the optimal parameter b^0 was obtained using the data available at the day t. Variance (15.20) is minimal, if

$$y^0_{t+1} = B_{t+1} + \sum_{i=1}^{p} A_{t+1}(i)a_i(b^0), \tag{15.21}$$

because the expectation of y_{t+1} is $B_{t+1} + \sum_{i=1}^{p} A_{t+1}(i)a_i(b^0)$ under the assumptions.

Table 15.1. The average "next-day" prediction results of ARMA and RW models.

$Data$	$DeltaARMA\%$	$MeanARMA$	$VarARMA$
$/£)	- 1.779090e-01	1.293609e+00	8.454827e-02
DM/$	- 1.191e-02	1.092e-01	9.985e-02
Yen/$	- 1.086e+00	6.369e+00	6.446505e+00
Fr/$	- 3.029e-01	4.285e-01	3.395e-01
AT&T	-1.375e+00	4.554e+00	3.621e+00
Intel Co	+2.814e-01	2.052e+01	1.936e+00
Hermis Bank	-4.280e+01	2.374e+01	1.998e+01
London Stock Exchange	-5.107e-01	2.751e+02	2.346e+01
Call Center	+3.076e+01	8.453e+02	7.111e+02

3.4 Evaluation of ARMA Prediction Errors

We compare the "next-day" ARMA prediction results and a simple Random Walk (RW) model. In RW model the next day value is $y_{t+1} = y_t + \epsilon_{t+1}$ and the predicted value is equal to the conditional expectation of $y_{t+1} = y_t$. This means that the present value is the RW prediction for the next day. There are three reasons to consider RW models

- simplicity of RW,
- RW is a trivial case of ARMA where $q = 0$, $p = 1$, and $a_1 = 1$,
- RW represents unpredictable time series, because random numbers ϵ_{t+1} are independent.

Therefore, the difference of average prediction errors of ARMA and RW models can be regarded as some parameter of "ARMA-unpredictability" of the data. That means that if this difference is zero or negative, then the time series are not predictable by the ARMA model. Table 15.1 shows the difference between the mean square deviations of ARMA and RW models. The table uses four data sets

- exchange rates of DM/$, $/£, Yen/$, and Fr/$,
- closing rates of AT&T , Intel Co., and Hermis Bank stocks ,
- index of the London Stock Exchange,

- call rates of a call center.

In Table 15.1, the symbol $MeanARMA$ denotes average prediction errors of ARMA model. The symbol $VarArma$ means the variance of ARMA predictions. The symbol $DeltaARMA\%$ denotes the relation $(RW - ARMA)/RW$ in percentages. Here RW defines average errors of the Random Walk. $DeltaARMA\% > 0$ means that the ARMA model predicts better.

The data is divided into three equal parts.
The first part is to estimate parameters a and b of an ARMA model using fixed numbers p and q.
The best values of p and q are defined using the second part of data.
The third part is to compare ARMA and RW models. The table shows the comparison results.

Table 15.1 shows that the ARMA model predicts all the financial data not better than RW. However, ARMA predicts call rates thirty one percent better then RW. That is a statistically significant difference. The observed deviations between RW and ARMA models of financial data are too small for practical conclusions. Figure 15.1 shows optimal parameters b and a. Here the optimal

```
 b[0] =  -2.888616e-01
a[0] = 9.131314e+02
a[1] = 4.565372e-02
a[2] = 3.765721e+02
a[3] = 1.129722e-01
a[4] = 2.993138e+02
a[5] = 3.935438e-02
a[6] = 1.109620e+02
a[7] = 2.491347e-02
a[8] = 3.427414e+01
a[9] = -1.073026e-01
a[10] = -1.432491e+03
a[11] = 8.030343e-01
```

Figure 15.1. The optimal parameters of the ARMA model predicting call rates.

$q = 1$ and $p = 12$. The "multi-day" predictions of call rates are shown in Figure 15.2.

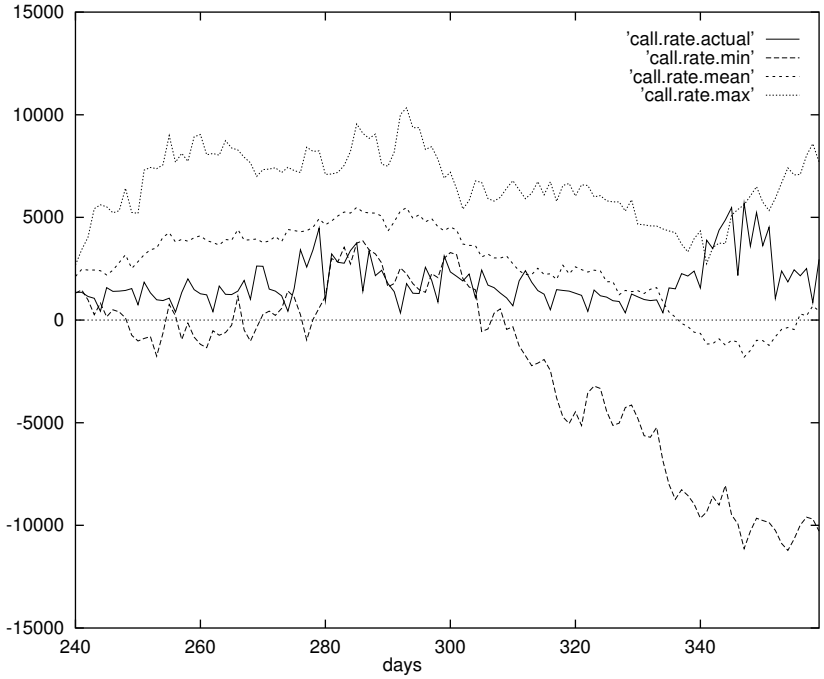


Figure 15.2. Multi-day predictions of call rates.

4. External Factors

Often we have to predict one main factor that depends on some external factors. This means that we are not interested in future values of external factors while predicting the main one. For example, we predict the stock rate as the main factor depending on the GNP as the external factor. We consider the cases when future values of all the factors are not known. Considering the cases when future values of external factors are known[3] the model should be modified accordingly.

Different symbols are used in ARMA models with external factors. The predicted value is denoted by $\nu(t)$ and the external factors are $\eta(t) = (\eta_1(t), \eta_2(t), ...)$. An influence of some external factors may be delayed.

[3]For example, while predicting call rates we know in advance such "external" factors as holidays, vacation times, e.t.c.

For illustration, consider two-dimensional case, omitting the Moving Average (MA) part. This means the two-dimensional Auto Regressive (AR) model with an external factor $\eta(t)$

$$\nu(t) = \sum_{i}^{p}(a_{1i}\nu(t-i) + a_{2i}\eta(t-d-i)) + \epsilon(t),\ p+d < t < T_0. \quad (15.22)$$

Here d is the delay parameter. First we minimize the squared deviation $\sum_t \epsilon(t)^2$ at fixed delay d and then chose the optimal delay d. The minimum of a sum (15.22), as a function of parameters a, is defined by the system of linear equations. These are obtained from the condition that all the partial derivatives are zero

$$\sum_{t=p+d+1}^{T_0-1}(\nu(t) - \sum_{i}^{p}(a_{1i}\nu(t-i) + a_{2i}\eta(t-d-i))\nu(t-i) = 0,\ i = 1, ..., p. \quad (15.23)$$

$$\sum_{t=p+d+1}^{T_0-1}(\nu(t) - \sum_{i}^{p}(a_{1i}\nu(t-i) + a_{2i}\eta(t-d-i))\eta(t-d-i) = 0,\ i = 1, ..., p. \quad (15.24)$$

We have to solve $2p$ linear equations with $2p$ variables $a_{1i}, a_{2i},\ i = 1, ..., p$ and obtain the least squares estimates $a_{1i}(d), a_{2i}(d),\ i = 1, ..., p$ at given d.

Sum (15.25), as a function of delay parameter d, is not necessarily uni-modal one. Thus to define the exact minimum

$$\nu(t) = \sum_{i}^{p}(a_{1i}(d)\nu(t-i) + a_{2i}(d)\eta(t-d-i)) + \epsilon(t), \quad p+d < t < T_0. \quad (15.25)$$

One should consider all the $K(d)$ values of the integer d. Here $K(d)$ is the number of "interesting" values of d. Therefore, expression (15.25) is an example of multi-modal function in AR models. We mentioned the delay time just to show a way how to include this important factor into a time series model. Later the delay factor is omitted, to simplify the expressions.

4.1 Missing Data

Suppose that predicting $\nu(t)$ we don't know some previous values $\nu(s),\ s < t$. Then we replace the missing data by the expected value $\hat{\nu(s)}$ defined as

$$\hat{\nu(s)} = \sum_{i}^{p}(a_{1i}\nu(s-i) + a_{2i}\eta(s-i)),\ p < t < T_0. \tag{15.26}$$

The least squares estimates of the regression parameters $a_{1i},\ a_{2i}$ are obtained using observations before the missing one and minimizing expression (15.22). The same idea can be extended, if two or more values of the factor ν are missing.

This algorithm is to "fill" the missing data only for the predicted factor ν. We replace missing values of the external factor η by the nearest previous value, because we do not predict the external factors in this model.

5. Applying ARMA to External Factors

Omitting the delay d in expression (15.22) we write

$$\nu(t) = \sum_{i}^{p}(a_{1i}\nu(t-i) + a_{2i}\eta(t-i)) + \epsilon(t),\ p < t < T_0. \tag{15.27}$$

Including this expression into the ARMA model (15.6) one obtains the following expression

$$\nu(t) = \sum_{i}^{p}(a_{1i}\nu(t-i) + a_{2i}\eta(t-i)) + \sum_{i=1}^{q} b_i\epsilon_{t-i} + \epsilon_t,\ p < t < T_0. \tag{15.28}$$

Extending expression (15.28) to M external factors

$$w^M(t) = \sum_{m=1}^{M}\sum_{i}^{p} a_i^m w^m(t-i) + \sum_{i=1}^{q} b_i\epsilon_{t-i} + \epsilon_t,\ p < t < T_0. \tag{15.29}$$

In expression (15.29), the predicted and the external factors are denoted by the same letter w with different upper indices. All the factors are represented as

one-dimensional time series of the special type

$$w(Mt) = \sum_{m=1}^{M} \sum_{i}^{p} \alpha_{Mi-(M-m)} w(M(t-i) - (M-m)) + \sum_{i=1}^{q} b_i \epsilon_{M(t-i)} + \epsilon_{Mt},\ p < t < T_0. \quad (15.30)$$

Here $\alpha_{Mi-(M-m)} = a_i^m,\ w(M(t-i) - (M-m)) = w^m(t),\ m = 1, ..., M$ and the index $mi = m * i$. Using this expression one applies the software developed for the one-dimensional ARMA model to M external factors. The data file should correspond to the expression (15.30).

6. Auto Regressive Models (AR-ABS)

The method of least squares is very sensitive to large deviations. For example, minimizing squared deviations, a large deviation of one hundred has the some influence as ten thousand small deviations, each equal to one. Therefore, replacement of squares by absolute values is beneficial in some cases.

6.1 Definitions

We define AR-ABS model as

$$w_t = \sum_{i=1}^{p} a_i w_{t-i} + \epsilon_t. \quad (15.31)$$

We assume that

$$z_{t-i} = 0,\ w_{t-i} = 0,\ \epsilon_{t-i} = 0,\ \ if\ \ t \leq i. \quad (15.32)$$

6.2 Definition of Residuals

Using equalities (15.31) we define residuals by expressions

$$\begin{aligned} \epsilon_1 &= w_1, \\ \epsilon_2 &= w_2 - a_1 w_1, \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \epsilon_t &= w_t - a_1 w_{t-1} - ... - a_p w_{t-p}. \end{aligned} \quad (15.33)$$

Next the sum

$$f(x) = \sum_{t=1}^{T} |\epsilon_t| \tag{15.34}$$

is minimized. This explains the acronym AR-ABS meaning that one minimizes a sum of absolute values of residuals.

6.3 Minimization of Residuals of AR-ABS Models

To minimize $f(x)$ we apply the Linear Programming.

$$\min_{a,u} \sum_{t=1}^{T} u_t \tag{15.35}$$

$$u_t \geq \epsilon_t,\ t = 1, ..., T \tag{15.36}$$

$$u_t \geq -\epsilon_t\ t = 1, ..., T \tag{15.37}$$

$$u_t \geq 0,\ \epsilon_t\ t = 1, ..., T \tag{15.38}$$

Here
$u = (u_t^1, u_t^2,\ t = 1, ..., T)$, $a = (a_i^1, a_i^2,\ i = 1, ..., p)$,
$u_t = u_t^1 - u_t^2,\ t = 1, ..., T)$, $a_i = a_i^1 - a_i^2,\ i = 1, ..., p$,
$u_t^k \geq 0,\ a_i^k \geq 0,\ k = 1, 2$. The advantage of AR-ABS is the lesser sensitivity to large deviations.

6.4 Applying AR-ABS to External Factors

Applying AR-ABS models to external factors is similar to that of ARMA models described in Section 5. One just omits the MA parameters b_i. Then, from (15.29) one can write

$$w^M(t) = \sum_{m=1}^{M} \sum_{i}^{p} a_i^m w^m(t-i) + \epsilon_t,\ p < t < T_0. \tag{15.39}$$

In expression (15.39), the predicted and the external factors are denoted by the same letter w with different upper indices. All the factors are represented as one-dimensional time series of the special type

$$w(Mt) = \sum_{m=1}^{M} \sum_{i}^{p} \alpha_{Mi-(M-m)} w(M(t-i) - (M-m)) + \\ +\epsilon_{Mt},\ p < t < T_0. \tag{15.40}$$

Here $\alpha_{Mi-(M-m)} = a_i^m,\ w(M(t-i)-(M-m)) = w^m(t),\ m = 1, ..., M$ and the index $mi = m * i$. Using this expression one applies the software developed for the one-dimensional AR-ABS model to M external factors. The data file should correspond to the expression (15.40).

6.5 Applying AR models to Linear Regression (LR)

AR models with external factors can be applied to estimate Linear Regression parameters, too. The difference of LR models from those of AR is that in the former case t denotes the test number. It is supposed that the observed values of the main parameter $w^M(t)$ and external parameters $w(t) = (w^m(t),\ m = 1, ..., M-1)$ at different tests t_i and t_j are independent. For example the diagnosis $w(t_i)$ of some patient t_i does not depend on the health of others patients $t \neq t_i$. In such a case the "memory" $p = 1$ and and expression (15.39) is modified this way

$$w^M(t) = \sum_{m=1}^{M-1} a^m w^m(t) + \epsilon_t, \tag{15.41}$$

where $t = 1, ..., T$ defines the test number. Here the main and the external factors are denoted by the same letter w with different upper indices where index $w = M$ means the main factor, for example a diagnosis, that is known for $t = 1, ..., T-1$ and should be predicted for $t = T$. The external factors are supposed to be known for all $t = 1, ..., T$ Here numbers $t = 1, ..., T-1$ denote known cases, number $t = T$ denotes a new case, for example a new patient just arrived.

Let us to represent the multi-factorial LR model (15.41) by one-dimensional time series of the special type

$$w(Mt) = \sum_{m=1}^{M-1} a^m w^m(t) + \epsilon_{Mt}, \tag{15.42}$$

where $Mt = M * t$. Using this expression one applies the software developed for the one-dimensional AR model to M factors of Linear Regression if data file corresponds to expression (15.42).

6.6 Simplest Illustration of AR and AR-ABS Models

Assume that

$$w(x) = ax + \epsilon \tag{15.43}$$

then the least squares estimate is

$$a(N) = \frac{\sum_{i=1}^{N} w(x_i)}{\sum_{i=1}^{N} x_i}, \tag{15.44}$$

and the ABS estimate is

$$a(N) = arg \min_{a} \sum_{i} |w(x_i) - ax_i|. \tag{15.45}$$

Condition (15.44) provides the minimum of square deviations, condition (15.45) corresponds to the minimum of absolute deviations. In the case (15.45) one minimizes a piece-wise linear function with breaking points $a_i = w(x_i)/x_i$. Therefore optimum is obtained at the point a_i that minimizes the sum of (15.45). In one-dimensional cases this can be done by comparing all values of $a_i,\ i = 1, ..., N$. In general, minimization of (15.45) is performed by linear programming (15.35).

7. Artificial Neural Networks Models (ANN)

With respect to the Moving Average parameters b, the ARMA model is non-linear (see expression (15.29)). With respect to data, this model is linear. If we are interested in the non-linearities, then we may apply many other non-linear models, including the ones that are non-linear with respect to data. In this book, we discuss two of them. Here the ANN model will be considered. In the next chapter, we introduce a bilinear term into the ARMA model.

7.1 Involving Auto Regression (AR) into ANN

We apply ANN by involving the non-linear activation function ϕ into the standard Auto-Regression (AR) model

$$w_t = \phi(\sum_{i=1}^{p} a_i w_{t-i}) + \epsilon_t. \tag{15.46}$$

The idea lurking behind ANN-AR model is that the activation function ϕ roughly represents the activation of a real neuron. We minimize the sum

$$f_m(x) = \sum_{t=1}^{T} \epsilon_t^2. \tag{15.47}$$

Here the objective $f_m(x)$ depends on l unknown parameters given as a l-dimensional vector $x = (x_k, k = 1, ..., p) = (a_i, i = 1, ..., p)$.

One see from expression (15.46) that residuals ϵ_t are non-linear functions of parameters a_t[4]. This means that the minimum conditions

$$\frac{\partial f_m(x)}{\partial a_i} = 0,\ i = 1, ..., p \tag{15.48}$$

is a system of non-linear equations with multiple solutions.

An interesting activation function is derived using the Gaussian distribution function

$$\phi(w_t(l)) = \frac{\beta}{\sqrt{2\phi}\sigma} \int_{-\infty}^{w_t(l)} e^{-\frac{w-\mu}{\sigma}} dw. \tag{15.49}$$

Here $w_t(l) = \sum_{i=1}^{l} a_i w_{t-i}$. β is a scale parameter. The function (15.49) is different from the activation of a real neuron[5] but is convenient for analysis. Here sum (15.46) depends on the parameters $\beta,\ \sigma,\ \mu$, too. These parameters are unknown, as usual. Therefore, they should be optimized. Therefore, the objective $f_m(x)$ depends on $p+3$ unknown parameters represented as a $p+3$-dimensional vector $x = (x_k, k = 1, ..., p+3) = (a_i, i = 1, ..., p, \beta, \sigma, \mu)$. That is the main difference of model (15.49) from the traditional ANN models.

In the traditional ANN models the activation functions are selected by their resemblance to the natural ones from biophysical experimentation. In this research the resemblance factor is neglected. The activation function (15.49) is considered just as a reasonable non-linearity that should be adapted to the available data. The multi-modality problems of ANN models are discussed in (Mockus et al., 1997).

8. Bilinear Models (BL)

It is well known that for the adequate description of some phenomena additional non-linear terms of the time series should be included. A simple example is a bilinear term (see (Rao and Gabr, 1984, Liu, 1989)). Here bilinear time series extend the ARMA model:

$$w_t = \sum_{i=1}^{p} a_i w_{t-i} + \sum_{i=1}^{q} b_i \epsilon_{t-i} + \sum_{i=1}^{s}\sum_{j=1}^{r} c_{ij} z_{t-i}\epsilon_{t-j} + \epsilon_t. \tag{15.50}$$

An illustrative example is in (Mockus et al., 1997).

[4]This is true, if the activation function ϕ is non-linear. The linear activation function reduces the ANN model to the standard Auto Regressive (AR)n model.

[5]The output of the traditional activation function is supposed to be non-negative.

9. Auto Regressive Fractionally Integrated Moving Average Models (ARFIMA)

9.1 Definitions

ARMA, ANN, and BL models consider stationary time series . The stationarity of a model is a simplification of reality. A well-known source of non-stationary behavior is the linear component, the trend. One eliminates the trend by differentiation, since derivatives of linear functions are constant. The elegant extension of this idea is the Auto Regressive Fractionally Integrated Moving Average model (ARFIMA).

We define an ARFIMA[6] process as the following time series z_t[7]

$$A(L)(1-L)^d z_t = B(L)\epsilon_t. \tag{15.51}$$

Here

$$A(L)w_t = w_t - \sum_{i=1}^{p} a_i w_{t-i} \tag{15.52}$$

and

$$B(L)\epsilon_t = \epsilon_t - \sum_{i=1}^{q} b_i \epsilon_{t-i}, \tag{15.53}$$

where $\epsilon_t = Gaussian\ \{0, \sigma^2\}$. We define the transformation $(1-L)^d$ as follows:

$$w_t = (1-L)^d z_t = z_t - \sum_{i=1}^{\infty} d_i z_{t-i}. \tag{15.54}$$

Here

$$d_i = \frac{\Gamma(i-d)}{\Gamma(i+1)\Gamma(-d)}, \tag{15.55}$$

where d is a fractional integration parameter, and $\Gamma(.)$ is a gamma function. We assume that

$$z_{t-i} = 0,\ w_{t-i} = 0,\ \epsilon_{t-i} = 0,\ \ if\ \ t \le i. \tag{15.56}$$

[6]Often the alternative title ARIMA(p,d,q) is used.

[7]Note that, contrary to the traditional practice, we do not assume the time series to be stationary. We allow the degree of integration of the series to be detected by direct estimate of the differentiation parameter d.

We truncate sequence (15.54)

$$d_i = 0, \;\; if \;\; i > R. \tag{15.57}$$

Here R is the truncation parameter, the number of non-zero components.

9.2 Minimization of Residuals

We define residuals by recurrent expressions:

$$\begin{aligned} \epsilon_1 &= w_1, \qquad (15.58) \\ \epsilon_2 &= w_2 - a_1 w_1 + b_1 \epsilon_1, \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \epsilon_t &= w_t - a_1 w_{t-1} - ... - a_p w_{t-p} + b_1 \epsilon_{t-1} + ... + b_q \epsilon_{t-q}. \qquad (15.59) \end{aligned}$$

Next, the sum

$$f(x) = \log f_m(x), \;\; f_m(x) = \sum_{t=1}^{T} \epsilon_t^2 \tag{15.60}$$

is minimized.

The logarithm is used to decrease the objective variation by improving the scales. The objective $f_m(x)$ depends on $m = p + q + 1$ unknown parameters. They are represented as an m-dimensional vector $x = (x_k, k = 1, ..., m) = (a_i, i = 1, ..., p, b_j, j = 1, ..., q, d)$.

It follows from (15.59), (15.54), and (15.52) that residuals ϵ_t are linear functions of the parameters a_t. This means that the minimum conditions

$$\frac{\partial f_m(x)}{\partial a_i} = 0, \;\; i = 1, ..., p \tag{15.61}$$

are given by a system of linear equations. They estimate linear parameters $a_i = a_i(b, d)$ as a function of non-linear ones $b_i, i = 1, ..., q, d$. It reduces the number of parameters of non-linear optimization to $n = q + 1$.

The system

$$\frac{\partial f_m(x)}{\partial b_i} = 0, \;\; i = 1, ..., q \tag{15.62}$$

may have a multiple solution, because the residuals ϵ_t depend on b_i as polynomials of degree $T-1$.

The equation

$$\frac{\partial f_m(x)}{\partial d} = 0 \tag{15.63}$$

may have multiple solutions, too, because the residuals depend on d as a polynomial of degree R, where R is a truncation parameter.

The objective $f_m(x)$ is a multi modal function of parameters d and $b_i, i = 1, ..., q$ [8]. Therefore, one uses methods of global optimization (see, (Mockus et al., 1997)). Denote

$$f(x) = \log f_{q+1}(x). \tag{15.64}$$

Here

$$f_{q+1}(x) = f_m(x),\ x_j = b_j, j = 1, ..., q,\ x_{j+1} = d,$$
$$x_{j+1+i} = a_i(b, d),\ i = 1, ..., p.$$

This means that condition (15.61) defines those x- components that represent parameters $a_i, i = 1, ..., p$.

There is no variance σ^2 in expressions (15.64) and (15.59). If necessary, we have to estimate the variance by another well-known technique.

9.3 Discussions

Table 15.2 shows results obtained using the ARFIMA model (Mockus et al., 1997). Parameters b and d were estimated using daily exchange rates of \$/\$, and DM/\$, and closing rates of AT&T and Intel Co shares. Table 15.2 shows that d's are very close to zero (see also Figures 15.3). An exception is Intel Co shares.

[8] The same reasoning applies to the log-likelihood function, too.

Table 15.2. Estimated parameters b and d of ARFIMA models.

$Data$	b_0	b_1	d	$\min \log f(x)$
\$/£	-1.195	-0.169	0.0005	1.51675
DM/\$	-1.019	0.0120	0.0007	1.60065
AT&T	-1.017	0.0118	0.00005	9.83208
Intel Co	0.9975	0.0055	0.012	7.35681

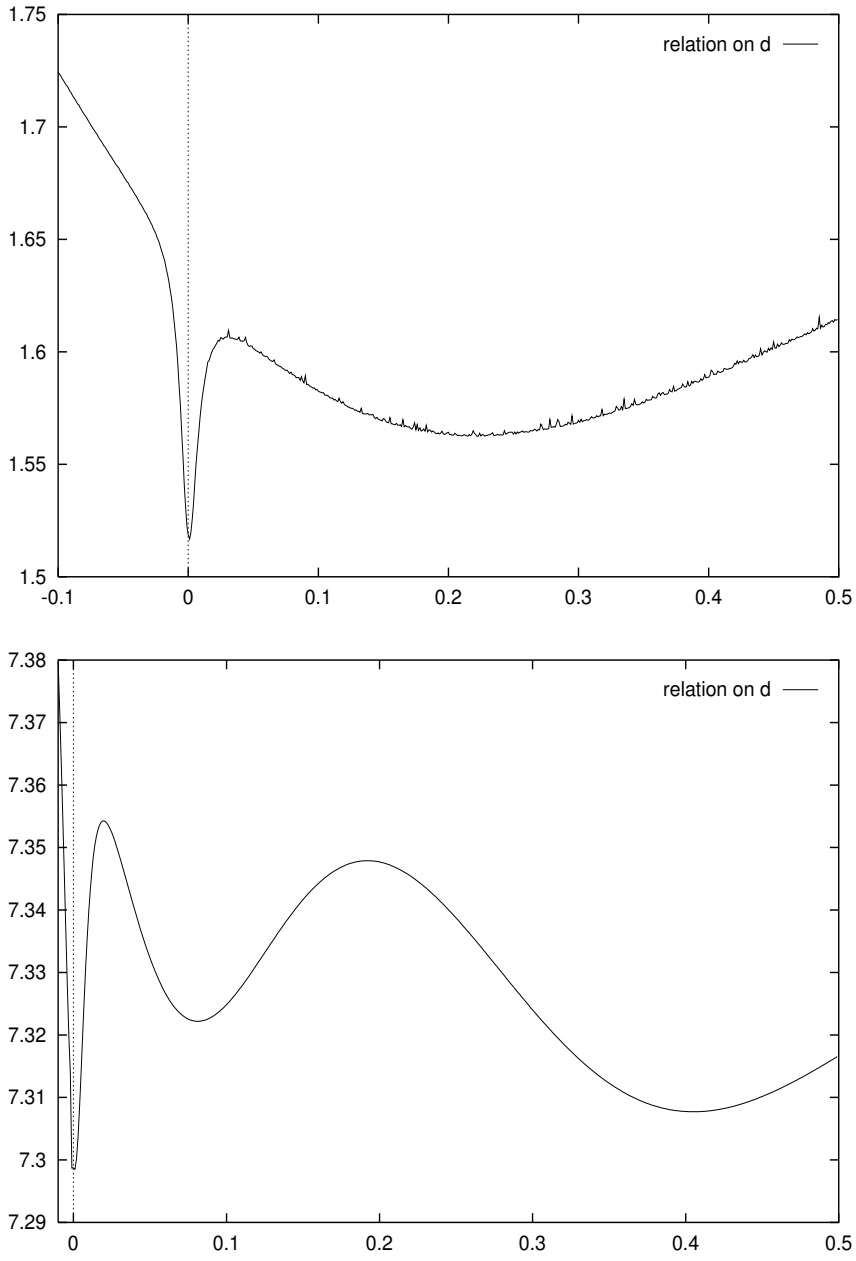


Figure 15.3. Log-sum (15.60) as a function of the parameter $d \in [-0.01, 0.5]$. Top figure shows \$/£ exchange rates. Bottom one shows Intel Co stocks closing rates.

Hence, following the traditional approaches (Cheung, 1993, Yin-Wong and Lai., 1993, Cheung and Lai, 1993) one concludes that the underlying stochastic

processes generating exchange rates do not exhibit persistence and are stationary. That contradicts the visual impression of the corresponding data (see Figures 15.4 and 15.10).

This apparent contradiction may be resolved by dropping the assumption that the parameters a, b, d of the ARFIMA model remain constant. This assumption is common for most of the traditional methods. An alternative is the structural stabilization model described in Chapter 11.

10. Multi-Step Predictions

Often one wants to predict for several days (hours, weeks, months, e.t.c) ahead. This is the Multi-Step Predictions (MSP). One can do that by using a Monte Carlo simulation. The residuals ϵ_t (see expression (15.59)) are determined up to the simulation starting moment $t(s)$ using the observed data. The rest of residuals $\epsilon_t,\ t \geq t(s)$ are generated by a Gaussian distribution with zero mean and variance σ^2. The simulation is repeated K times. Considering external factors one just replaces them by the nearest previous values (see Section 4.1).

The illustration is in Figure 15.2. The line $call.rate.actual$ shows the observed call rate. Lines
$call.rate.min$, $call.rate.mean$, and $call.rate.max$ show the minimal, the average, and the maximal results of MSP predictions. The "min" and "max" lines denote the lower and the upper values of simulation. Therefore, these lines are called "MSP- confidence intervals," meaning that if the model is true, one may expect those "intervals" to cover the real data with some "MSP-confidence level" $\alpha(MSP)$.

It is difficult to define $\alpha(MSP)$ exactly. If "interval deviations" may be considered as independent and uniformly distributed random variables, we obtain $\alpha(MSP) = 1 - K$. Here K is the number of Monte-Carlo repetitions. In the example, $K = 10$, thus $\alpha(MSP) = 0.9$.

This assumption over-simplifies the statistical model. Therefore, the "MSP-confidence level" $\alpha(MSP)$ is just a Monte Carlo approximation.

11. Structural Stabilization

11.1 Stabilization of Structures of Time Series

The traditional time series models assume that their parameters do not change. Examples are parameters a, b of the ARMA model, parameters a, b, d of the ARFIMA model, parameters a, b, c of the BL model, and parameters a, β, σ, μ of the ANN model. Natural changes of economical conditions introduce a variability of parameters of models describing the financial data, such as stock rates or currency exchange rates. The variability remains in the stable economical conditions, too. The reason is the "Feed-Back" processes. It is well known that the stock rate predictions may influence the supply and demand and consequently the future rates. In such cases the statistical best fit models should be complemented by the game-theoretical equilibrium models. There is the possibility that the equilibrium model would be namely the Wiener process. That means that market rates in economics are governed by models similar to the Brownian motion. In such a case market rates would be as unpredictable as movements of individual molecules in gases. This is a working hypothesis supported by available data. Table 15.1 shows that the predictions of the financial data by the simple Wiener process, called as the Random Walk (RW) model, are as good as that of the sophisticated ARMA models.

The objective of traditional time series models is to define such parameters that minimize a deviation from the available data. One may call them as the best fit models. The goodness of fit is described by continuous parameters C called as state variables. For example, in the ARMA model (see expression (15.1)) the state variables are
$C = (a_i,\ i = 1, ..., p,\ b_j,\ j = 1, ..., q)$.

If the parameters remain constant, then models that fit best to the past data will predict the future data as well. Otherwise, the best fit to the past data can be irrelevant or even harmful for predictions. Therefore, one needs models which are not sensitive to the changes of parameters. Such models may predict the uncertain future better by eliminating the nuisance parts from the structure of the model.

Trying to solve this problem, one introduces a notion of the model structure. The model structure is determined by the Boolean parameters S called as structural variables. A structural variable is equal to unit, if the corresponding component of time series model is included. Otherwise, the structural variable is equal to zero.

For example, in the ARMA model $S = (s_i^a,\ i = 1, ..., p,\ s_j^b,\ j = 1, ..., q)$. Here $s_i^a = 1$, if the parameter a_i is included into the ARMA model. Otherwise, $s_I^a = 0$ [9]. We search for such structure S of the model that minimizes the prediction errors in the changing environment. To achieve this we divide available data $W = (w_t,\ t = 1, ..., T)$ into two parts $W_0 = (w_t,\ t = 1, ...T_0)$ and $W_1 = (w_t\ t = T_0 + 1, ..., T)$.

The first part W_0 is to estimate continuous parameters $C = C(S)$ that depends on Boolean structural parameters S. The estimates are obtained for a set of all feasible S by minimizing the least square deviation using data W_0.

The second part W_1 is used to select such S that minimize the least square deviation. This means that the second part W_1 is to estimate Boolean structural parameters.

Denote by $R_t(S, C, W)$ the predicted value of a model R with fixed parameters S, C using the data $(w_1, ..., w_{t-1}) \subset W$. The difference between the prediction and the actual data w_t is denoted by $\epsilon_t(S, C, W) = w_t - R_t(S, C, W)$. Denote by $C_0(S)$ the fitting parameters C which minimize the sum of squared deviations $\Delta_{0,0}(C, S)$ using the first data set W_0 at fixed structure parameters S.

$$\Delta_{0,0}(C, S) = \sum_{t=1}^{T_0} \epsilon_t^2(S, C, W), \tag{15.65}$$

$$C_0(S) = \arg\min_C \Delta_{0,0}(C, S). \tag{15.66}$$

We stabilize the structure S by minimizing the sum of squared deviations $\Delta_{1,0}(S)$ using the second data set W_1 and the fitting parameters $C_0(S)$ that were obtained from the first data set

$$\Delta_{1,0}(S) = \sum_{t=T_0+1}^{T} \epsilon_t^2(S, C_0(S), W), \tag{15.67}$$

$$S_1 = \arg\min_S \Delta_{1,0}(S). \tag{15.68}$$

This is a way to reach a tradeoff between the fitting parameters and the structural ones. The fitting parameters $C_0(S)$ provide the best fit to the first data set W_0 at fixed structure S. One stabilizes the structure S by minimizing the prediction errors for the second set of data W_1 using the fitting parameters $C_0(S)$. Here

[9] Note that describing a real data one needs diverse structures including a number of different models, not just one specific model as in this illustrative example.

stabilization is achieved because the fitting parameters are defined by the first set of data W_0. The stabilized structure $S = S_1$ of R is obtained by eliminating unstable[10] parameters and parts of the time series.

We consider two data sets W_0 and W_1 just for simplicity. One may partition the data W into many data subsets $w_t \in W_k,\ t \in T_k,\ k = 1, ..., K,\ \cup_k W_k = W,\ \cup_k T_k = T$. In this case we minimize the sum

$$S_K = \arg\min_S \Delta(S), \tag{15.69}$$

where

$$\Delta(S) = \sum_{k,l} \Delta_{k,l}(S). \tag{15.70}$$

Here

$$\Delta_{k,l}(S) = \sum_{t \in T_k} \epsilon_t^2(S, C_l(S), W). \tag{15.71}$$

Note, that dividing the data W into many parts one may obtain sequences T_k too short for the meaningful estimate of parameters C_k, if T is not large. If T is large, one may expect that most of the fitting parameters C_k would be different in different data subsets W_k. Thus, they will be eliminated by the stabilization procedure (15.69). Therefore, $K = 2$ seems a reasonable number, at the first stabilization attempt. One may try $K > 2$ later.

The idea of the structural stabilization follows from the following observation. The best estimate of time series parameters, using a part W_0 of the data $W = W_0 \cup W_1$, is optimal for another part W_1 only if all the parameters remain the same. Otherwise, one may obtain a better estimate by elimination of the changing parameters from the model. For example, in the case of changing parameters $(a_i,\ i = 2, ..., p,\ b_j,\ j = 1, ..., q)$ of the ARMA model, the best prediction is obtained by elimination of all these parameters, except $a_1 = 1$ (see Table 15.1).

11.2 Simple Example

Consider, for illustration, a simple example.

$$w_t = a_1(t)w_{t-1} + a_2(t)w_{t-2} + \epsilon_t,\ t = 3, 4, 5. \tag{15.72}$$

[10]The parameters are considered as unstable if they change too much in different data sets.

The observed values are $w_1 = -1,\ w_2 = w_3 = 1,\ w_4 = 2,\ w_5 = 1$. The first data set $W_0 = w_i, ..., w_4$ is to estimate continuous parameters $C = (a_1, a_2)$. The second data set $W_1 = w_5$ is to estimate Boolean parameters $S = (s_1^a, s_2^a),\ s_i^a = 0, 1$. Equality to zero $s_i^a = 0$ suggests the elimination of the continuous parameter a_i. There are three feasible S, namely $S_1 = (0, 1),\ S_2 = (1, 0),\ S_3 = (1, 1)$[11].

Assume that unknown parameters depend on t this way

$$a_1(t) = \begin{cases} 2, & \text{if } t = 3, \\ 1, & \text{if } t = 4, \\ 0, & \text{if } t = 5, \end{cases} \tag{15.73}$$

$$a_2(t) = 1,$$
$$\epsilon(t) = 0.$$

In the case $S = S_3$, the least square estimates are $a_1(S_3) = 1.5$ and $a_2(S_3) = 0.5$. The prediction is $w_5(S_3) = 3.5$.
If $S = S_1$, the least squares estimate of the only remaining parameter is $a_2(S_1) = 0.5$. The prediction is $w_5(S_1) = 0.5$.
In the case $S = S_2$, the least square estimate is $a_1(S_2) = 1.5$ and the prediction is $w_5(S_2) = 3.0$.

The best prediction $w_5(S_1) = 0.5$ is provided by the structure $S_1 = (0, 1)$. The reason is obvious: this structure eliminates the highly unstable parameter a_1. Applying the structural stabilization one eliminates the nuisance parameter $a_1(t)$ and simplifies AR model (15.72)

$$w_t = a_2(t)w_{t-2} + \epsilon_t,\ t = 3, 4, 5. \tag{15.74}$$

Note, that we eliminate the larger parameter a_1 ($a_1(S_2) = 1.5 > a_2(S_2) = 0.5$) because it changes (see expression (15.73)).

11.3 Examples of Structural Optimization with External Factors

Various examples of structural optimization are described in (Mockus, 1997). Here the structural optimization is considered in time series models with external factors.

[11]The elimination of both continuous parameters a_1 and a_2 is unfeasible.

If predictions depend on several factors, the multi-dimensional ARMA should be used. Denote by $\nu(Mt),\ t \geq p/M$ the main statistical component and by $\nu(Mt-i),\ i=1,..,p-1$ the external factors. One extends the traditional ARMA model this way

$$\nu(Mt) = \sum_{i=0}^{p-1} a_i \nu(Mt-i-1) + \sum_{j=0}^{q-1} b_j \epsilon(M(t-j-1)) + \epsilon(Mt), \quad (15.75)$$
$$p/M \leq t \leq T0$$

Here the number of the Auto Regressive (AR) components is denoted by p and the number of the Moving Average (MA) ones is denoted by q. The continuous variables a and b define the state of the ARMA model. We call them the state variables. The discrete parameters p, q, and $t0$ define the structure. One calls them the structural variables. The structural variable $t0$ defines the time when one starts scanning the time series for the optimization of state variables a, b. Denote by $T0$ the scanning end.

One minimizes the squared deviation at fixed structural variables $p, q, t0$

$$\Delta_{00}(p,q,t0) = \sum_{t=t0}^{T0} \epsilon(Mt)^2,\ t0 \geq p/M. \quad (15.76)$$

Here $T0 < T1 < T$ and $\epsilon(Mt)^2$ is from expression (15.75). Denote the optimal values $a = a(p,q,t0)$ and $b = b(p,q,t0)$.

If b is fixed, the optimal values of $a = a^b$ are defined by a system of linear equations. These equations follows from the condition that all the partial derivatives of sum (15.75) are equal to zero $\partial\Delta_{00}/\partial a_i = 0,\ i = 0,...,p-1$. Therefore, to obtain the least squares estimates $a_i = a_i^b,\ i = 0,...,p-1$ at given b one solves p linear equations with p variables $a_i,\ i = 0,...,p-1$ (see Chapter 3 for details).

To optimize discrete structural variables $p, q, t0$ one uses another data set. It starts at $T0$ and ends at $T1$. During optimization of structural variables one keeps the best fitting values of state variables $a = a(p,b,t0)$ and $b = b(p,q,t0)$. These values are obtained by minimization of the sum $\Delta_{01}(p,q,t0)$. The data is from $t0$ to $T0$. Here the sum

$$\Delta_{01}(p,q,t0) = \sum_{t=T0}^{T1} \epsilon(Mt)^2. \quad (15.77)$$

12. Examples of Squared Residuals Minimization

12.1 Multi-Modality Examples

Consider these examples:

- exchange rates of \$/£, DM/\$, yen/\$, and franc/\$,
- closing rates of stocks of AT&T, Intel Corporation, and Hermis bank[12],
- London stock exchange index,
- daily call rates of a call center.

The following figures show the data and optimization results. The optimization results show how least square deviations depend on parameters b of ARMA models. The $3D$ optimization results are in two forms: as surfaces and as contours.

Figures 15.4, 15.5, and 15.6 consider exchange rates of \$/£ and DM/\$.
Figures 15.7, 15.8, and 15.9 consider exchange rates of yen/\$ and franc/\$.
Figures 15.10 reflect closing rates of AT&T (top) and Intel Co.(bottom) stocks.
Figures 15.13 , 15.14, and 15.15 consider the London stock exchange index.
Figures 15.16, 15.17, and 15.17 shows stock rates of the Hermis bank and optimization results.
Figures 15.18 and 15.19 show the daily call rates of a call center and illustrate optimization results.

[12] A small Lithuanian bank.

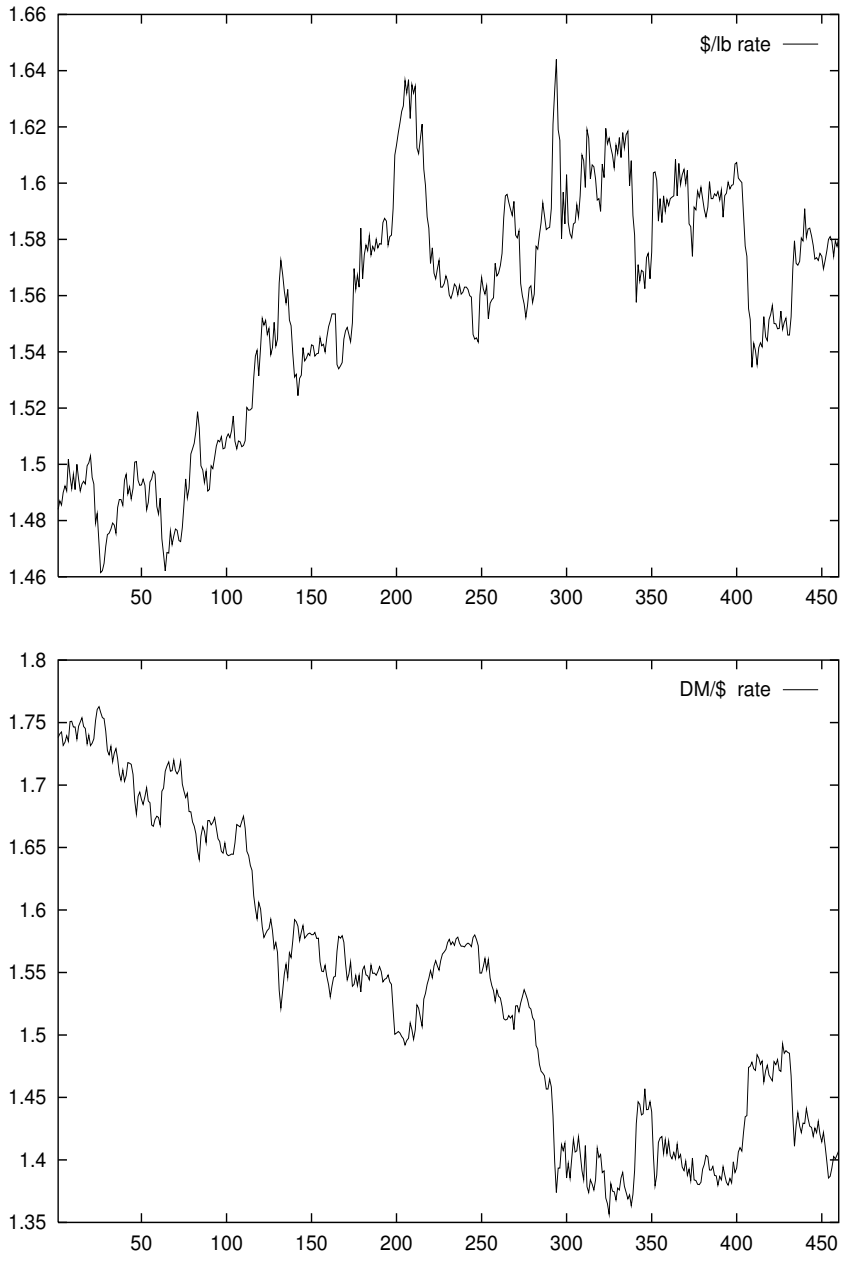


Figure 15.4. Daily exchange rate of \$/\$(top figure) and DM/\$ (bottom figure) starting from September 13, 1993.

To estimate unknown ARMA parameters we minimize a log-sum of squared residuals defined by expression (15.60). Estimates of parameters a are from

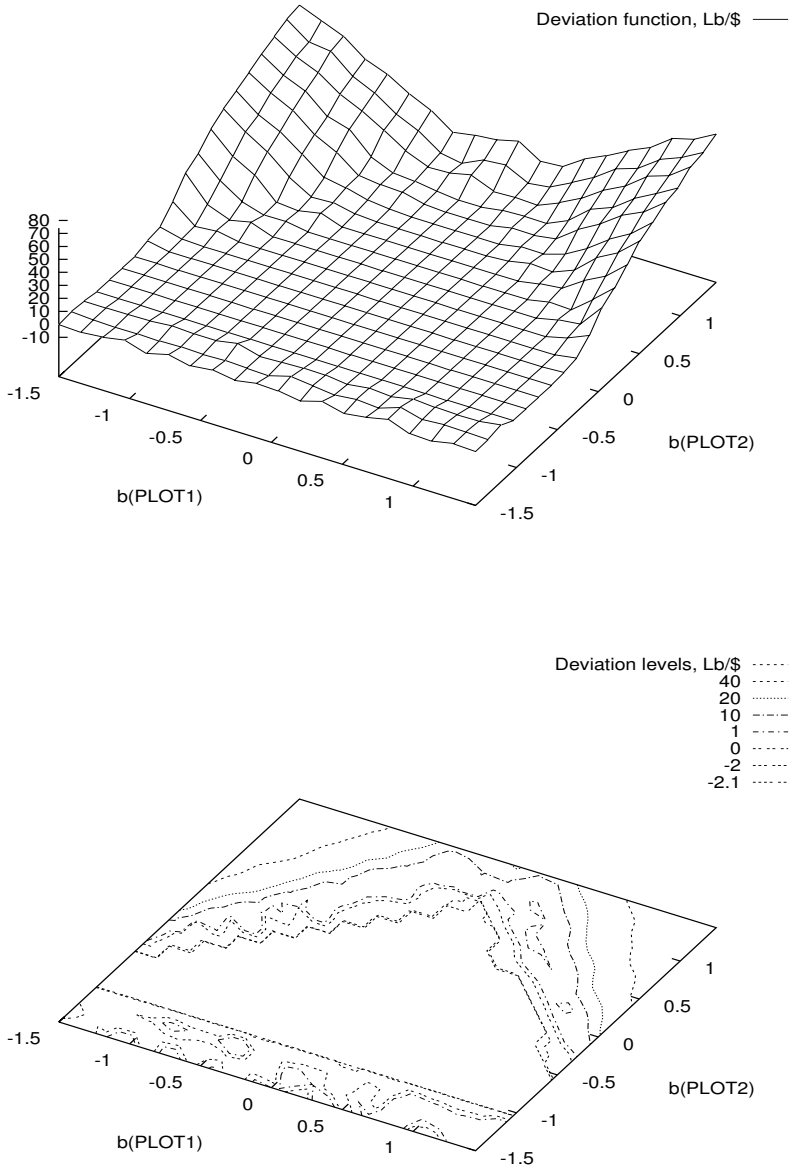


Figure 15.5. Exchange rates of \$/\$: surface f depending on parameters $b_0,\ b_1$ (top), contours of f depending on parameters $b_0,\ b_1$ (bottom).

expression (15.13). In most of the cases, figures show the multi-modality of log-sum (15.60), as a function of parameters b_0, b_1. Areas in vicinity of the

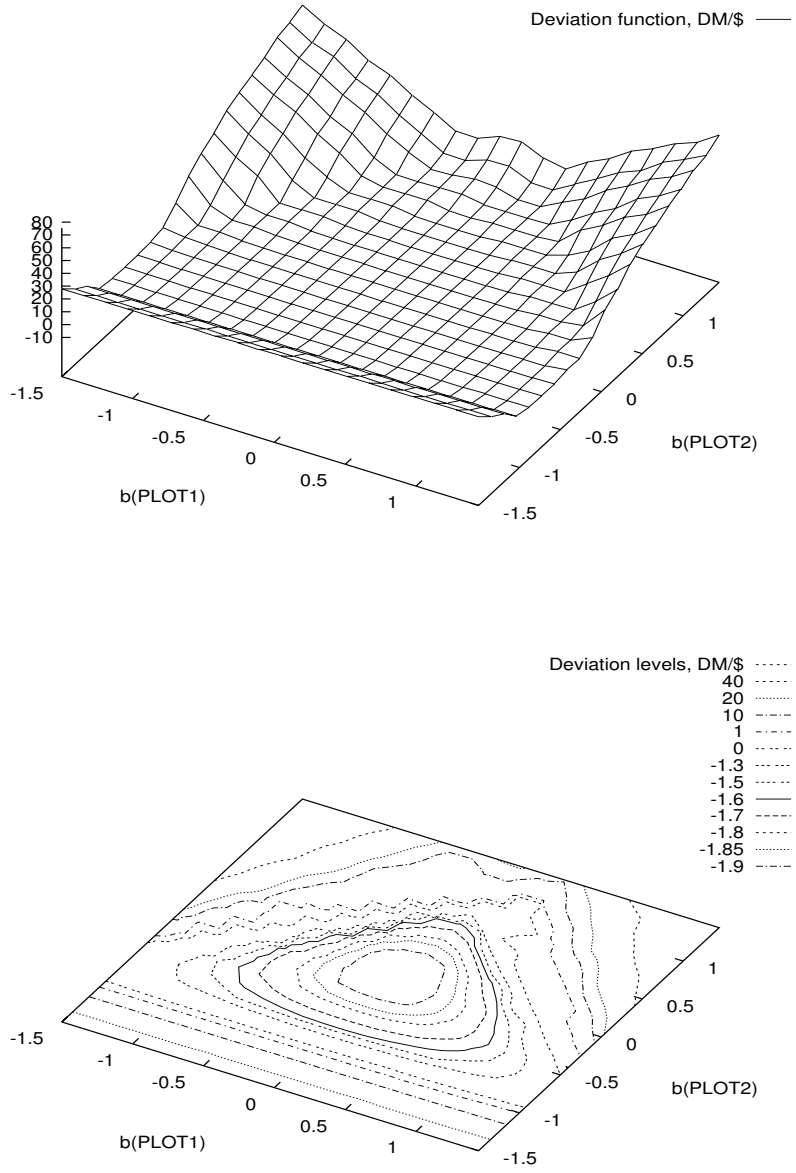


Figure 15.6. Exchange rates of DM/$: surface of f depending on parameters $b_0,\ b_1$ (top), contours of f depending on parameters $b_0,\ b_1$ (bottom).

global minima, often appear flat. A reason is that differences between values

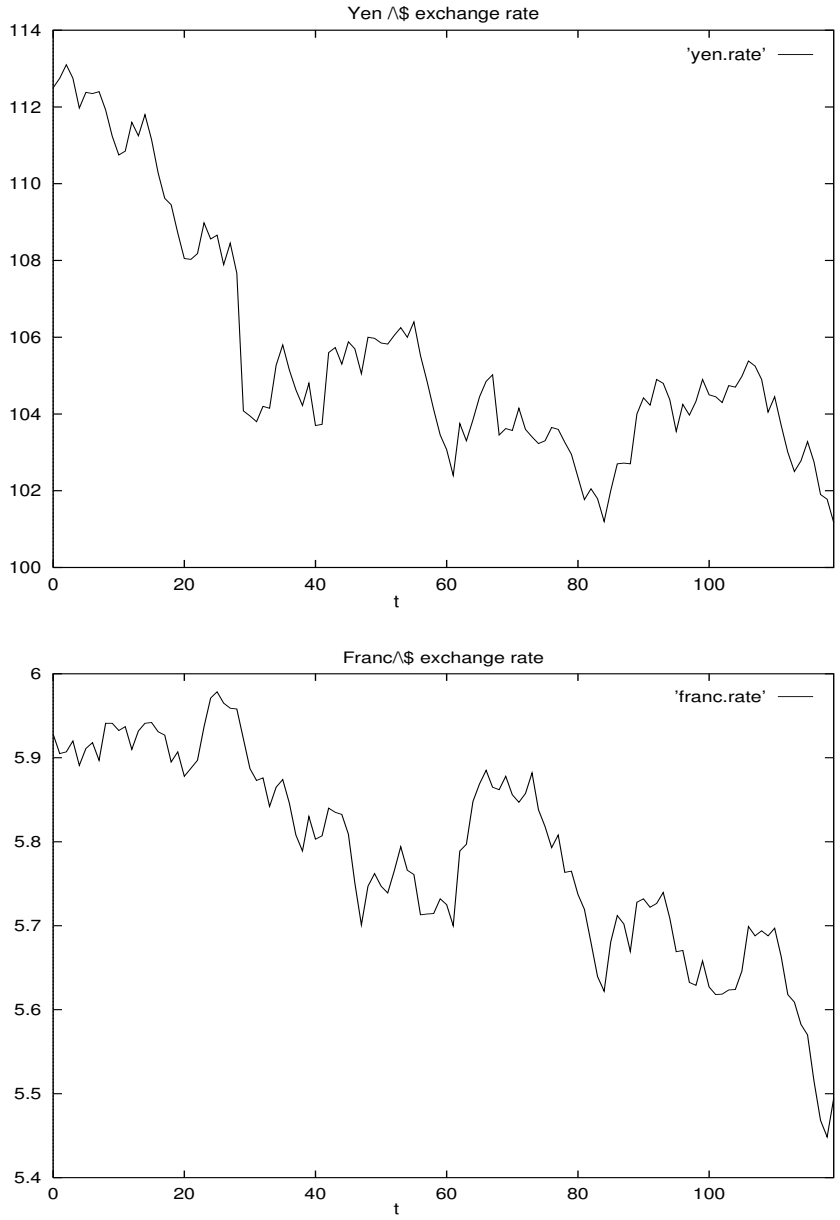


Figure 15.7. Exchange rates: yen/\$ (top), franc/\$ (bottom).

of the deviation function f in an area around the global minimum are smaller as compared with these outside this area (see Figures 15.15).

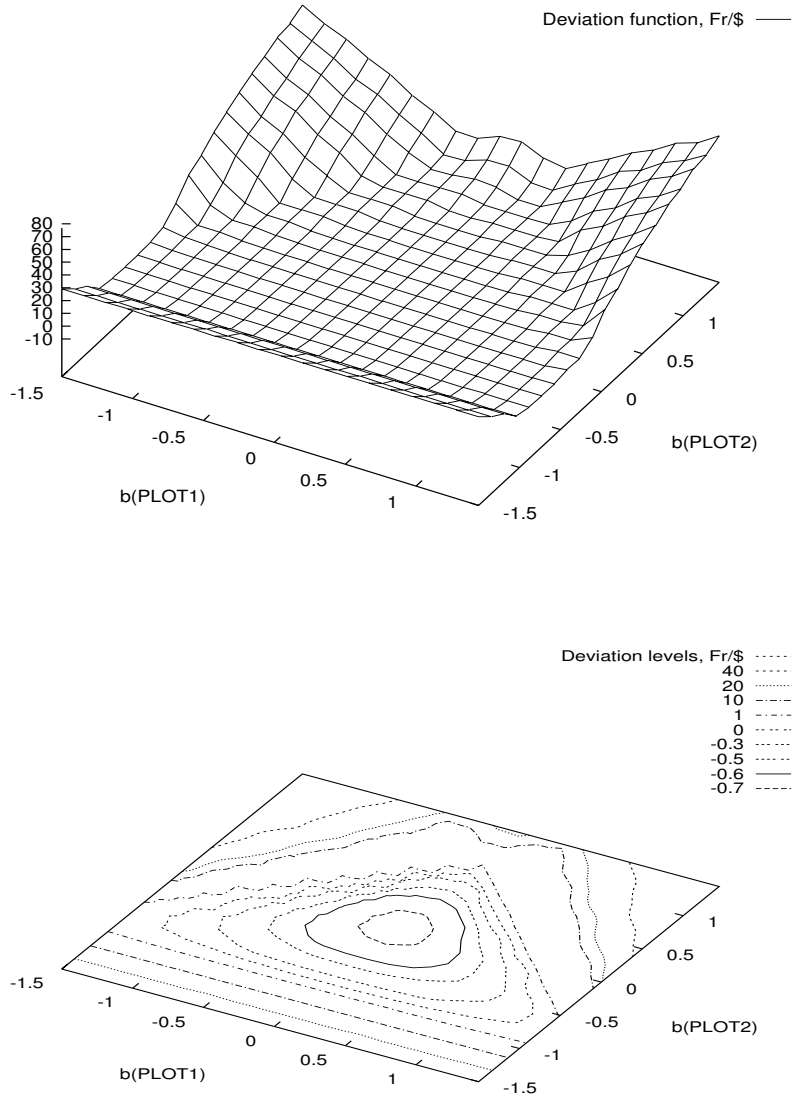


Figure 15.8. Exchange rates of fr/$, surface f depending on parameters $b_0,\ b_1$ (top), contours of f depending on parameters $b_0,\ b_1$ (bottom).

12.2 Optimization Results

The ARMA model optimization results are the points b and a (see Figure 15.1). These results were defined using a sequence of two global methods

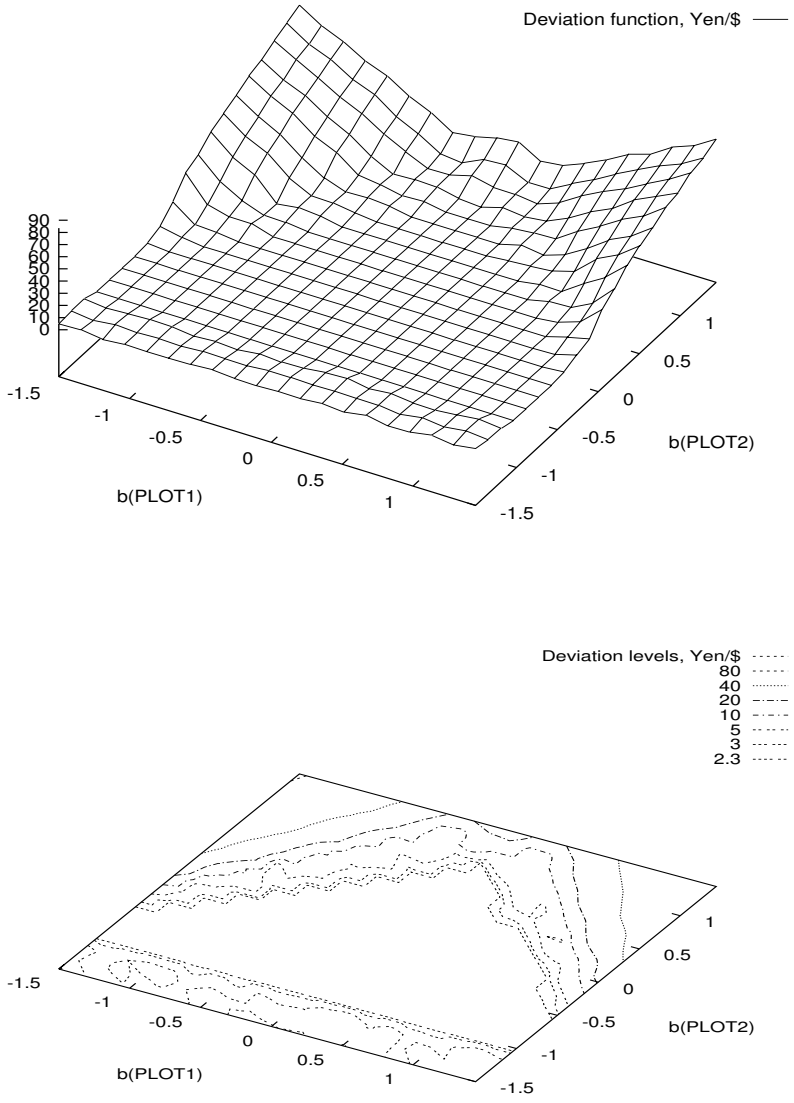


Figure 15.9. Exchange rates of yen/$, surface f depending on parameters $b_0,\ b_1$ (top), contours of f depending on parameters $b_0,\ b_1$ (bottom).

called $BAYES1$ and $EXKOR$ (Mockus et al., 1997). $BAYES1$ denotes a search by a multi-dimensional Bayesian model (Mockus et al., 1997). The best result obtained by $BAYES1$ after fifty iterations is a starting point for local

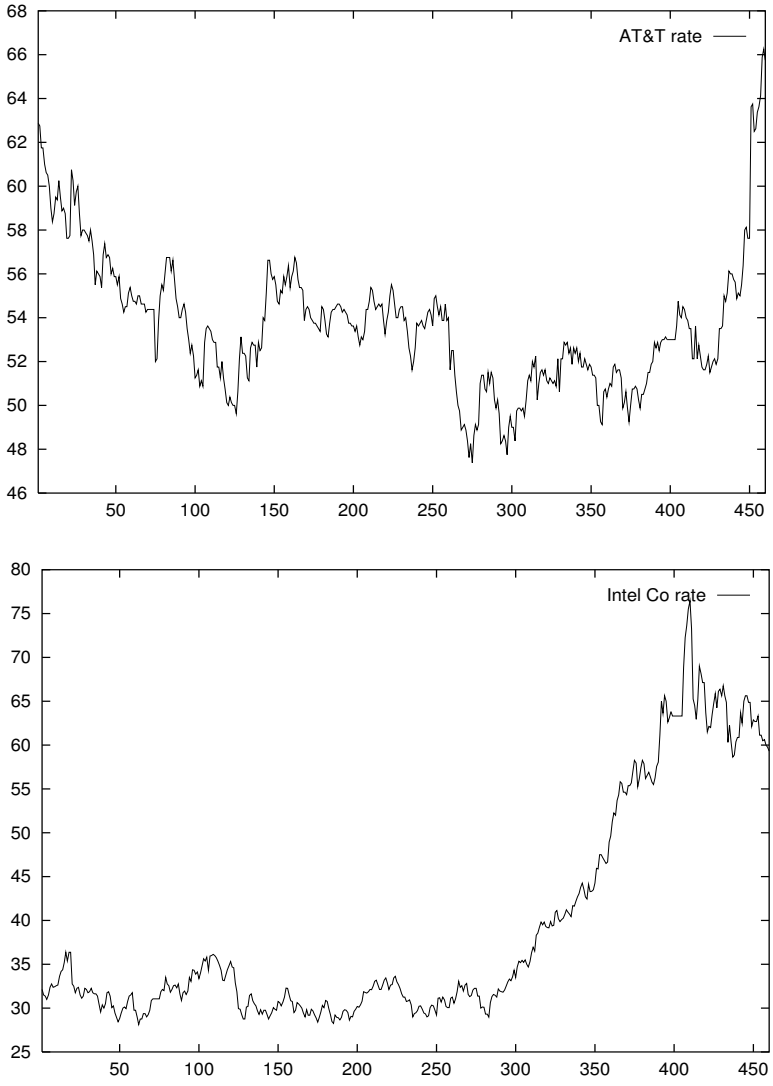


Figure 15.10. AT&T (top) and Intel Co.(bottom) stocks closing rates starting from August 30, 1993.

optimization. The local optimization is by one-dimensional coordinate search $EXKOR$ (Mockus et al., 1997), using sixty iterations.

The maximal number of Auto Regressive (AR) parameters p was $10 * M$. Here $M = 1$, if no external factors are involved. The optimal number p is defined by structural stabilization (see Chapter 11). Only two Moving Average (MA) parameters $q = 2$ were considered while plotting surfaces and contours.

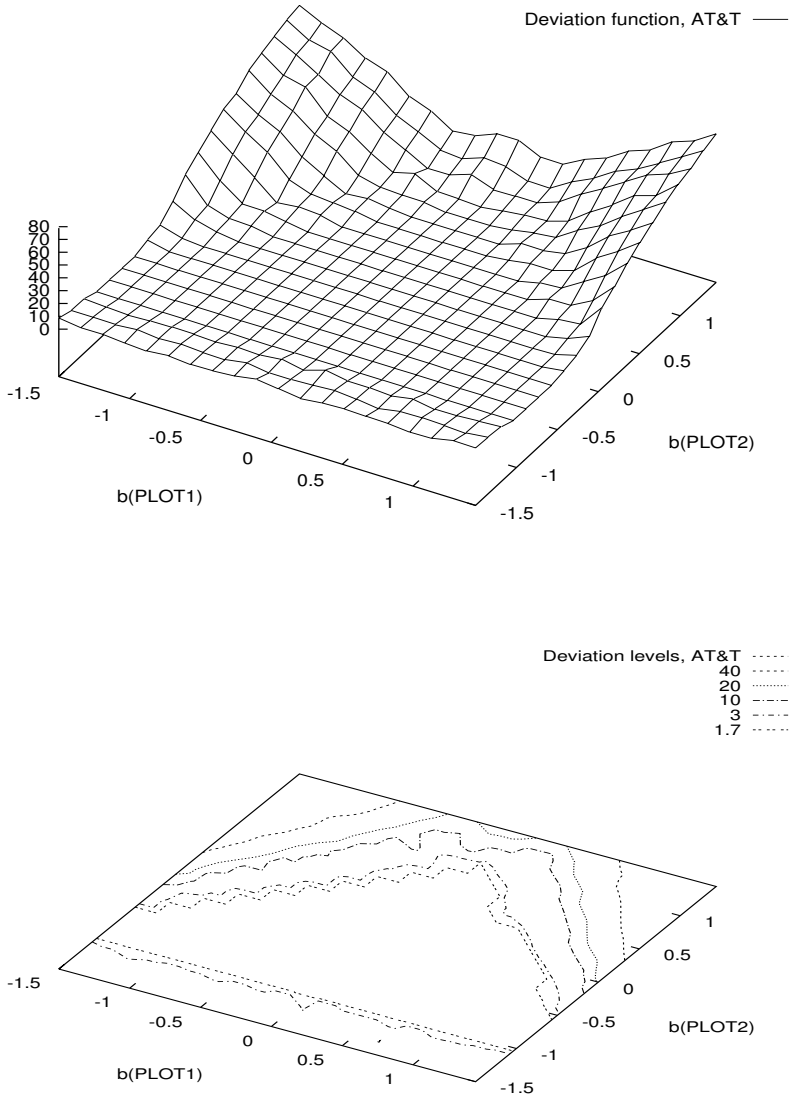


Figure 15.11. Closing rates of AT&T stocks: surface f depending on parameters $b_0,\ b_1$ (top), contours of f depending on parameters $b_0,\ b_1$ (bottom).

The results of Table 15.1 were obtained by optimization of both structural variables p and q.

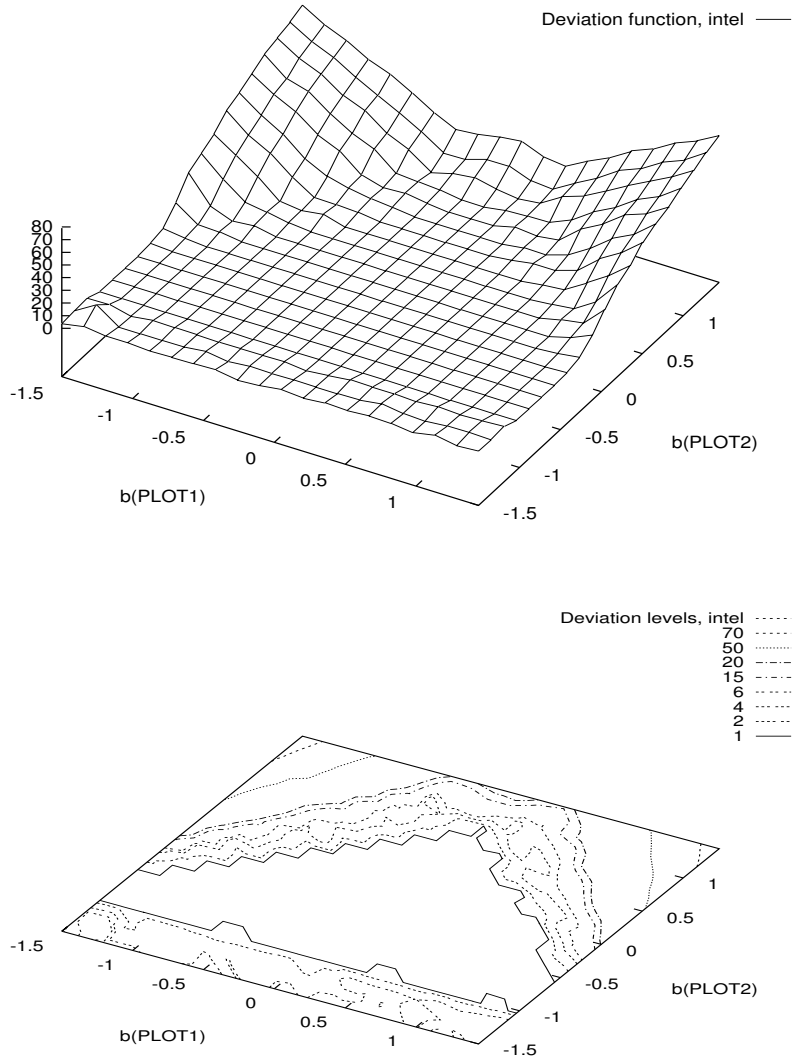


Figure 15.12. Closing rates of Intel Co stocks: surface of f depending on parameters $b_0,\ b_1$ (top) contours of f depending on parameters $b_0,\ b_1$ (bottom).

The objective of this part of research is to show a multi-modality arising in the prediction problem. Therefore, to save the computing time, the global optimization is carried out approximately, using not many iterations. The results of global optimization are starting points for local optimization. The reason is that the squared deviation as a function of parameters b becomes uni-modal

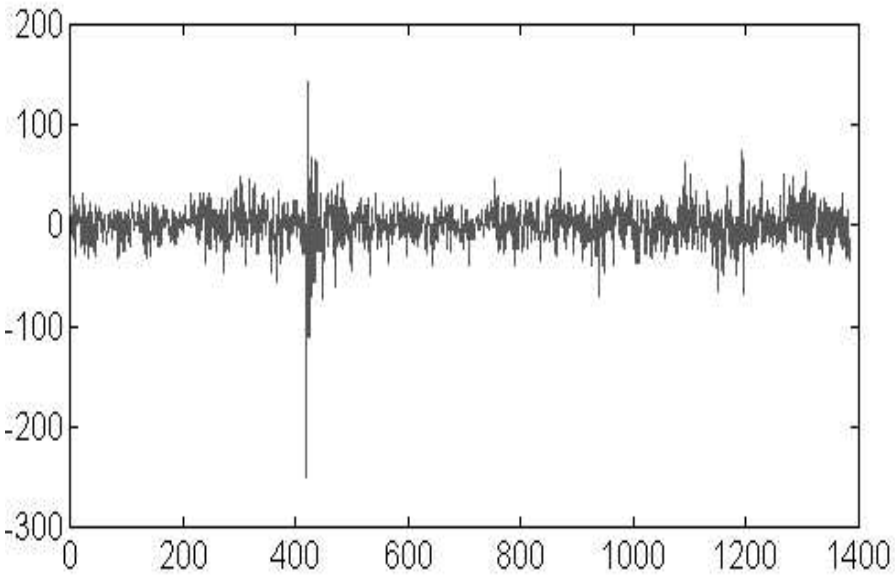


Figure 15.13. London stock exchange index.

near the global minimum (see Figures 15.15). Therefore, the results are at least as good as those obtained by the traditional local optimization.

The high-accuracy global optimization is very expensive. In the global optimization, the computing time is an exponential function of accuracy m, in the sense that $\epsilon \leq 2^{-m}$. Therefore, the problem of future investigations is how to balance computing expenses and accuracy of estimates. This task is important in both the time series prediction and the global optimization.

The investigation of multi-modality of squared deviation and variability of the parameters is the natural first step. The multi- modality is involved in non-linear regression models, including the ARFIMA ones[13].

[13]Meaning that the sum (15.60) of the ARFIMA model is a non-linear function of the parameters *b* and *d*.

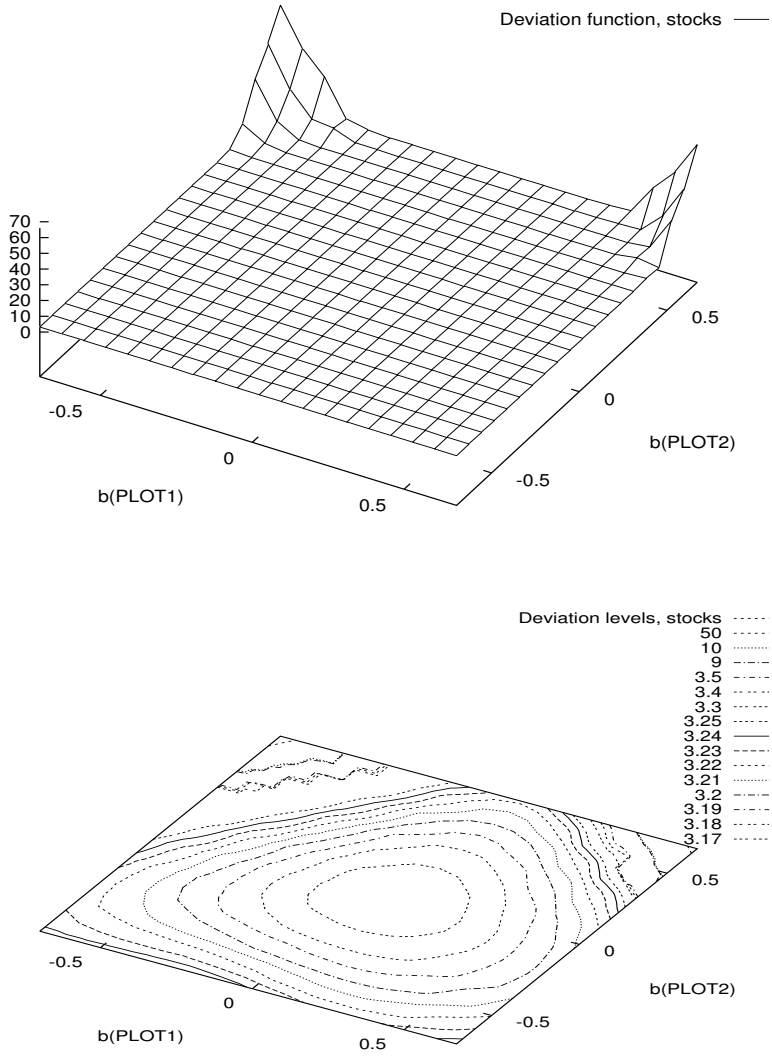


Figure 15.14. London stock exchange index, surface of f depending on parameters $b_0,\ b_1$ (top), contours of f depending on parameters $b_0,\ b_1$ (bottom).

13. Software Examples

13.1 C Version of ARMA Software (ARMAC)

The call rates depend on several factors what is usual in statistical prediction problems. That is the main difference from the traditional ARMA and ARFIMA

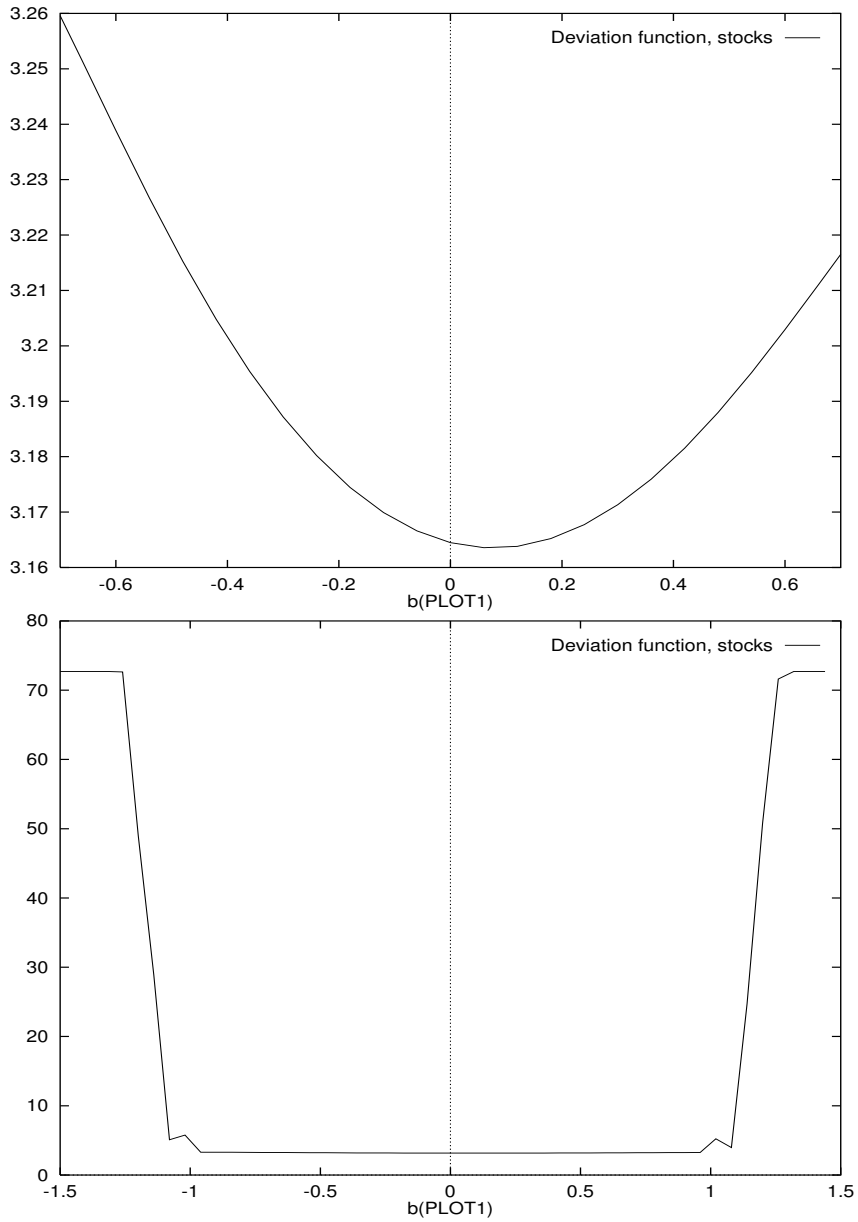


Figure 15.15. London stock exchange index: deviation f depending on parameter b_0, unimodal part (top), multi-modal part (bottom).

software for the exchange rate prediction (Mockus et al., 1997). The ARFIMA software and applications are described in (Mockus et al., 1997).

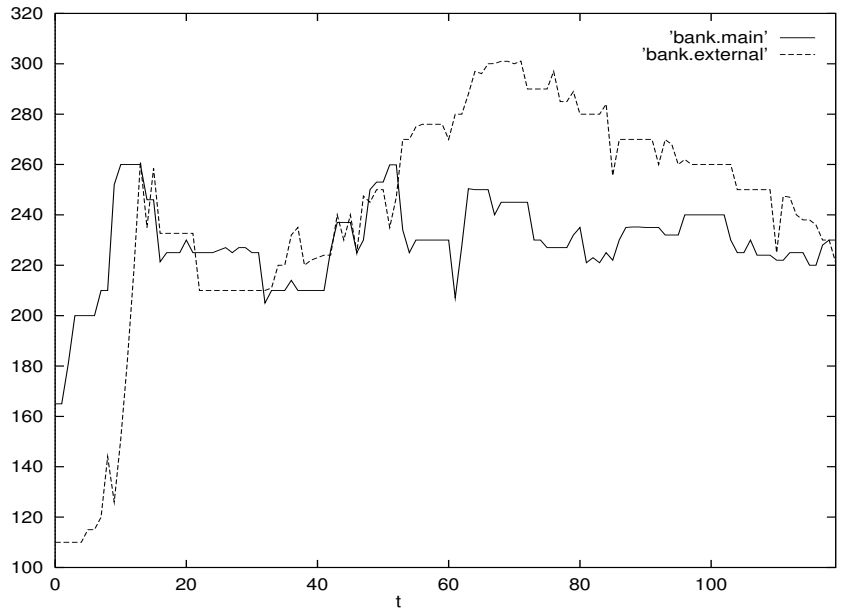


Figure 15.16. Closing rates of stocks of the Hermis bank (private) and the Lithuanian Savings bank (state owned).

We consider a version of an extended ARMA model that includes external factors (see expressions 15.28 and 15.29). The programs are in the files 'main.C', 'fi.C', and 'fitimeb.h' on the web-site (see Section 4). The first version of ARMA software is designed for data sets with no future data. That means that no future factors are not known. Therefore, the external factors are treated as missing data. It is assumed that future values of external factors are equal to the last ones. In the ARMA software under development, this is considered as the default case. In the new software version, the known future values of external factors will replace the default ones.

The results of a simple test are in the files 'test.out' and 'test.progn.out'. The data is in the file 'test.data', the initiation file is 'test.ini'. The results of the call rate example are in the files 'call.out' and 'call.progn.out'. The data is in the file 'call.data'. The initiation file is 'call2.ini'. The names of the data files are referred as INP in the initiation file INIFILE. The software is compiled and run this way:

```
Compile by 'make onestep',
Run by 'onestep > results.out'
```

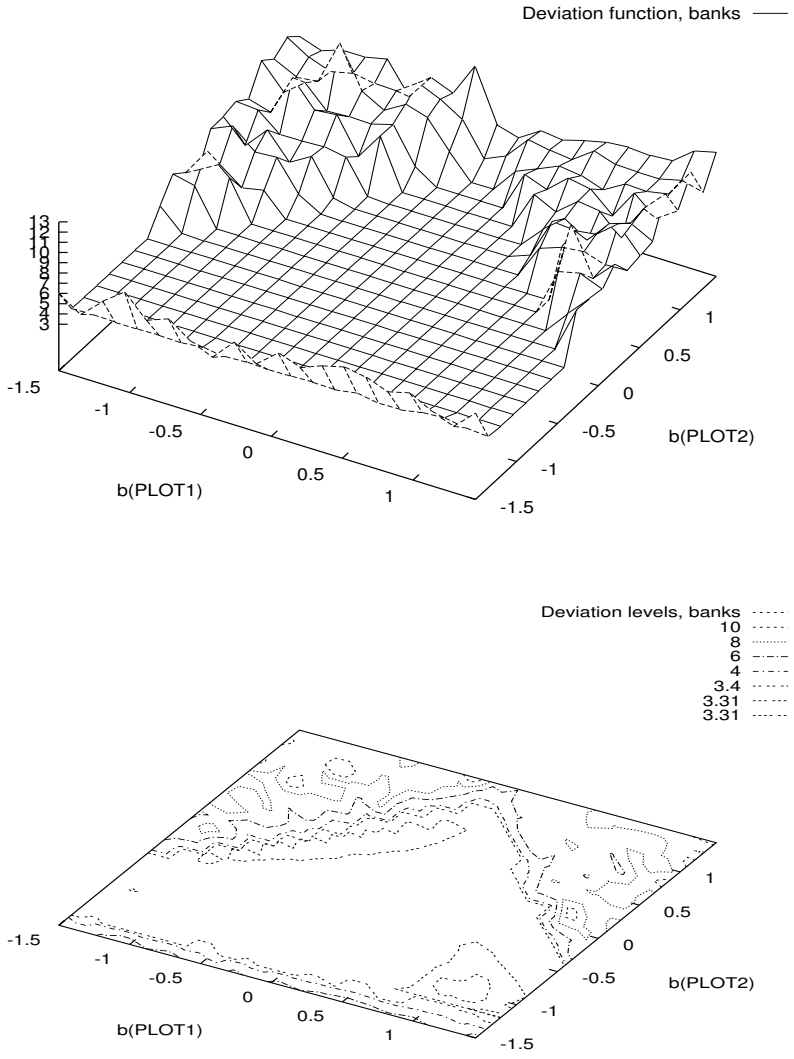


Figure 15.17. Closing rates of Hermis bank stocks: surface of f depending on parameters $b_0,\ b_1$ (top) contours of f depending on parameters $b_0,\ b_1$ (bottom).

Illustrative Example of ARMA Software: Stock Rate Prediction

Figures 15.20 and 15.21 show the part of the file 'fitimeb.h' defining the control parameters and data files.

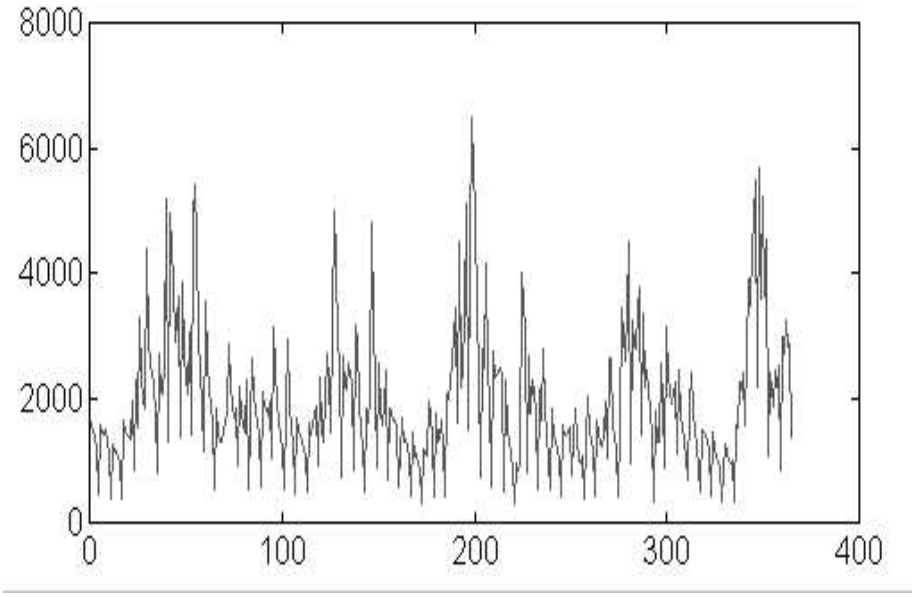


Figure 15.18. Call rates.

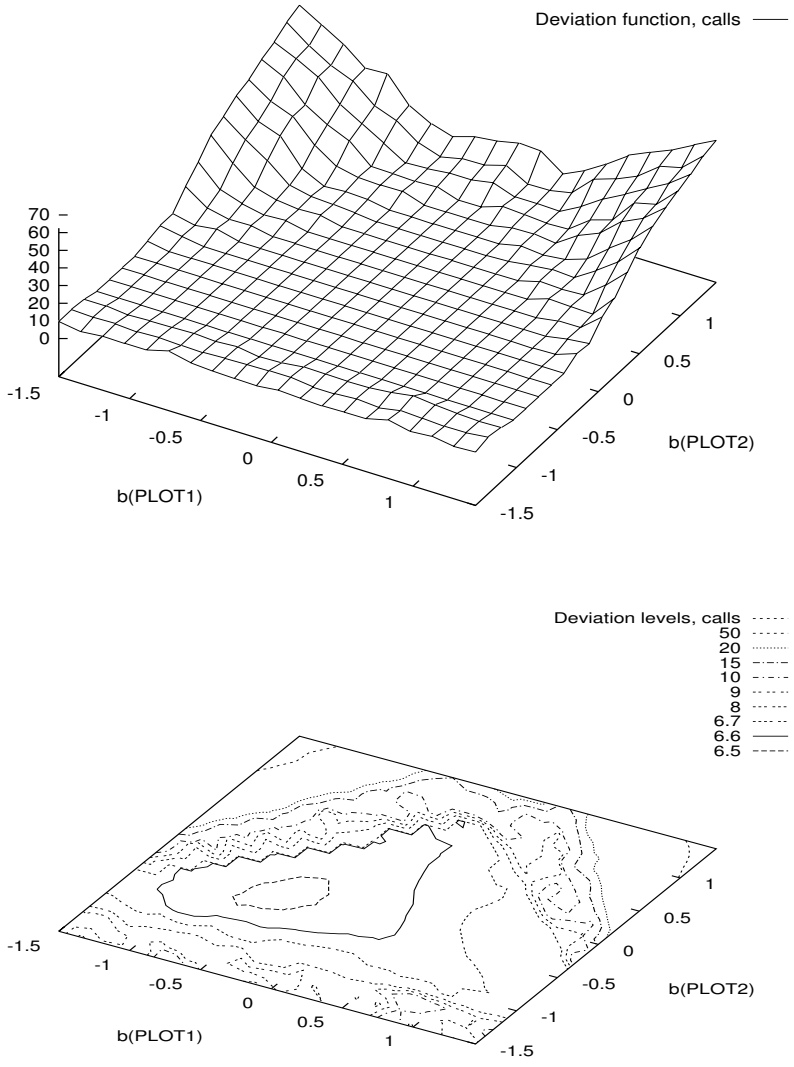


Figure 15.19. Call rates: surface of f depending on parameters $b_0,\ b_1$ (top), contours of f depending on parameters $b_0,\ b_1$ (bottom).

```
#define RAND()  drand48()
#define S 0 /* number of rows of matrix c */
#if S == 0
#define S1 1
#else
#define S1 S
#endif
#define R 0 /* number of columns of matrix c */#define K 5 /*number of "multi-s
#define W 0 /*W 0 means the one-step
 structural optimization,
 W 1 defines the multi-step one*/
#define V 1 /*V 1 means with the multi-step prognoses,
V 0 means without*/
#define F  1 /*indicator of variance,
 F 1 involves variance*/
#define EPS 0/*indicator of residual printing*/
#define INP  0 /*indicator of input control*/
#define SA  0 /*indicator of simulated annealing */
#define PL  0/*plotting dimension*/
#define PLOT1 0 /*first plotting coordinate is b[PLOT1}*/
#define PLOT2 1/*second plotting coordinate is b[PLOT2}*/
#define A1 -1.5 /*lower bound of  b[PLOT1}*/
#define B1 1.5/*upper bound of  b[PLOT2}*/
#define A2 -1.5 /*lower bound of  b[PLOT2}*/
#define B2 1.5/*upper bound of  b[PLOT2}*/
#define DN 50 /*number of plotting steps*/
#define ST 10000. /*temperature of simulated annealing */
#define SI 100 /*number of
simulated annealing iterations*/
#define Ps M /*starting number of AR parameters*/
#define Qs 0 /*starting number of MA parameters*/
#define Pmin M /*minimal number of AR parameters*/
#define Qmin 0 /*minimal number of MA parameters*/
#define Pmax 2*M  /*maximal number of AR parameters*/
#define Qmax 2 /*maximal number of MA parameters*/
```

Figure 15.20. Example of the file 'fitimeb.h', part 1.

```
#define T 120*M /* number of data entries in DATAFILE
(divisible by M)*/
#define T0 T/3 /*T0<T number of  entries
for a and b optimization (divisible by M)*/
#define T1 2*T0 /*T1>=T0, number of  entries
for P and Q optimization(divisible by M)*/
#define TR -1*M /*TR <= T, TR/M is the  number of
the first line in DATAFILE
used for simple regression
(negative TR prints no regression)*/
#define TE 25*M/*TE>TR, TE/M is the number of
the last line for regression*/
#define INIFILE "bank2.ini"
#define M 2 /*number of factors*/
#define MAXCOLS  120
#define DM 200 /*array size*/
#define MAX_B_BOUND 1.0
    /* -MAX_B_BOUND <= b[i] <= MAX_B_BOUND */
#define MAX_C_BOUND 1.0 /* -MAX_B_BOUND <= c[i][j]
<= MAX_B_BOUND */
//#define LOCAL_METH NLP
#define LOCAL_METH EXKOR
//#define GLOBAL_METH EXKOR
//#define GLOBAL_METH GLOPT
#define GLOBAL_METH BAYES1
//#define LOCAL_METH GLOPT
#define GLOPT_MAX_IT 800    /* glopt IT */
#define GLOPT_LT 100        /* glopt LT */
#define GLOPT_MAXL 200  /* glopt MAXL */

#define NLP_MAX_IT 10       /* nlp IT */
#define NLP_M 0         /* nlp M  */
#define NLP_ME 0        /* nlp ME */

#define BAYES1_MAX_IT   5*M /* bayes1 IT */
#define BAYES1_LT    5  /* bayes1 LT */

#define EXKOR_MAX_IT     6*M    /* exkor IT */
#define EXKOR_INIT_POINTS 6 /* exkor LT */
```

Figure 15.21. Example of the file 'fitimeb.h', part 2.

The control parameters and the optimization results (in the case INP 0) are shown in Figure 15.22.

```
Number of Factors M 2
Multi-step prognoses indicator V 1
Number of prognoses repetitions K 5
indicator of input control INP 0
the first line in datafile for regression, TR  -2
the  last line in datafile for regression, TE  50
starting number of AR parameters Ps 2
minimal number of AR parameters Pmin 2
minimal number of MA parameters Qmin 0
maximal number of AR parameters Pmax 4
maximal number of MA parameters Qmax 2
number of data entries T 240
number of  entries for a and b optimization T0 80
number of  entries for P and Q optimization T1 160
indicator of variance F 1
Global Max Iterat 10
Local Max Iterat 12
b-Bounds 1.000000e+00
Diskret Params Init
(Ps-AR params, Qs-MA params,ts-starting time) :
Ps Qs ts  = 2 0 0
Diskret  Params Step :  dP dQ dt = 2 1 1
Diskret  Params Max :  Pmax Qmax tmax = 4 2 0
Start AR:  l=0_Ps a[l] =  -9.152631e-02
Start AR:  l=0_Ps a[l] =  1.083756e+00
Start Results: opt delta_perc delta_mean=
1.140519e+01  4.830240e+01  6.171501e+00
Start Discret: Po Qo to   opt   delta_perc  delta_mean =
2 0 0 1.140519e+01 4.830240e+01 6.171501e+00
Start  AR:  l=0_Po  a[l] =  -9.152631e-02
Start  AR:  l=0_Po  a[l] =  1.083756e+00
BAYES,
EXKOR,
Finish Discret: Po Qo toP Q t0 opt delta_perc delta_mean=
2  0  0       2  0  0
1.140519e+01  4.830240e+01  6.171501e+00
Finish  ARo:  l=0_P  a[l] = -9.152631e-02
Finish  ARo:  l=0_P  a[l] = 1.083756e+00
Finish: sigma 2.054141e+01
Finish progn_err 2.438730e+01
```

Figure 15.22. Example of the file 'results.out.'

The comments in Figures 15.20 and 15.20 are short. Here are some additional explanations

- `#define RAND()  drand48(),`

  defines the random number generator

- ```
  #define S 0 /* number of rows of matrix c */
  #define S 0 /* number of rows of matrix c */
  ```

  defines the bilinear component (see expression 15.50)

- `#define K 5 /*number of "multi-step" repetitions*/,`

  K 5 means that the multi-step predictions is repeated 5 times (see Section 10 and Figure 15.23)

- ```
  #define W 0 /*W 0 means the one-step
  structural optimization,
  W 1 defines the multi-step one*/,
  ```

  one-step structural optimization minimizes the average error of the "next day" predictions (see Section 11.3), multi-step one minimizes the average error of predictions for longer periods of time

- ```
  #define V 1 /*V 1 means with the multi-step prognoses,
  V 0 means without*/
  ```

  the zero value of the indicator V switches off the multi-step prediction (see Section 10) and switches on the next day prediction (see Section 3.3 and Figure 15.23)

- ```
  #define F  1 /*indicator of variance,
   F 1 involves variance*/,
  ```

  the unit value of the indicator F means that the variance of the errors ϵ is estimated and included into the multi-step prediction process

- `#define EPS 0/*indicator of residual printing*/,`

  not used in this version

- `#define INP  0 /*indicator of input display*/,`

  if INP 1 then the input values of the predicted factor are printed with their numbers (see Figure 15.24)

- `#define SA  0 /*indicator of simulated annealing */`

  if SA 1 then optimization of the parameters p and q are performed using the simulated annealing method, otherwise the exhaustive search is used

```
progn t  ymin[M*t]  yav[M*t]  ymax[M*t]
 80    2.212638e+02    2.415062e+02    2.549196e+02

progn t  ymin[M*t]  yav[M*t]  ymax[M*t]
81    2.479502e+02    2.657038e+02    2.954255e+02

progn t  ymin[M*t]  yav[M*t]  ymax[M*t]
82    2.327805e+02    2.725516e+02    3.380554e+02

progn t  ymin[M*t]  yav[M*t]  ymax[M*t]
83    1.911297e+02    2.686203e+02    3.272436e+02

progn t  ymin[M*t]  yav[M*t]  ymax[M*t]
84    2.059219e+02    2.715981e+02    3.346959e+02

progn t  ymin[M*t]  yav[M*t]  ymax[M*t]
85    1.989816e+02    2.731343e+02    3.315979e+02

progn t  ymin[M*t]  yav[M*t]  ymax[M*t]
86    1.998928e+02    2.742784e+02    3.425031e+02

progn t  ymin[M*t]  yav[M*t]  ymax[M*t]
87    1.946436e+02    2.708346e+02    3.392612e+02

progn t  ymin[M*t]  yav[M*t]  ymax[M*t]
88    1.870341e+02    2.722931e+02    3.534729e+02

progn t  ymin[M*t]  yav[M*t]  ymax[M*t]
89    1.647438e+02    2.618776e+02    3.345000e+02
```

Figure 15.23. A fragment of the file 'progn.out', ten day predictions.

- `#define PL  0/*plotting dimension*/,`

  if PL 1 then the values of objective function depending on the parameter b[PLOT1] will be written in the file 'plot.out',
  if PL 2 then the values of objective function depending on two parameters b[PLOT1] and b[PLOT2] will be written in the file 'plot.out' (see Figure 15.25) and plotted as Figures 15.17 using the "Gnuplot" system,
  if PL 0 there will be no file 'plot.out'

- `#define PLOT1 0 /*first plotting coordinate is b[PLOT1}*/`
  `#define PLOT2 1/*second plotting coordinate is b[PLOT2}*/,`

```
 0    165.00
  1    110.00
  2    165.00
  3    110.00
  4    181.00
  5    110.00
  6    200.00
  7    110.00
  8    200.00
  9    110.00
  10    200.00
```

Figure 15.24. A fragment of input data the file 'bank.out' when INP is set to unit.

```
-1.000000e+00 0.000000e+00 3.323284e+00
-1.000000e+00 0.000000e+00 3.323284e+00
-1.000000e+00 0.000000e+00 3.323284e+00
-1.000000e+00 0.000000e+00 3.323284e+00
-1.000000e+00 0.000000e+00 3.323284e+00
-1.000000e+00 0.000000e+00 3.323284e+00
-1.000000e+00 0.000000e+00 3.323284e+00
-1.000000e+00 0.000000e+00 3.323284e+00
-1.000000e+00 0.000000e+00 3.323284e+00
-1.000000e+00 0.000000e+00 3.323284e+00
```

Figure 15.25. A fragment of the ten entries the 'bank.plot.out' file.

defines which components of vector-parameter b are considered

- ```
  #define A1 -1.5 /*lower bound of  b[PLOT1}*/
  #define B1 1.5/*upper bound of  b[PLOT2}*/
  #define A2 -1.5 /*lower bound of  b[PLOT2}*/
  #define B2 1.5/*upper bound of  b[PLOT2}*/
  #define DN 50 /*number of plotting steps*/
  ```

  defines the range and the density of the plotting points

- ```
  #define ST 10000. /*temperature of simulated annealing */
  #define SI 100 /*number
   of simulated annealing iterations*/
  ```

  defines the parameters of simulated annealing method (if SA is applied)

- ```
  #define Ps M  /*starting number of AR parameters*/
  #define Qs 0  /*starting number of MA parameters*/
  #define Pmin M /*minimal number of AR parameters*/
  #define Qmin 0  /*minimal number of MA parameters*/
  #define Pmax 2*M  /*maximal number of AR parameters*/
  #define Qmax 2  /*maximal number of MA parameters*/
  ```

  defines the initial, the minimal and the maximal number of parameters p and q in the structural optimization

- ```
  #define T 120*M /* number of data entries in DATAFILE
  (divisible by M)*/,
  ```

  defines the total number of entries in DATAFILE

- ```
  #define T0 T/3 /*T0<T number of  entries
  for a and b optimization (divisible by M)*/
  #define T1 2*T0 /*T1>=T0, number of  entries
  for P and Q optimization(divisible by M)*/,
  ```

  divides the DATAFILE (see Figures 15.26 and 15.16) into three parts: the first part for a,b optimization, the second part for p,q optimization, and the third part for testing the results

  ```
  #define TR -1*M /*TR <= T, TR/M is the  number of
  the first line in DATAFILE used for simple regression
  (negative TR prints no regression)*/
  #define TE 25*M/*TE>TR, TE/M is the number of
  the last line for regression*/,
  ```

  is used only in the case when the ARMA model of time series prediction is reduced to the linear regression model of diagnosis.

- ```
  #define INIFILE "bank2.ini",
  ```

  defines the input control file INIFILE (see Figure 15.27)

- ```
  #define M 2 /*number of factors*/,
  ```

  defines the number of factors, including the predicted and the external ones

- ```
  #define LOCAL_METH EXKOR,
  ```

  means that the EXKOR method is used for local optimization

- ```
  #define GLOBAL_METH BAYES1
  ```

  means that the BAYES1 method is used for global optimization

- ```
  #define BAYES1_MAX_IT   5*M /* bayes1 IT */
  ```

```
      5786.09   180.00   224.00      250.00
40.00    20.25    85.00    16.00
      5796.10   180.00   222.00      225.00
40.00    20.25    85.00    16.00
      5806.11   180.00   222.00      247.50
40.00    20.00    80.00    16.00
      5816.12   181.00   225.00      247.00
40.00    20.00    80.00    15.99
      5826.13   182.00   225.00      240.00
40.00    20.00    84.50    15.99
      5836.16   185.00   225.00      238.00
36.00    20.00    85.00    15.95
      5846.17   188.00   220.00      238.00
36.00    20.00    85.00    14.36
      5856.18   188.00   220.00      236.00
36.00    20.00    85.00    14.30
      5866.19   190.00   228.00      230.00
36.00    20.00    85.00    15.45
      5876.20   195.00   230.00      230.00
36.00    20.00    85.00    15.00
      5886.23   190.00   230.00      221.00
36.00    20.00    85.00    15.50
```

Figure 15.26. A fragment of the ten recent days of the DATAFILE 'bank.data' including the date code and data of seven main Lithuanian commercial banks.

```
INP bank.data
COL 3
COL 4
```

Figure 15.27. Example of the INIFILE 'bank2.ini.'

```
#define BAYES1_LT    5  /* bayes1 LT */
```

means that 5*M iterations and 5 initial iterations of the BAYES1 method are used

- ```
  #define EXKOR_MAX_IT     6*M    /* exkor IT */
  #define EXKOR_INIT_POINTS 6 /* exkor LT */
  ```

  means that 6*M iterations and 6 initial iterations of the EXKOR method are used
  ```

Illustrative Example of ARMA Software: Simple Test File

The test file is defined by 'test.ini':

```
INP test2
COL 1
COL 2
```

Here INP defines the data file 'test2'

```
1.   1.
2.   1.
1.   1.
2.   1.
1.   1.
2.   1.
1.   1.
2.   1.
1.   1.
```

Here COL 1 means that the first column of the file 'test2' should be considered as the factor to be predicted. COL 2 indicates the second column as an external factor.

A fragment of the optimization results is shown in the Figure 15.28. Here 'delta_perc' denotes the error in percents of the "Random Walk" prediction. 'delta_mean' is the average error. The results of optimization are printed as Po, Qo, ao, bo. Predicted results are in the file 'progn.out.old' (see Figure 15.29). The minimal $ymin$, the average yav, and the maximal $ymax$ predicted values are equal. Therefore, the prediction variance $progn$ is zero.

13.2 Java version of ARMA Software (ARMAJ)

The Java version of ARMA software (ARMAJ) (Kuzminaite, 1999) implements the same algorithms as the C++ version ARMAC. The ARMAJ is on the web-sites (see Section 4) and can be run by remote users.

```
#define V 1 /*V 1 means with the multi-step prognoses,
V 1 means without*/
#define F  0 /*indicator of variance,
 F 1 involves variance*/
#define Ps M /*starting number of AR parameters*/
#define Qs 0 /*starting number of MA parameters*/
#define Pmin M /*minimal number of AR parameters*/
#define Qmin 0 /*minimal number of MA parameters*/
#define Pmax 2*M  /*maximal number of AR parameters*/
#define Qmax 0 /*maximal number of MA parameters*/
#define T 16 /* number of data entries in DATAFILE
(divisible by M)*/
#define T0 6 /* number of  entries
 for a and b optimization*/
#define T1 12 /* number of  entries
 for P and Q optimization*/
#define INIFILE "test.ini"
#define M 2 /*number of factors*/
Ps Qs ts  = 2 0 0
Diskret  Params Step :  dP dQ dt = 2 1 1
Diskret  Params Max :  Pmax Qmax tmax = 20 0 0
Start AR:  l=0_Ps a[l] =  3.000000e+00
Start AR:  l=0_Ps a[l] =  -1.000000e+00
Start Results: opt delta_perc delta_mean= 0.000000e+00
-1.000000e+02 0.000000e+00
Start Discret: Po Qo to   opt   delta_perc  delta_mean =
2 0 0
0.000000e+00 -1.000000e+02 0.000000e+00
Start  AR:  l=0_Po  a[l] =  3.000000e+00
Start  AR:  l=0_Po  a[l] =  -1.000000e+00
Finish Discret: Po Qo toP Q t0 opt delta_perc delta_mean=
2  0  0
2  0  0      0.000000e+00  -1.000000e+02  0.000000e+00
Finish  ARo:  l=0_P  a[l] = 3.000000e+00
Finish  ARo:  l=0_P  a[l] = -1.000000e+00
Finish: sigma 0.000000e+00
Finish progn 0.000000e+00
Finish: sigma 0.000000e+00
```

Figure 15.28. A fragment of the control parameters and optimization results of the simplest illustration.

```
progn t  ymin[M*t]  yav[M*t]  ymax[M*t]      6
1.000000e+00    1.000000e+00    1.000000e+00
progn t  ymin[M*t]  yav[M*t]  ymax[M*t]      7
2.000000e+00    2.000000e+00    2.000000e+00
progn 0.000000e+00
```

Figure 15.29. Example of the prediction results of the simplest illustration.

Users Guide

The applet 'index.html' is started by a browser, for example, by Netscape 4.6, or by the appropriate appletviewer. One clicks the button 'Show' (see the top Figure 15.30) to open the main window 'ARMA Frame' (see the bottom Figure 15.30). There are four buttons: 'File,' 'Input,' 'Options,' and 'Output' which open corresponding windows.

File is for data input.
There are two fields: 'INI File' and 'Working directory or URL.'
There is the button 'Browse...' and two options:
'Local file' and 'Local URL.'

The option 'Local file' activates the 'Browse...' button to select some local file.
The option 'Local URL' closes this button. Then the contents of fields 'INI File' and 'Working directory or URL' determine the data file.
The file name, for example, 'arma.ini,' is in the field 'INI File.'
The file 'arma.ini' controls the data input from the test file 'arma.test.' The directory is in the field 'Working directory or URL.'
If the 'Local URL' option is on, then the directory is, for example, this:

```
http:/optimum.mii.lt/~jonas/armajavaj
```

The default URL is the applet directory.

Input is to change the default values of parameters (see the top Figures 15.31).
The list of parameters is the same as in the C file 'fitimeb.h.'

Options are to select options (see the top Figure 15.32):

Output options defines the output file,

None means no output,

System output means console output,

Frame defines output into separate windows,

File outputs data into local files[14]

Graphic options involves graphics.

Output is for data output (see the bottom Figure 15.32):
one starts computations by clicking the button 'Calculate',
the message ' ARMA Frame: calculated' indicates the end.

Some input and output operations are not permitted by the 'Security Manager.' Using the 'appletviewer,' it is possible to bypass the 'Security Manager.' For example, in the file $/.hotjava/properties$ one writes:

```
acl.read=/home/jonas/public\_html/armajavaj/arma.ini
acl.write=/home/jonas/public\_html/armajavaj/arma.out
```

That permits to read from 'arma.ini' and to write into 'arma.out.'

The file $arma\_all.jar$ is the 'jar' archive including all the 'class' files.
$source.jar$ or $source.zip$ are archives of 'java' files.
'index.html' is a starting applet.
'arma.ini' is the input control file.
'armatest' is the test data file.

Illustrations

The top Figure 15.30 shows the applet sign with the 'Show' button that starts ARMAJ. The bottom Figure 15.30 shows the initial window where the input file 'arma.ini' is defined. button that starts the ARMAJ.

Figures 15.31 show the list of default values of control parameters, similar to those in the ARMA C++ version. The upper part of the list is on the top figure, the middle part is on the bottom one.

[14]This operation is not permited by browsers, as usuall.

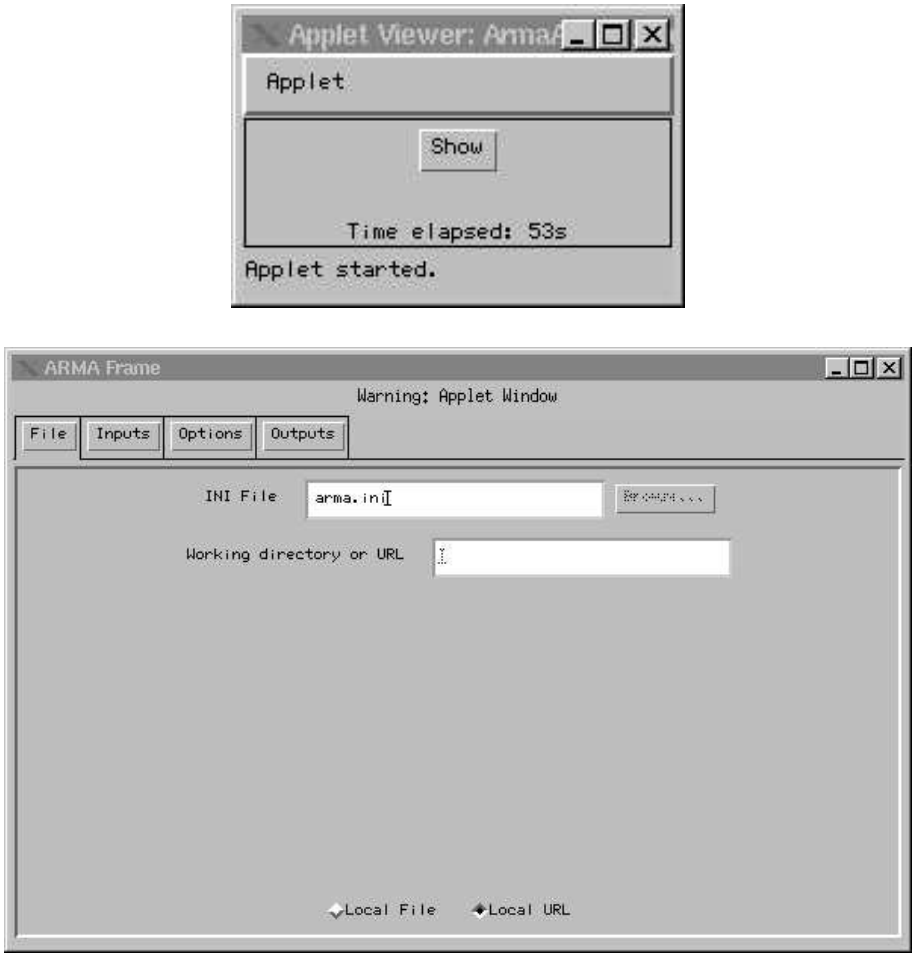


Figure 15.30. The applet sign (top figure), and the initial window defining the input file 'arma.ini'(bottom figure).

The top Figure 15.32 shows the option window. The bottom Figure 15.32 shows the last fragment of the output window. Here the results, similar to those in the ARMA C++ version, are written.

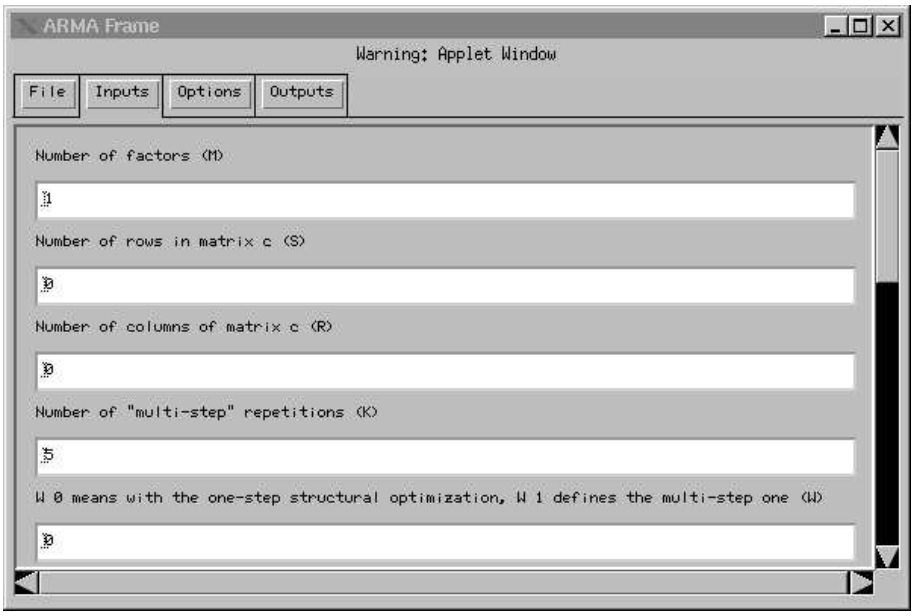
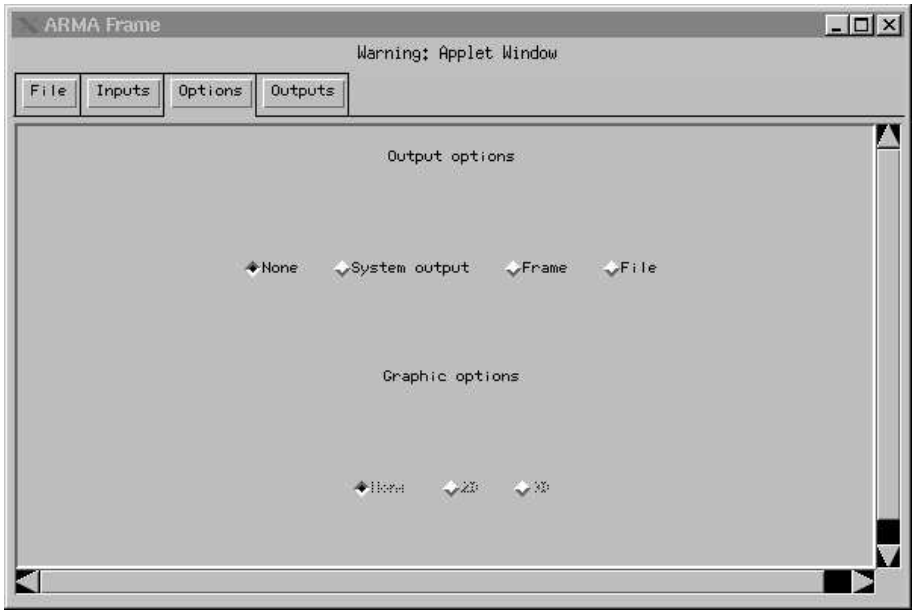


Figure 15.31. The list of default values of control parameters, the upper part is on the top figure, the middle part is on the bottom one.



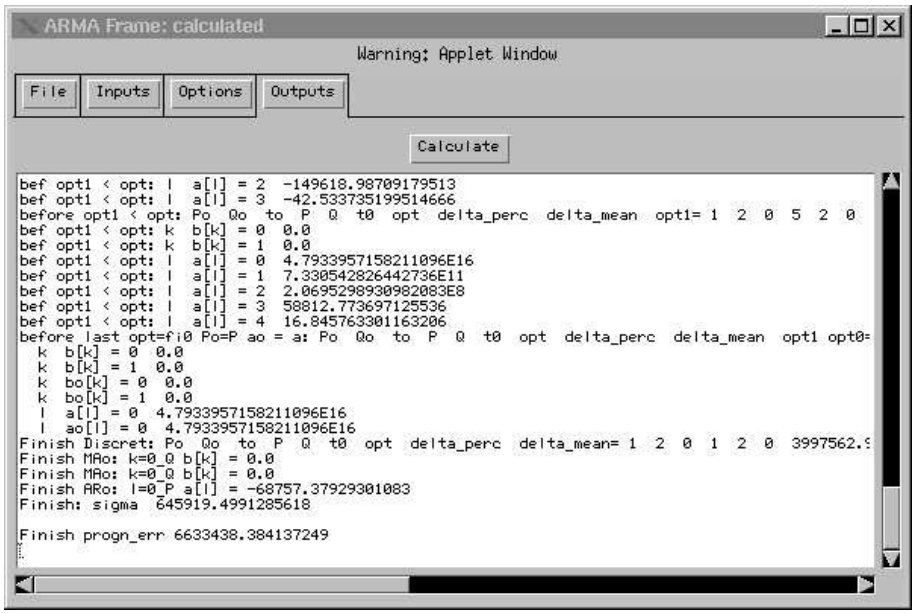


Figure 15.32. The option window (top figure) and the last fragment of the output window.

13.3 AR-ABS Model

The Java1.3 version for estimate of of AR-ABS parameters is on the web-sites and can be run by all the Internet users.

The applet is started by a browser or by the appropriate appletviewer. First one clicks the label 'Model' (see the Figure 15.33) to select the data file, for example, the file 'armatest,

To select the parameters one clicks the label 'Inputs', see Figure 15.34).

The calculations are started by the label 'Count', Figure 15.35 shows the results.

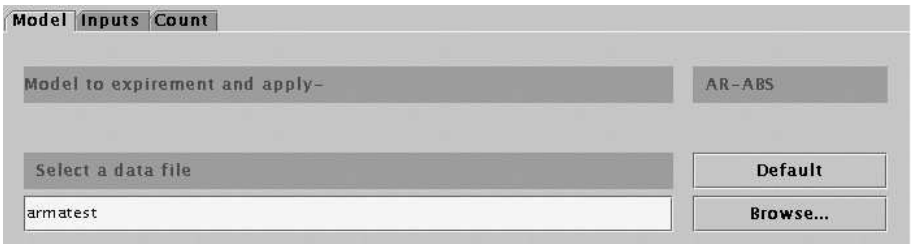


Figure 15.33. The initial window defining the input file 'armatest'.

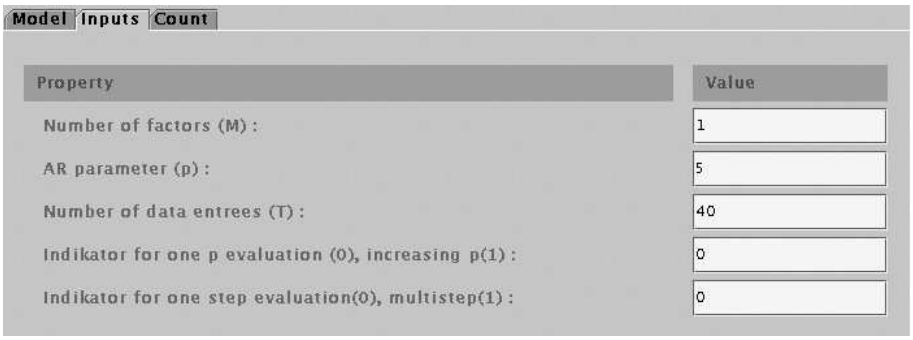


Figure 15.34. The input data.

13.4 ANN Software

A version of the ANN model (see expressions 15.46 and 15.49) is in the file $fi.C.anngauss$ on the web-site (see Section 4). $fi.C.anngauss$ is designed as an objective function of the GMC global optimization system.

The software is compiled and run this way:

```
Extract files by 'tar -zxf gmc.tgz'
Rename the file 'fi.C.anngauss' by 'cp fi.C.anngauss fi.C'
Compile by 'make',
Run by './test'
```

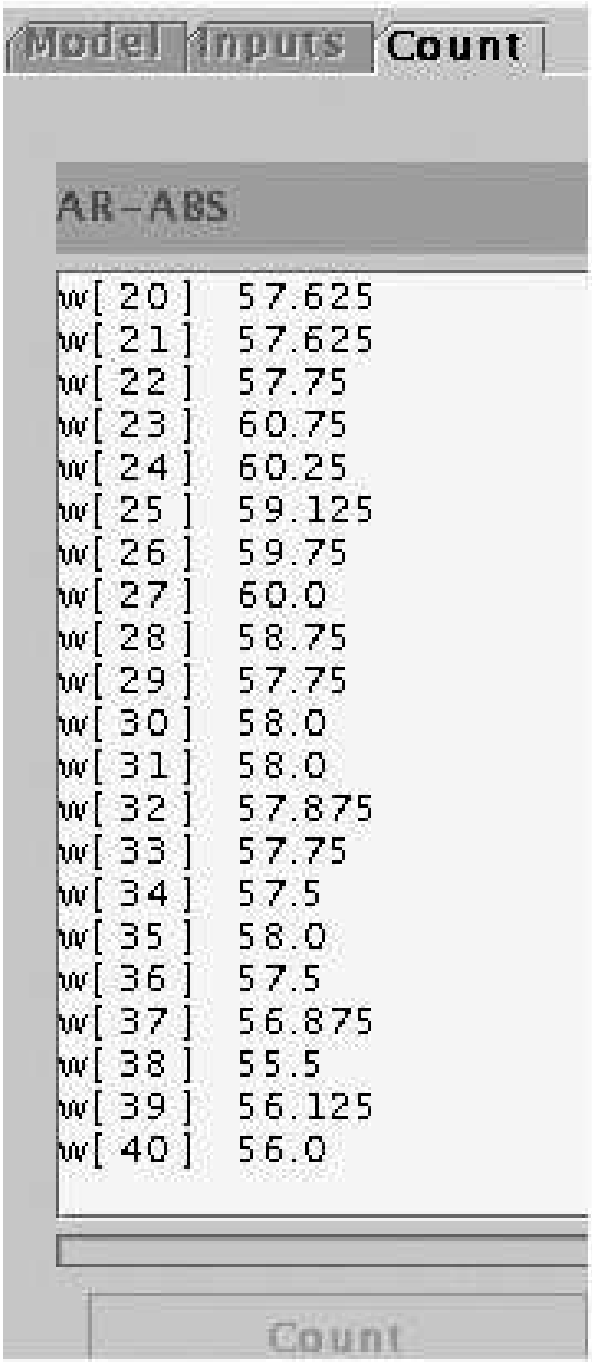


Figure 15.35. The results.

14. Stock Exchange Game Model

It is easy to obtain daily stock rates for a long time. However, this information alone is not always useful while predicting the future stock rates (Mockus et al., 1997, Raudys and Mockus, 1999).

To improve predictions, one adds additional factors, such as, relations of cash sales[15] to the sum of inventories and credit sales[16]. However, these factors are available quarterly, as usual. Therefore, it is difficult to unite them with the daily exchange rates in the same model.

The aim of the Stock Rate Exchange Model is to explain the poor predictability of stock rates. Therefore, we start by considering the simplest case: a single stock-broker, I major customers $i = 1, ..., I$, and J joint-stock companies $j = 1, ..., J$..

14.1 Monte Carlo Algorithm

One assumes that the stock rates $z^j(t)$ of a joint stock company j at a time t depend on bying-selling strategies of customers $i = 1, ..., I$ and some random factors $\epsilon_j(t)$. At a time $t = 1, ..., T$ the customer i orders a stock-broker to buy a number $n = n_i^j$ of j shares[17] , if

$$z^j(t) \leq z_i^{j_{buy}}(n, t). \tag{15.78}$$

Here $z_i^{j_{buy}}(n, t)$ is a buying threshold of the customer i of the number n of j shares at the moment t. At a time $t = 1.2, ..., T$ the customer i orders a stock-broker to sell a number $n = n_i^j(t)$ of j shares, if

$$z^j(t) \geq z_i^{j_{sell}}(n, t). \tag{15.79}$$

Here $z_i^{j_{sell}}(n, t)$ is a selling threshold of the customer i of the number n of j shares at the moment t.

The buying threshold is feasible if

$$n \leq N(t) \tag{15.80}$$

[15] Sales with immediate payment.
[16] Sales with delay of payments.
[17] Shares of company j.

Here $N_i(t)$ is the buying capacity of the customer i at the time t. This is defined, for example, as the initial wealth N_i plus the balance at the time t[18].

In this case, the stock rate of the j share at a moment $t+1$

$$z^j(t+1) = \begin{cases} z^j(t) + \epsilon_j(t+1), & \text{if } z^{j_{buy}}(t) \le z^j(t) \le z^{j_{sell}}(t), \\ z^{j_{buy}}(n,t) + \epsilon_j(t+1), & \text{if } z^j(t) \le z^{j_{buy}}(n,t), \\ z^{j_{sell}}(n,t) + \epsilon_j(t+1), & \text{if } z^j(t) \ge z^{j_{sell}}(n,t) \end{cases} \tag{15.81}$$

Here

$$z^{j_{buy}}(t) = \max_{n,i} \; z_i^{j_{buy}}(n,t) \tag{15.82}$$

and

$$z^{j_{sell}}(t) = \min_{n,i} \; z_i^{j_{sell}}(n,t). \tag{15.83}$$

The buying and selling thresholds depend on the difference $\Delta_i^j(t+1)$ between the expected profit of j share and that of certificate of deposit (CD) of the same value.

$$\Delta_i^j(t+1) = \left(\beta_i^j(t+1) + \delta_j(t+1) - \alpha(t+1)\right) z(t). \tag{15.84}$$

Here

$$\beta_i^j(t+1) = \frac{z_i^j(t+1) - z^j(t)}{z^j(t)}, \tag{15.85}$$

where $z_i^j(t+1)$ is the stock rate of the share j predicted by the customer i for the time $t+1$.
Denote the fraction

$$\delta_j(t+1) = \frac{d_j(t+1)}{z^j(t)}, \tag{15.86}$$

where $d_j(t+1)$ is the dividend expected for a share j at the moment $t+1$.
Denote the relation

$$\alpha(t+1) = a_j(t+1)/z^j(t), \tag{15.87}$$

where $a_j(t+1)$ is the yield of CD of value equal to $z^j(t)$ at the time $t+1$.
Now one defines the buying threshold

$$z_i^{j_{buy}}(t) = k^{buy} \; \Delta_i^j(t), \; k_i^{buy}(n) > 1 \tag{15.88}$$

[18]The balance is negative, if the buying expences exceed the selling ones.

and the selling threshold

$$z_i^{j_{sell}}(t) = k^{sell}\ \Delta_i^j(t),\ k_i^{sell}(n) < 1 \tag{15.89}$$

Here parameters $k_i^{buy}(n)$ and $k_i^{sell}(n)$ shows the risk aversion of the customer i and approximately reflects a personal utility function. As usual, $k_i^{buy}(n+1) \leq k_i^{buy}(n)$ and $k_i^{sell}(n+1) \geq k_i^{sell}(n)$.

The number of j shares own by the customer i at the time t from (15.81) assuming the feasibility condition (15.80)

$$N_i^j(t+1) \quad = \quad \begin{cases} N_i^j(t) & \text{if } z_i^{j_{buy}}(t) \leq z^j(t) \leq z_i^{j_{sell}}(t), \\ N_i^j(t) + n_i^j(t) & \text{if } z^j(t) \leq z_i^{j_{buy}}(t). \\ N_i^j(t) - n_i^j(t) & \text{if } z^j(t) \geq z_i^{j_{sell}}(t) \end{cases} \tag{15.90}$$

The profit of the customer i during the interval $T_0 \leq t \leq T$

$$u_i(T, T_0) = \tag{15.91}$$

$$\sum_{j=1,\ldots,n} (N_i^j(T) z_T^j - N_i^j(T_0) z^j(T_0) - \sum_{t=T_0}^{T} (N_i^j(t+1) - N_i^j(t))\ z^j(t)).$$

The profit u_i depends on the accuracy of stock rate predictions $\beta_i^j(t+1)$ and on random deviations $\epsilon_j(t)$.

Suppose, the customer i predicts the stock rate of the j share for the time $t+1$ using the Auto Regressive (AR) model:

$$z_i^j(t+1) = \sum_{k=1}^{p_i} a_i^k(j)\ z_{t-k}^j + \epsilon_{ji}(t+1). \tag{15.92}$$

Here the unknown parameters $a_k(j)$ are defined by minimization of squared deviations at fixed number p

$$\min_{a_i(j)} \sum_{s=t_0}^{t}\ \epsilon_{ji}^2(s), \tag{15.93}$$

where

$$\epsilon_{ji}(s) = z^j(s) - \sum_{k=1}^{p_i}\ a_i^k(j)\ z^j(s-k). \tag{15.94}$$

This time series model, hopefully, represents some aspects of stock rates.

14.2 Nash Model

The time series generated by Monte Carlo model depend on parameters p_i that are not defined by the model. To define these parameters, the "Nash Model" is used. The Nash Model searches for such numbers $(p_i,\ i = 1, ..., m)$ that satisfies the Nash equilibrium conditions. This means that no server can obtain higher expected profit u_i by changing p_i individually.

To ensure the existence of the Nash equilibrium one introduces mixed strategies. Here the mixed strategy is to select the integer p_i at random with probabilities

$$x_{p_i},\ i = 1, ..., m,\ p_i = 1, ..., P_i,\ \sum_{p_i=1}^{P_i} x_{p_i} = 1,\ 0 \le x_{p_i} \le 1. \qquad (15.95)$$

That means that one choses p_i by some lottery where x_{p_i} is the probability to win the integer p_i. The lottery is implemented by generating random numbers ξ uniformly distributed in the zero-one interval. This interval is partitioned into P_i parts proportional to p_i.

To make sense of this randomization, one repeates the Monte Carlo simulation K times. Denote $u_i(l, x_{p_i}\, i = 1, ..., m)$ the l-th sample of the profit function $u_i(T, T_)$ obtained by the i-th player using mixed strategies $(x_{p_i}\ i = 1, ..., m)$.

The average of all the K samples:

$$u_i^K(x_{p_i},\ i = 1, ..., m) = 1/K \sum_{l=1}^{K} u_i(l, x_{p_i},\ i = 1, ..., m). \qquad (15.96)$$

Denote the "contract" vector by $x_{p_i}^0,\ i = 1, 2$ Define the "fraud" vector (Raudys and Mockus, 1999) as

$$x_{p_i}^1 = \arg\max_{x_{p_i}} u_1^K(x_{p_i}, x_{p_l}^0,\ l \ne i), \qquad (15.97)$$

Ones searches for the Nash equilibrium $x_{p_i}^*$ by minimizing the difference between the fraud and contract vectors

$$(x_{p_i}^*,\ i = 1, ..., m) = \arg \min_{x_{p_i}^0,\ i=1,...,m} \sum_i (x_{p_i}^1 - x_{p_i}^0)^2 \qquad (15.98)$$

The optimization is carried out by methods of stochastic optimization. The convexity of profit functions is tested, if the minimum is greater than simulation errors (Raudys and Mockus, 1999).

14.3 Prediction of Simulated Stock Rates by ARMA Model

To compare the predictions of real and simulated stock rates the well known ARMA model is used. The estimates of parameters p, q of this model are obtained by stabilization procedure.

14.4 Software Example

This Java1.1 software implements the stock excange model in the case $I = 2$, $J = 1$, and $n = 1$. That means two major players buying a single share at a time and one major joint stock company. The purpose of this simplest case is to obtain some starting data and show advantages and disadvantages of the model. This information helps developing more complicated mdels.

Figure 15.36 describes the simulation results in the case when both players are predicting by the Wiener model: $p = 1,\ a_1 = 1,\ a_i = 0,\ i > 1$. The working hipothesis is that the behaviour of players based on the predictions obtained by the Wiener model provides the Nash equilibrium. Figure 15.37 describes the simulation results in the case when both players are predicting by the AR model: $p = 10,\ a_i = 0,\ i > 10$. One can see that the graphs are similar in both cases, the average deviations are close, too. Thus, there is no incentive for the players to change predictions based on the Wiener model. That means that the Wiener model can be regarded as a sort the equilibrium situation. One may conveniently investigate various examples directly by visiting the web-site: http://mockus.org/optimum and starting the corresponding Java applet (see the task "Stock Exchange Model" in the Section "Global Optimization")

This conclusion is supported by the results of AR models and partly supported by the results of ARMA models (Mockus et al., 1997, Raudys and Mockus, 1999). Using other models one may obtain different results. For example, the AR model predicts better if the relation $c(t)$ is included as an external factor, where

$$c(t) = \frac{cash(t)}{sold(t) + inventor(t)} \tag{15.99}$$

Here $cash(t)$ denotes the cash income during a time period t, $sold(t)$ means the contract price of the sold production during the same time, and $inventor(t)$ is the list price of the unsold production during the time t. There exists a positive correlation between the factors $z(t + delay\_time)$ and $c(t)$.

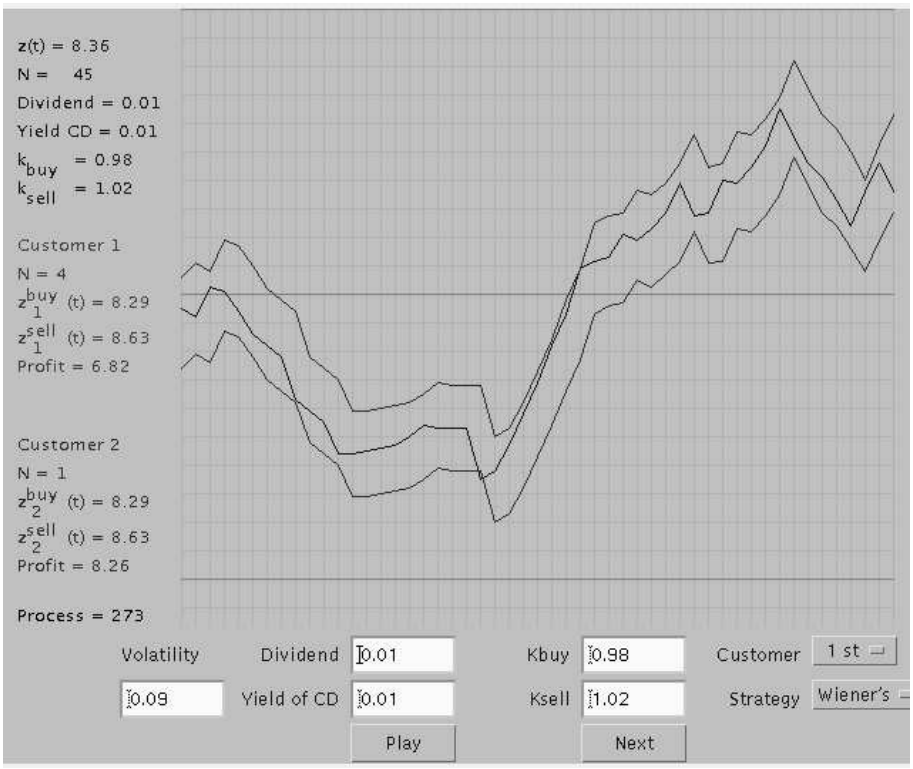


Figure 15.36. Simulation Results; Wiener Model

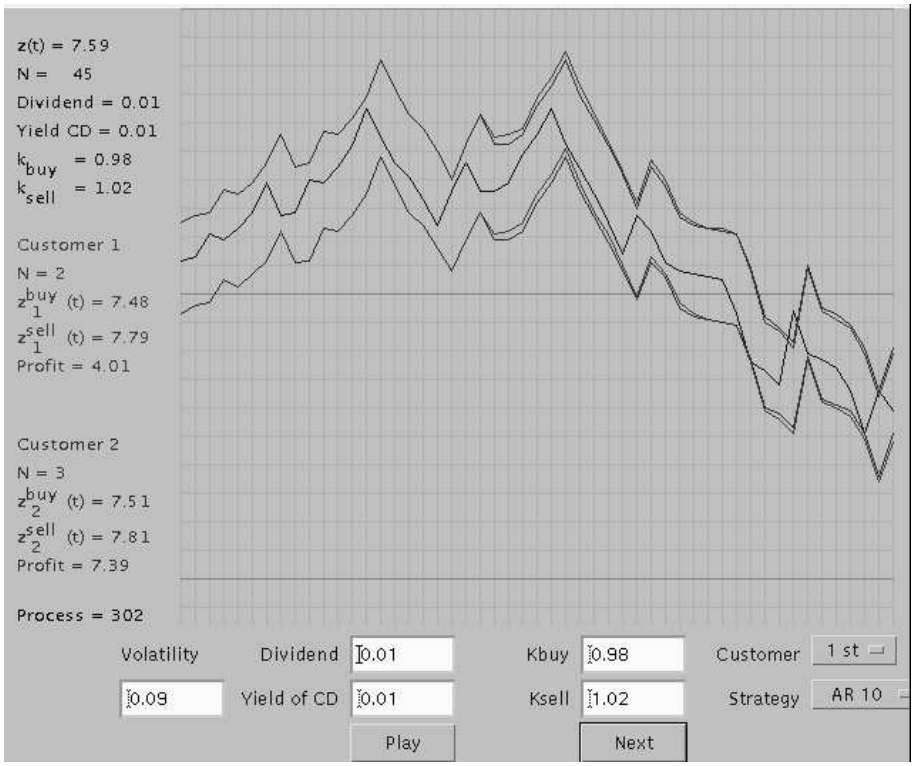


Figure 15.37. Simulation Results; AR Model, p=5

.

Chapter 16

CALL CENTER MODEL

1. Introduction

1.1 Outline

Call centers are important and rapidly developing commercial activities. Call centers serve customers by phone, by fax and by Internet. There are different call centers depending on their objectives and environments. However, investigating the call centers one encounters three problems:

- modeling;
  queuing systems are common tools while modeling call centers,
- optimization;
  one needs methods of stochastic and global optimization are to optimize these systems,
- prediction; parameters, which are needed for modeling and optimization, depends on predictions, predictions involve both the observed data and the expert knowledge.

1.2 Assumptions, Notations, and Objectives

One models call centers as queuing systems. The modeling system includes features for call rate predictions and service optimization. One assumes that

- incoming calls are united into one stream with the call rate λ, a call rate is the average number of calls in a time unit,
- there are m servers with the same service rate μ, the service rate is the average number of calls that can be served in a time unit,
- each server can serve any call, one at a time,
- calls are Poisson with the rate λ,
- service times are exponential with the rate $\mu > 1/m\ \lambda$, in such a case

$$\lambda = 1/T_\lambda, \tag{16.1}$$
$$\mu = 1/T_\mu, \tag{16.2}$$

  where T_λ is an average time between the calls and T_μ is an average service time,
- arriving calls enter the first available server, if all the servers are busy then the call takes a free waiting place,
- there are r waiting places, that means that a call waits, if there are not more than r other calls waiting, otherwise, the call disappears,
- waiting places are common for all the calls
- the system is stationary, meaning that the parameters μ and λ are constant,
- c_m is the server running cost (\$ per time unit),
- c_t is the customer time cost (\$ per time unit),
- c_p is the lost customer cost (\$ per lost call),
- the optimal number of servers $m = m(c, r)$ minimizes the total cost $C(c, m)$[1] per time unit, including the server running cost, the customer time cost, and the lost customer cost at fixed parameters r and $c = (c_s, c_t, c_p)$,
- results are presented as the waiting time distribution functions $F_\tau^{m,r}(t) = P^{m,r}\{\tau < t\}$, where $P^{m,r}\{\tau < t\}$ is the probability that the waiting time τ will be less than t at fixed number of servers m and waiting places r,
- a family of functions $F_\tau^{c,r}(t)$ is defined for different parameters c, r, these functions are presented in a simple format, assuming that the number of servers $m = m(c, r)$ is obtained by minimizing the total cost $C(c, m, r)$.

[1] Only two parameters c and m are in $C(c, m)$. The reason is uncertainty of c, therefore, the optimization of m is repeated several times at different "scenarios" c.

2. Calculation of Stationary Probabilities

Under these assumptions (Tijms, 1994), the probability of k calls in the system, including both, waiting calls and calls in services

$$P_k(\mu,\lambda) = \begin{cases} \frac{\rho^k}{k!}P_0(\mu,\lambda), & \text{if } 1 \le k \le m, \\ \frac{\rho^k}{m!m^{k-m}}P_0(\mu,\lambda), & \text{if } m < k \le m+r, \end{cases} \tag{16.3}$$

where $\rho = \lambda/\mu$. The no-calls probability

$$P_0(\mu,\lambda) = (\sum_{k=0}^{m-1}\frac{\rho^k}{k!} + \frac{m^m}{m!}\sum_{k=m}^{m+r}(\frac{\rho}{m})^k)^{-1} \tag{16.4}$$

follows from the condition

$$\sum_{k=0}^{m+r} P_k(\mu,\lambda) = 1. \tag{16.5}$$

The probability of losing a call at given numbers m, r, μ, λ is

$$P_{m+r}(\mu,\lambda) = \frac{\rho^{m+r}}{m!m^r}P_0(\mu,\lambda). \tag{16.6}$$

The average waiting time

$$T(m,r,\mu,\lambda) = \sum_{l=0}^{r-1} P_{m+l}(\mu,\lambda)\, t_l,\ r > 0, \tag{16.7}$$

where $P_k(\mu,\lambda)$ is defined by expression (16.3) and $t_l = \frac{l+1}{m\mu}$. The waiting time distribution functions

$$\begin{aligned} F_\tau^{m,r}(t) = P^{m,r}(\mu,\lambda)\{\tau < t\} = 1 - \Psi_\tau^{m,r}(t), \\ \Psi_\tau^{m,r}(t) = \sum_{k=m}^{m+r-1} P_k(\mu,\lambda)\, P_k\{\tau > t\}, \end{aligned} \tag{16.8}$$

Here $P_k\{\tau > t\}$ denotes the probability that waiting time τ will exceed some fixed t under condition that there are k calls

$$P_k\{\tau > t\} = \sum_{s=0}^{k-m} q_s(t),\ k \ge m. \tag{16.9}$$

Here $q_s(t)$ is the probability that s calls will be served during the time t and

$$q_s(t) = e^{-m\mu t}\frac{(m\mu t)^s}{s!},\ s = 0, 1, ..., r. \tag{16.10}$$

3. Asymptotic Expressions

If the number r of waiting places is very large, one considers asymptotic expressions ($r \to \infty,\ \rho < m$) as a reasonable approximation. The asymptotic values are denoted by the index ∞. They are used to test the software designed for finite r. The no-calls probability

$$P_0^{\infty}(\mu,\lambda) = (\sum_{k=0}^{m-1} \frac{\rho^k}{k!} + \frac{\rho^m}{(m-1)!(m-\rho)})^{-1}. \tag{16.11}$$

The average waiting time

$$T(m,\infty,\mu,\lambda) = \frac{\rho^m}{m!m\mu(1-\rho/m)^2} P_0^{\infty}(\mu,\lambda). \tag{16.12}$$

The waiting time distribution functions

$$\begin{aligned} F_{m,\infty}(t) &= P_{m,\infty}(\mu,\lambda)\{\tau < t\} \\ &= 1 - \Psi_{m,\infty}(t), \\ \Psi_{m,\infty}(t) &= \frac{1}{1-\rho/m} e^{-(m\mu-\lambda)t} P_m^{\infty}(\mu,\lambda). \end{aligned} \tag{16.13}$$

Here

$$P_m^{\infty}(\mu,\lambda) = \frac{\rho^m}{m!} P_0^{\infty}(\mu,\lambda). \tag{16.14}$$

The asymptotic probability of losing a call is zero.

4. "Surrogate" Services

An incoming call gets a regular service by an agent, if the actual or estimated waiting time is less then τ_w sec. Otherwise, a call gets a "surrogate" service by voice-mail. For simplicity, one replaces this service system by a system with $r = \tau_w/\tau_s$, where the expected service time is $\tau_s = 1/m\mu$. Then one can use all the expressions for the queuing system with r waiting places. Here, the call served by voice-mail is considered as the "lost" call. One considers calls of different type separately, because they are served by different agents.

5. Call Rate Estimate

The lost calls are not registered, as usual. Therefore, it is difficult to estimate the call rate λ directly, if r is limited. Then one uses least square estimates.

The square deviations $\Delta(\mu, \lambda)$ between stationary probabilities $P_k(\mu, \lambda)$ and their estimates P_k^0 are minimized.

$$\Delta(\mu, \lambda) = \sum_{k=0}^{m+r} (P_k(\mu, \lambda) - P_k^0)^2. \tag{16.15}$$

The stationary probabilities that there are k calls in the system are defined by expression

$$P_k(\mu, \lambda) \quad = \quad \begin{cases} \frac{\rho^k}{k!} P_0(\mu, \lambda), & \text{if } 1 \leq k \leq m \\ \frac{\rho^k}{m! m^{k-m}} P_0(\mu, \lambda), & \text{if } m \leq k \leq m + r. \end{cases} . \tag{16.16}$$

The least squares estimate of the call rate

$$\lambda^o = \arg \min_{\lambda} \Delta(\mu, \lambda). \tag{16.17}$$

The estimates P_k^0 are obtained by counting the numbers of waiting calls at different time moments. Additional errors are expected, if the average number of waiting calls in a time unit is counted, instead of moment numbers.

6. Optimization of Number of Servers

The total cost of a service system

$$C(c, m) = c_m m + c_t T(m, r, \mu, \lambda) + c_p P_{m+r}(\mu, \lambda). \tag{16.18}$$

One optimizes the server number m by simple comparison of different m within reasonable bounds $m_{min} \leq m \leq m_{max},\ m_{min} > \lambda/\mu$

$$m(c, r) = arg \min_{m_{min} \leq m \leq m_{max}} C(c, m, r). \tag{16.19}$$

7. Monte Carlo Simulation (MCS)

The explicit steady-state solutions (16.6)-(16.13) are simple and exact, under the assumptions. However, there is no general explicit solution, if the call rate λ, and/or the service rate μ depends on time. That often happens in real service systems. The configuration and operating rules of real service systems are too complicated for the exact solution, as usual.

Then the Monte Carlo Simulation (MCS) is needed. We start the discussion of MCS from the steady-state systems keeping all the assumptions described

in Section 1.2. That helps to test both the analytical and the Monte Carlo solutions by comparing the results. Later, MCS is extended to more complicated configurations and to time-dependent cases.

7.1 Event Generation

The events in a queuing system are the moments when a call arrives, when a call enters a server, and when a call leaves the system. To generate events, one needs two types of random number generators. The first type generates times until the next call. The second type generates service times.

Denote by $F_a(t) = P_a\{\tau < t\}$ the distribution function, where $P_a\{\tau < t\}$ is the probability that a random time τ will be less then t. Here a is the expected value of τ. Denote by $\xi \in [0, 1]$ the random variable uniformly distributed between zero and one. Then

$$\tau = F^{-1}(\xi). \tag{16.20}$$

In exponential cases

$$F_a(t) = 1 - e^{-1/at}, \tag{16.21}$$

$$\tau = -aln(1 - \xi). \tag{16.22}$$

Here $a = 1/\lambda_s$, if times until the next call are generated.
$a = 1/(m_s\mu_s)$, if service times are generated.
The model defines moments when a call enters a server and leaves the system. These moments depend on the specific structure of the system. The model is designed trying to represent the actual operations as realistically as possible. However, the are limits that depend on the available computing power.

Here is a description of an algorithm of modeling and optimization of call centers:

- denote by t_n the "arriving" time of the nth call, then $t_{n+1} = t_n + \xi_n(\lambda)$, $\xi(\lambda) = -1/\lambda\, log(1-\eta)$, where η is a random number, uniform in $[0, 1]$, and λ is the average number of calls in a time unit, the moment of the first call moment $t_1 = 0$ is the start of MCS,

- denote by τ_n the "departing" time of nth call, then $\tau_{n+1} = \tau_n + \xi_n(\mu_n)$, $\xi(\mu_n) = -1/\mu_n\, log(1 - \eta)$, where μ_n is the average number of calls which can be served in a time unit by all $1 \le m_n \le m$ servers operating at the moment

τ_n, thus $\mu_n = m_n\,\mu$, and μ is the average number of calls, served by single server in a unit time,

- count all calls in the system at the moment t_{n+1} including those in m servers and r waiting places by this expression

$$I_{n+1} = \begin{cases} I_n - 1, & \text{if } t_{n+1} > \tau_n \\ \min(I_n + 1, m + r), & \text{if } t_{n+1} < \tau_n, \end{cases} \tag{16.23}$$

where I_n is the number of calls in the system at the moment t_n,

- define the number of servers m_n operating at the moment τ_n by this condition

$$m_n = \min(I_l, m),\ t_l \le \tau_n \le t_{l+1}, \tag{16.24}$$

- count calls lost up to the moment t_{n+1} by the expression

$$L_{n+1} = \begin{cases} L_n + 1, & \text{if } I_n + 1 > m + r, t_{n+1} < \tau_n \\ L_n, & \text{otherwise,} \end{cases} \tag{16.25}$$

where L_n is the number of calls lost up to the the moment t_n,

- count the waiting time by this counter

$$T_{n+1} = T_n + \delta_n, \tag{16.26}$$

$$\delta_n = \tau_n - t_n, \tag{16.27}$$

- count the number of calls q^k that waited less than t^k, $t^{k+1} > t^k$, $k = 1, ..., K$

$$q^k_{n+1} = \begin{cases} \min(q^k_n + 1, m + r), & \text{if } \delta_n < t^k \\ q^k_n, & \text{if } \delta_n \ge t^k, \end{cases} \tag{16.28}$$

- define the distribution of the call numbers q^k that waited less than t^k, $t^{k+1} > t^k$, $k = 1, ..., K$

$$Q(t^k) = 1/N\ q^k_N, \tag{16.29}$$

- define the optimal number of servers m by simple comparison of different m within some reasonable bounds $m_{min} \le m \le m_{max}$

$$m(c, r) = arg \min_{m_{min} \le m \le m_{max}} C(c, m, r), \tag{16.30}$$

- compare the results with the corresponding steady-state explicit solutions, including the average waiting time, the average of lost calls and the waiting time distribution functions.

7.2 Monte Carlo Errors

Under independence conditions, the standard deviation $\sigma(K)$ of the results $\theta(K)$ of a statistical model depends on the number of observations K as

$$\sigma(K) = \sigma(1)/\sqrt{K}. \tag{16.31}$$

Here $\sigma(1)$ is the "initial" standard deviation

$$\sigma(1) = \sqrt{E(\theta(1) - \Theta(1))^2}, \tag{16.32}$$

where $\Theta(1) = E\theta(1)$ and E is the expectation symbol.

In Monte Carlo simulation, $\Theta(1)$ cannot be estimated directly using the results $\theta(1)$ obtained by the 1-th observation. Many observations are needed, to reach a stationary state (one observation means one call). Therefore, we repeat the Monte Carlo simulation L times doing K observations each time. Then the error variance can be estimated as

$$\sigma_L^2(K) = 1/(L-1)\sum_{l=1}^{L}(\theta_l(K) - 1/L\sum_{j=1}^{L}\theta_j(K))^2. \tag{16.33}$$

Expression (16.33) is simpler, if the exact solution $\Theta = \lim_{K\to\infty}\theta(K)$ is known . Then

$$\sigma_L^2(K) = 1/(L-1)\sum_{l=1}^{L}(\theta_l(K) - \Theta)^2 \tag{16.34}$$

Therefore, to test a Monte Carlo procedure, one considers a model with exact solution, first. Then one applies MCS to more complicated models.

7.3 Stopping Monte Carlo

The Monte Carlo errors are important to define stopping rules. The traditional idea is that the computing errors should not be greater then the data gathering errors. In call centers, data errors depend on the errors of call rate predictions, as usual. Thus, the stopping rule is:
- stop at the first observation $k \geq K$ satisfying this inequality:

$$\sigma(K) < \sigma. \tag{16.35}$$

Here σ is a standard deviation of call rate predictions.

8. Common Waiting

The common waiting system is an example were one applies approximate models, including the Monte Carlo. By common waiting we define a system that serves calls of different type. One represents them as a call-vector $\lambda = (\lambda_1, ..., \lambda_l)$. Different calls are served by different servers. However, there are $r(c)$ of common waiting places. There are no simple formulas, such as (16.16) and (16.15). Therefore, we consider two approximate solutions: the analytical "reservation" model, and the Monte Carlo one.

8.1 Analytical Approximation: Reservation Model

One estimates $\lambda = (\lambda_1, ..., \lambda_l)$ by minimizing the square deviations

$$\Delta_s(\mu, \lambda, r_s) = \sum_{k_s=0}^{m_s+r_s} (P_{k_s}(\mu_s, \lambda_s, r_s) - P_{k_s}^0)^2. \tag{16.36}$$

Here $P_{k_s}(\mu_s, \lambda_s, r_s)$ is the stationary probability that there are k_s calls in the s-th server. This probability is defined by expressions similar to (16.3), assuming that there are r_s waiting places reserved for the calls λ_s. $r = (r_1, ..., r_l)$ is the reservation vector and $\sum_s r_s = r(c)$.

$$P_{k_s}(\mu_s, \lambda_s, r_s) = \begin{cases} \frac{\rho_s^k}{k_s!} P_0(\mu_s, \lambda_s, r_s), & \text{if } 1 \le k_s \le m_s, \\ \frac{\rho_s^k}{m_s! m_s^{k-m_s}} P_0(\mu_s, \lambda_s, r_s), & \text{if } m_s \le k_s \le m_s + r_s. \end{cases}.$$

The estimates $P_{k_s}^0$ are obtained by counting the numbers of waiting s-calls at different time moments. The least square estimate of the call-vector λ is as follows

$$\lambda^o = \arg \min_{\lambda} \min_{r} (\sum_s \Delta(\mu_s, \lambda_s, r_s)). \tag{16.37}$$

The reservation model is simple and clear. However, one must test the reservation assumption by the Monte Carlo model.

8.2 Statistical Approximation: Monte Carlo Model

8.3 Call Rate Estimate

The statistical model can be used to estimate the call rates λ in the same way as the analytical one. One estimates $\lambda = (\lambda_1, ..., \lambda_l)$ by minimizing the square

deviations $\Delta(\mu, \lambda, r)$ between the probabilities and their estimates

$$\Delta(\mu, \lambda, r) = \sum_{s} \sum_{k_s=0}^{m_s+r} (P_{k_s}(\mu, \lambda, r) - P_{k_s}^0)^2. \tag{16.38}$$

Here $P_{k_s}(\mu, \lambda, r)$ is the probability[2] that there are k_s calls in the s-th server[3]. The estimates $P_{k_s}^0$ are obtained by counting the numbers of waiting s-calls at different time moments.

The least squares estimate of the call-vector λ

$$\lambda^o = \arg\min_{\lambda} \Delta(\mu, \lambda, r)). \tag{16.39}$$

The statistical model needs considerable computing power.

8.4 Testing Analytical Approximation

The analytical approximation (see Chapter 8.1) is simpler. However, it is based on the reservation assumption. The statistical model may represent most of the important factors. However, it takes a long time to "filter out" the random deviations. In the on-line operations, this time is too long, as usual.

To test analytical approximation, one compares estimates of call rates $\lambda = (\lambda_1, ..., \lambda_l)$ obtained by both the statistical and the reservation models. A small deviation means that the reservation model is acceptable. The large deviation means that the analytical approximation should be improved.

9. Time-Dependant Cases

Consider the MCS when the call rates $\lambda = \lambda(t)$, the number of servers $m = m(t)$, and the number of waiting places $r = r(t)$ depend on time. Therefore, all the results should be represented as functions of time. One modifies algorithms, similar to those in the previous chapter, by including the time factor:

- denote by t_n the "arriving" time of the nth call, then $t_{n+1} = t_n + \xi_n(\lambda(t))$, $\xi(\lambda(t)) = -1/\lambda(t)\ log(1-\eta)$, where η is a random number uniform in $[0, 1]$ and λ is

[2] Estimated using the Monte Carlo techniques described in the previous chapter.
[3] There are no reserved waiting places for the calls λ_s, thus the symbols r_s are omitted.

the average number of calls arriving in a time unit that includes the moment t,

- denote by τ_n the "departing" time of nth call, then $\tau_{n+1} = \tau_n + \xi_n(\mu_n)$, $\xi(\mu_n) = -1/\mu_n \; log(1-\eta)$ where μ_n is the average number of calls which can be served in a time unit by all the $1 \leq m_n \leq m(\tau_n)$ servers operating at the moment τ_n, thus $\mu_n = m_n \; \mu$ and μ is the average number of calls served by single server in unit time,
- count the number of all the calls in the system at the moment t_{n+1}, including those in $m(t_{n+1})$ servers and $r(t_{n+1})$ waiting places

$$I_{n+1} \quad = \quad \begin{cases} I_n - 1, & \text{if } t_{n+1} > \tau_n \\ \min(I_n + 1, m(t_{n+1}) + r(t_{n+1})), & \text{if } t_{n+1} < \tau_n, \end{cases}$$

where I_n is the number of calls in the system at the moment t_n,

- count the number of servers m_n operating at the moment τ_n

$$m_n = \min(I_l, m(\tau_n)), \; t_l \leq \tau_n \leq t_{l+1}, \tag{16.40}$$

- count the number of calls lost up to the moment t_{n+1}

$$L_{n+1} \quad = \quad \begin{cases} L_n + 1, & \text{if } I_n + 1 > m(t_{n+1}) + r(t_{n+1}), t_{n+1} < \tau_n \\ L_n, & \text{otherwise,} \end{cases}$$

where L_n is the number of calls lost up to the the moment t_n,

- define the waiting time

$$T_{n+1} = T_n + \delta_n, \tag{16.41}$$

$$\delta_n = \tau_n - t_n, \tag{16.42}$$

- count the number of calls s q^k that waited less than t^k, $t^{k+1} > t^k$, $k = 1, ..., K$

$$q^k_{n+1} \quad = \quad \begin{cases} \min(q^k_n + 1, m(t_{n+1}) + r(t_{n+1})), & \text{if } \delta_n < t^k \\ q^k_n, & \text{if } \delta_n \geq t^k, \end{cases}$$

- define the distribution of the call numbers q^k that waited less than t^k, $t^{k+1} > t^k$, $k = 1, ..., K$

$$Q(t^k) = 1/N \; q^k_N, \tag{16.43}$$

- optimize the number of servers $m(t)$ at different times t by solving the corresponding stochastic scheduling problem[4]

[4]Solution of this problem is difficult, thus, later on we shall consider some simple cases as illustrations.

- define the difference between the steady-state and dynamic solutions by comparing the average results with the corresponding steady-state explicit solutions, including the waiting time distribution functions $F_{\tau}^{m,r}(t) = P^{m,r}\{\tau < t\}$, where $P^{m,r}\{\tau < t\}$ is the probability that the waiting time τ will be less than t,
- define the iteration number N by Monte Carlo experimentation.

9.1 Simple Example

Consider this simple example

$$\lambda(t) = a + b\, sin(\phi + 2\pi\omega t). \tag{16.44}$$

Here $\omega = 1/24$ and $\phi = 8/24$. t is caunted in hours. a is average number of calls in 24 hours. b is amplitude of daily variation of calls, $r(t) = r$.

$$m(t) \quad = \quad \begin{cases} m^0, & \text{if } sin(\phi + 2\pi\omega t) < 0, \\ m^1, & \text{if } sin(\phi + 2\pi\omega t) > 0. \end{cases} \tag{16.45}$$

One optimizes the numbers m^0 and m^1 of two different servers considering two periods. This is a short description of the optimization problem:

- there are two optimization parameters m^0 and m^1 restricted by bounds $m_{min} \leq m^0 \leq m^0_{max}$ and $m^1_{min} \leq m^1 \leq m^1_{max}$,
- one may obtain the optimal pair by direct comparison of all the feasible pairs

$$m* = (m^{*0}, m^{*1}) = arg \min_{m^0, m^1} C(c^0, c^1, m^0, m^1, r), \tag{16.46}$$

- the total system cost is defined as

$$\begin{aligned} C(c^0, c^1, m^0, m^1, r) = \\ c^0 m^0 + c^1 m^1 + c_t T(m^0, m^1, r, \mu, a, b, \phi, \omega) + \\ c_p P_{m^0, m^1, r}(\mu, a, b, \phi, \omega). \end{aligned} \tag{16.47}$$

10. Call Rate Predictions

10.1 Introduction

Consider ideas and models that supplement the ones described in chapter 15. The call rate depends on many factors (see Figures 16.1 and 16.2). There-

fore, one applies the extended version of the Auto-Regression-Moving-Average (ARMA) model and software (see Chapter 15).

```
1        10/1/96  Tue       1607    1607     1.2     1.2
0       0       0       0       0       0       0       0
2        10/2/96  Wed       1431    1431     1.1     1.1
0       0       1       0       0       0       0       0
3        10/3/96  Thu       1400    1400     1.1     1.1
0       0       0       0       0       0       0       0
4        10/4/96  Fri       1194    1194     1.1     1.1
0       0       0       0       0       0       0       0
5        10/5/96  Sat        476     476     1.1     1.1
0       0       0       0       0       0       0       0
6        10/7/96  Mon       1553    1553       1       1
0       1       0       0       0       0       0       0
7        10/8/96  Tue       1421    1421       1       1
0       0       0       0       0       0       0       0
8        10/9/96  Wed       1486    1486     1.1     1.1
0       0       0       0       0       0       0       0
9       10/10/96  Thu       1339    1339     1.1     1.1
0       0       0       0       0       0       0       0
10      10/11/96  Fri       1106    1106    0.86    0.86
0       0       0       0       0       0       0       0
```

Figure 16.1. A fragment of the the DATAFILE 'call.data' including the number, the date, the call rate (repeated twice) , the real-valued external factor (repeated twice) and the indicators of eight Boolean external factors.

The results show that one does not improve predictions by including the external factors directly into the ARMA model. A reason is that in the ARMA framework it is difficult to estimate the delay time (see expression (15.22)) and the duration of impacts Special Events (SE)[5]. The additional difficulty is that most of SE are rare (see Figure 16.1). Therefore, columns of date file are filled mostly by zeros. Most of the SE indicated in Figure 16.2 are predictable. For example, one can predict factor 6 (public holidays, traditional celebrations) and factor 7 (last day for ordering) exactly. Other SE can be predicted approximately. The knowledge of future values of external factors helps to predict the main one. However, the present software version cannot use this possibility yet.

[5]While predicting call rates, we call the external factors as "Special Events" (SE).

```
Events
index   code    description

1     Cp        Postage of main catalogue
2     Cr        Members receive main catalogue
3     D      Selecting members for dunning
4     Ep        Postage of extra catalogue
5     Er        Members receive extra catalogue
6     H      Public Holidays, traditional celebrations
7     L      Last day for ordering
8     Op        Postage of Order-reminder
```

Figure 16.2. Eight Boolean external factors related to call rate.

Here, we consider call rates $\lambda(t)$ as a sum of two stochastic functions

$$\lambda(t) = z(t) + \nu(t). \tag{16.48}$$

Often the rate $\lambda(t)$ has several components $\lambda(t) = (\lambda_1(t), ..., \lambda_l(t))$ corresponding to different types of calls. In expression (16.48) $\nu(t)$ denotes a "stationary" component which is described by the ARMA model. A "non-stationary" component is denoted by $z(t)$. This way we separate the stationary part from the non-stationary one.

The theoretical analysis of non-stationary stochastic functions is difficult. Therefore, the separation of non-stationary component is important. The non-stationary component is defined by a local expert, as usual. The expert applies his knowledge while using the previous data and making the future predictions. Thus, we call $z(t)$ as the "expert" component and $\nu(t)$ as the "statistical" one. The estimate of the statistical component $\nu(t)$ is investigated in Chapter 15, considering the ARMA model. An alternative is "Scale" models. They predict the expert component $z(t)$ by estimated scales. The scales express the differences between different SE and different times.

10.2 Call Rate Prediction by Scale Models

The ARMA models considered in Chapter 11 reflect non-stationarity by eliminating unstable parameters applying the structural stabilization techniques. This way some data and some model parameters $a_i,\ b_j$ are eliminated. The estimates of the remaining ones are obtained. Therefore, one may regard ARMA

parameters $a_i,\ b_j$ as some scales, too. These scales reflect the influence of the corresponding data sets by minimization of prediction errors. No expert knowledge is involved.

Now we shall consider the scale models, where the expert opinion is involved. This is done by choosing data sets and scales reflecting the expert opinion. Two versions of scales models are considered.

In the first model it is supposed that the prediction z_i is a product of the present call rate z_{i-1} and some scale s_i

$$z_i = z_{i-1}\ s_i. \tag{16.49}$$

Here z_i is a predicted call rate, called a "prediction." z_{i-1} is an observed call rate, called a "data set." The index $i-1$ defines the data set used to predict z_i. The parameter s_i is the "scale" to predict z_i by the data set z_{i-1}. Expression (16.49) is a time scale model. In this model the scale is estimated as

$$s_i = \frac{z_{p(i)}}{z_{p(i-1)}}. \tag{16.50}$$

In (16.50) the subscript $p(i)$ defines a period which precedes i. The subscript $p(i-1)$ denotes a period preceding $i-1$. The term "preceding" means the nearest previous period of the same type. For example, Saturdays, Sundays, holidays, Christmas weeks, sporting events, days of marketing messages, e.t.c., all are different types.

In the second model

$$z_i = z_{p(i)}\ s_{p_i}. \tag{16.51}$$

Here the scale is estimated as

$$s_{p(i)} = \frac{z_{i-1}}{z_{p(i-1)}}. \tag{16.52}$$

Both expressions(16.49) and (16.51) predicts the same call rates. That means that one obtains the same results by the event scales (16.50) and the time scales (16.52). Only the interpretation differs.

In the vector case, expressions (16.49) and (16.51) predict different call rate graphs. One represents these graphs as vectors $z_i = (z_{ij},\ j = 1, ..., J)$. The components z_{ij} denote call rates of different parts j of a period i For example, hours if we predict next day call rates.

The assumption of the first model (16.49) is that the next day graph $z_i = (z_{ij},\ j = 1, ..., J)$ is equal to the present one $z_{i-1} = (z_{ij},\ j = 1, ..., J)$ multiplied by scales s_i

$$s_i = \frac{Z_{p(i)}}{Z_{p(i-1)}}. \tag{16.53}$$

Here $Z_i = 1/J \sum_{j=1}^{J} z_{ij}$ denotes the average call rates of the period i. Expression (16.53) means that the shapes of the next and the present graphs remains the same. Only the scales differ.

The second model (16.51) assumes that the next day graph is equal to the graph of the preceding period $z_{p(i)} = (z_{p(i),j},\ j = 1, ..., J)$ multiplied by scales

$$s_{p(i)} = \frac{Z_{i-1}}{Z_{p(i-1)}}. \tag{16.54}$$

This means that the shapes of the next graph and the preceding one are the same. Only the scales differ. The numerator $Z_{p(i)}$ of scales s_i of the first model depends on the Special Events (SE) of the next period i. We call the first model (16.49) as the event scale model.

The numerator $Z_{p(i)}$ of scales s_i of the first model depends on the Special Events (SE) of the next period i. By definition, the preceding period $p(i)$ depends on the type of next one. The period type is defined by the Special Event (SE) which is active in the period. Examples of SE are Saturdays, Sundays, holidays, Christmas weeks, sporting events, days of marketing messages e.t.c..

The numerator Z_{i-1} of scales $s_{p(i)}$ of the second model depends on the average call rate now $i - 1$. Therefore, we call the second model (16.51) as the time scale model.

Using the event scale model (16.49), the next day event SE defines the next day scale. The shape of the next day graph remains the same as today. Therefore, this model reflects the changes of the graph shapes without delay.

Using the time scale model (16.51), the shape of the next day graph is the same as the shape of the preceding one. Here the graph changes, but with some delay. The interval between the next period and the previous period of the same type defines the delay time.

10.3 Expert Model, Event Scale Version

Definition of Special Events

Here the call rate in the next period (16.49) is the same as in the present one multiplied by the scale s_i. The scale depends on the Special Event (SE) which is expected to be active in the next period. The periods where no special events are active, we consider as periods with "neutral" special events. Scales s_i determine the impact of SE to the call rate z_i of next period.

The scale s_i reflecting the impact of a special event is estimated using empirical data by expressions (16.50), (16.53) and/or directly by expert opinions. The impact of some special events, such as Saturdays, Sundays, holidays, Christmas week e.t.c.. is well defined and instant. One calls them Instant Special Events (ISE).

The impact of some others, such as marketing messages, is delayed. Call rates react to these events with some delay. Therefore, one calls them Delay Special Events (DSE). The starts and the durations of DSE are fixed. The delay times d and the durations τ of DSE impacts are estimated using the observed data and expert knowledge. In some periods several DSE may be active simultaneously. One calls these Multiple Special Event (MSE).

The starts and the durations of ISE and their impacts are equal to starts and durations of the corresponding periods. Beginnings and durations of DSE impacts are not necessarily equal to the beginnings and durations of the periods. Therefore, a special sorts of DSE appear, called Partial Special Events (PSE). In PSE the impact of a special event is active only during some part of the period. The delay times d, the durations τ of the impacts of DSE, and the scales s defining the impact of all the special events, are estimated using the available data and the expert knowledge.

Procedures

We consider three expert procedures and the method of least squares, while defining estimates of scales s_i. The first procedure is the empirical one. The scales s describing the impact of all sorts of events, including the partial and the multiple ones, are estimated by expressions (16.50) or (16.53). These expressions define the relation of the call rates in the pair of periods $p(i-1)$ and $p(i)$. A period $p(i-1)$ precedes the present period $i-1$. Denote by $p(i)$ a

period that precedes the next one, denoted i. The numerator $z_{p(i)}$ is the call rate[6] of the period preceding the next one. The denominator $z_{p(i-1)}$ is the call rate of the period preceding the present one.

Using the empirical procedure, partial and multiple special events are considered as different SE. Here scales s are defined separately for each of them. Therefore, the empirical procedure is convenient, if the data is available for all types of periods, including the partial and multiple ones. In such a case, estimate of the delays d and durations τ of DSE is not needed. It is supposed that an identical situation[7] is in the available data. One needs large data sets for that.

The second direct procedure is the subjective one. All unknown parameters of the expert model, such as the scales s, the delay times d and the duration times τ, are defined by the expert opinion. This is a reasonable way to start a new system, when no data is available.

The third direct procedure is a mixture of the empirical and the subjective ones. We use empirical estimates, if the corresponding data is available[8]. Otherwise, one uses subjective estimates. Using the method of least squares, we search for such scales s, such delay and duration times d, t that minimize the sum of squared deviations. The deviations are differences between the call rates predicted by the expert model and the observed call rates z_{ij}. All four procedures are applied to different times: hours, days, weeks and seasons. First, we consider the scalar case.

Scalar Prediction

In the scalar case, one predicts a single number, the average call rate of the next period. In the formal terms, one considers the call rate function z_i as a sample of some non-stationary stochastic process. Four different periods; hours, days, weeks, and seasons, are considered independently. Let us start by describing hours.

[6] In vector predictions that is an average call rate $Z_{p(i)}$.

[7] That means, there exist periods in the data file, with the identical combinations of impacts. Besides, these combinations are the same as that of the new and the present periods.

[8] If a pair of periods, identical to the pair of the present and the next one, can be found.

Hourly Prediction. Using the event scale model, a call rate z_i of the next hour i is expressed this way

$$z_i = z_{i-1}\ s_i. \tag{16.55}$$

Here z_{i-1} is the present call rate. s_i is a scale of the next hour i. The hourly scales s_i differ depending on the impact of SE felt at the day i. For example, working days, weekends, Christmas days, marketing messages days, e.t.c.

$$s_i = \frac{z_{p(i)}}{z_{p(i-1)}}. \tag{16.56}$$

Here $p(i)$ is a subscript of the hour preceding the next one. $p(i-1)$ is a subscript of the hour preceding the present one. Following examples illustrate what one means by the term "preceding."

Examples of Multiple Special Events (MSE). Suppose that the impact of the special event "marketing messages send" starts at the next hour i, that is the second one of a working day. Assume that the present hour $i-1$ is the first one of a working day. Then

$$s_i = \frac{z_{p(i)}}{z_{p(i-1)}}. \tag{16.57}$$

Here $p(i)$ is the subscript of the most recent previous second hour of a working day with the same impact SE. $p(i-1)$ is the subscript of the most recent previous first hour of a working day with no SE. In this example, we consider a multiple SE as two single SE. The first single SE means the first working hour without the impact of marketing messages. The second single SE defines the second working hour with the impact of marketing messages.

Another way is to consider the impacts of those two SE separately. Then one defines the scale s_i as a product of two scales

$$s_i = s_i^1\ s_i^2. \tag{16.58}$$

Here

$$s_i^1 = \frac{z_{p^1(i)}}{z_{p^1(i-1)}}, \tag{16.59}$$

where $p^1(i)$ is the index of the first hour and $p^1(i-1)$ is the index of the second hour. Both indices denote the previous working day.

$$s_i^2 = \frac{z_{p^2(i)}}{z_{p^2(i-1)}}, \tag{16.60}$$

where $p^2(i)$ is the index of the most recent hour with "marketing" and $p^1(i-1)$ is the index the most recent hour with no "marketing." Expression (16.58) needs less data. However, it is based on the assumption that scales are multiplicative. Expression (16.58) involves expert opinion by assuming multiplicativity of scales. Expression (16.57) is an example of empirical approach; multiple SE are considered as different special events.

Examples of Partial Special Events (PSE). The only difference from the previous example is that the impact of "marketing messages" starts at the 20-th minute of the next hour i. Supose that PSE is a special sort of SE with the scale

$$s_i = \frac{z_{p(i)}}{z_{p(i-1)}}. \tag{16.61}$$

Here $p(i)$ is the subscript of the most recent previous day when the "marketing" starts at the 20-th minute of second working hour. $p(i-1)$ is the subscript of the most recent previous first hour of a working day with no marketing. Another way is to consider the impact of PSE by setting the scale s_i of marketing messages, in proportion to the duration of their impact

$$s_i^{20} = 2/3\ s_i, \tag{16.62}$$

where

$$s_i = \frac{z_{p(i)}}{z_{p(i-1)}}. \tag{16.63}$$

Here $p(i)$ is the subscript of the most recent previous day when the "marketing" starts at the second working hour. $p(i-1)$ is the subscript of the most recent previous first hour of a working day with no marketing.

Expression (16.62) needs less data. However, it is based on the assumption that scales are proportional to the impact duration. Expression (16.62) involves expert opinion by assuming the proportionality of scales. Expression (16.61) is an example of the empirical approach where a PSE is considered as a different special event.

Method of Least Squares

Denote by $\psi_i(s, d, \tau)$ the call rate prediction by an expert model with fixed parameters s, d, τ. One predicts call rates of the next period i, using data up to the period $i-1$. The parameters of the model include scales s and delay and

duration times d, τ. We minimize the sum

$$\min_{s,d,\tau} \sum_{i=T_0}^{T} (z_i - \psi_i(s,d,\tau))^2. \tag{16.64}$$

Here T is the end and T_0 is the beginning of the "learning" set.

One minimizes (16.64) by various global and local optimization methods. That means that estimates of parameters of expert scale models are obtained in the same way as estimates of ARMA parameters (see Section 15). The difference of (16.64), is that parameters s, d, τ of the expert model EM are optimized, instead of ARMA parameters b. The optimization problem (16.64) is very difficult. Here, the number of variables is great and the objective function is multi-modal.

Vector Prediction, Event Scale Version

One often predicts the graph of call rates of the next perid. The graph is represented as call rates of different parts of the next period. These call rates are components of some vector. That justifies the term "vector prediction"[9].

The only difference from the scalar prediction is that we predict not a single call rate z_i of the next period i, but some vector $z_i = (z_{ij},\ j = 1, ..., J)$. Here the component z_{ij} definines the call rate of the part j of the period i. First, we consider daily predictions, where index i denotes days and the index j denotes hours.

Daily Vector Prediction. We predict the hourly call rates for the next day. The predicted call rate z_{ij} of the hour j of the day i is

$$z_{ij} = z_{i-1,j}\ s_i. \tag{16.65}$$

Here $z_{i-1,j}$ is the call rate of the hour j of the present day $i-1$. s_i is a scale of the next day i. The scales s_i depend on the impact of SE felt at the day i. For example, working day, Saturday, Sunday, Christmas day, marketing day, e.t.c.

$$s_i = \frac{Z_{p(i)}}{Z_{p(i-1)}}. \tag{16.66}$$

[9] In section 10 the terms "one-step" and "multi-step" were used instead of "scalar" and "vector."

Denote by $p(i)$ the subscript of the nearest previous day, similar to the next one. Denote by $p(i-1)$ the subscript of the nearest previous day, similar to the present one. The meaning of "nearest previous day similar to" is illustrated by following examples.

Examples of MSE. Suppose that the impact of the SE "marketing messages send" starts at the next day i. That is Tuesday. The present day $i-1$ is Monday with no additional special events. Then the scale

$$s_i = \frac{Z_{p(i)}}{Z_{p(i-1)}}. \tag{16.67}$$

Here $p(i)$ is the subscript of the most recent previous Tuesday with the same SE. $p(i-1)$ is the subscript of the most recent previous Monday with no SE. In this example we considered a multiple SE as two single SE: Monday without marketing, and Tuesday with marketing.

Another way is to consider the impacts of those two SE by defining the scale s_i as a product of two scales.

$$s_i = s_i^1 \, s_i^2 \tag{16.68}$$

Here

$$s_i^1 = \frac{Z_{p^1(i)}}{Z_{p^1(i-1)}}, \tag{16.69}$$

where $p^1(i)$ is the index of Monday, $p^1(i-1)$ is the index of Tuesday of the previous week, and

$$s_i^2 = \frac{Z_{p^2(i)}}{Z z_{p^2(i-1)}}. \tag{16.70}$$

$p^2(i)$ is the index of the most recent day with "marketing." Index $p^1(i-1)$ denotes the most recent day with no "marketing." Expression (16.67) is an example of empirical approach. Here multiple SE are regarded as different special events. Expression (16.68) needs less data but is based on an expert assumption that scales are multiplicative.

Examples of PSE. One feels the impact of the special event, called "marketing," next day i on the sixth working hour. That is the only difference from the previous example. Let us to consider PSE as another SE. Then

$$s_i = \frac{Z_{p(i)}}{Z_{p(i-1)}}. \tag{16.71}$$

Here $p(i)$ is the subscript of the most recent previous week when the "marketing" starts Tuesday at the sixth working hour. $p(i-1)$ is the subscript of the most recent previous Monday with no marketing.

Another way is to regard PSE as a reduced SE. Then, the scale s_i is proportional to the impact duration

$$s_i^6 = 1/4\ s_i. \tag{16.72}$$

Here

$$s_i = \frac{Z_{p(i)}}{Z z_{p(i-1)}}, \tag{16.73}$$

$p(i)$ is the subscript of the most recent previous week, such that "marketing" starts Tuesday on the sixth hour, and $p(i-1)$ is the subscript of the most recent previous Monday with no marketing.

Expression (16.71) is an example of empirical approach when a PSE is regarded as a different special event. Expression (16.72) needs less data. However, it is based on an expert assumption that scales are proportional to the impact duration.

Method of Least Squares

Denote by $\psi_{ij}(s,d,\tau)$ the call rate prediction by an expert model with fixed parameters s,d,τ. One predicts call rates of the part j of the next period i using data up to the period $i-1$. The parameters of the expert model include scales s and delay and duration times d,τ. All special events, including DSE and PSE, are regarded. We minimize the sum

$$\min_{s,d,\tau} \sum_{i=T_0}^{T} \sum_{j=1}^{J} (z_{ij} - \psi_{ij}(s,d,\tau))^2. \tag{16.74}$$

Here T is the end and T_0 is the beginning of the "learning" set.

Define the expression (16.74) as a function 'fi' in the file 'fi.C'. Then one minimizes (16.74) by various global and local optimization methods. That means that estimates of parameters of expert scale models are obtained in the same way as estimates of ARMA parameters (see Section 15).

10.4 Time Scale Version, Vector Prediction

Scalar predictions by time scales are identical to those using event scales. Vector predictions by time scale models (16.51) are different from those using event scale models (16.49). Here the shape $z_I = (z_{ij},\ j = 1, ..., J)$ of the next day graph remains the same as that on the preceding day $p(i)$. Only scales s_{i-1} change. This follows from (16.52). Scales s_{i-1} of the present period $i-1$ reflect changes of average call rates between the present period and the preceding one. The scales are estimated using empirical data and/or expert opinion. The definitions and examples of all the special events are the same as in the Event Scale case (see Section 10.3).

Procedures

We shall consider four procedures to obtain estimates: empirical, subjective, mixed, and least squares. The estimates are similar to those in the Event Scale Model (see Section 10.3). Only scales and preceding periods are different. One predicts a vector $z_i = (z_{ij},\ j = 1, ..., J)$. Here z_{ij} means the call rate of the part j of the period i. First, we consider daily predictions. Then the index i denotes days, and the index j denotes hours.

Daily Vector Prediction

We predict the hourly call rates of the next day. The predicted call rate z_{ij} of the hour j of the next day i is

$$z_{ij} = z_{p(i)j}\ s_{i-1}. \tag{16.75}$$

Here $z_{p(i)j}$ is the call rate of the hour j of the day preceding the next one. s_{i-1} is a scale of today $i-1$. Scales s_{i-1} depend on the impact of SE felt at the day i. For example, working day, Saturday, Sunday, Christmas day, marketing day, e.t.c.

$$s_i = \frac{Z_{i-1}}{Z_{p(i-1)}}. \tag{16.76}$$

Here $p(i-1)$ is a subscript of the day preceding the present one. The term "preceding" is illustrated by the following examples.

Examples of MSE. Suppose that the impact of the SE "marketing messages send" begins and ends today $i-1$, which is Monday. The next day i is Tuesday

with no additional special events. Then the scale

$$s_i = \frac{Z_{i-1)}}{Z_{p(i-1)}}. \tag{16.77}$$

The index $p(i-1)$ denotes the day, which precedes the present one. Here, we regard a multiple SE as two single SE. First single SE is Monday with marketing. The second single SE is Tuesday without marketing.

Another way is to regard the impacts of those two SE by defining a scale s_{i-1} as a product of two scales.

$$s_{i-1} = s^1_{i-1}\, s^2_{i-1}. \tag{16.78}$$

Here

$$s^1_{i-1} = \frac{z_{p^1(i-1)}}{z_{p(p^1(i-1))}}. \tag{16.79}$$

$p^1(i-1)$ is the index of the day preceding Monday without marketing. $p(p^1(i-1))$ is the index of the day preceding $p^1(i-1)$.

$$s^2_i = \frac{z_{p^2(i-1)}}{z_{p(p^2(i-1))}}. \tag{16.80}$$

$p^2(i-1)$ is the index of the day preceding the day with marketing. $(p(p^2(i-1))$ is the index of the day preceding $p^2(i-1)$.

Expression (16.77) is an example of empirical approach when multiple SE are regarded as different special events. Expression (16.78) needs less data but is based on the assumption that scales are multiplicative.

Examples of PSE. Here, the impact of "marketing" starts at the second working hour of the present Monday $i-1$. Regarding PSE as another SE the scale is

$$s_{i-1} = \frac{z_{i-1}}{z_{p(i-1)}}. \tag{16.81}$$

Here $p(i-1)$ is the index of the day preceding the present one.

Another way is to regard the PSE as a reduced SE. Then the scale s_{i-1} is proportional to the impact duration

$$s^2_{i-1} = 1/4\; s_{i-1}. \tag{16.82}$$

Here

$$s_{i-1} = \frac{z_{p^3(i-1)}}{z_{p(p^3(i-1))}}. \tag{16.83}$$

$p^3(i-1)$ is the index of the preceding marketing Monday. $p(p^3(i-1))$ is the index of the day preceding $p^3(i-1)$. Applying expression (16.82) one requires less data. However, this expression is based on the assumption that scales are proportional to the impact duration.

Expression (16.82) involves expert opinion indirectly, by assuming proportional scales. Expression (16.81) is an example of empirical approach when a PSE is regarded as another SE.

Method of Least Squares

Denote by $\psi_{ij}(s, d, \tau)$ the call rate prediction by an expert model with fixed parameters s, d, τ. One predicts call rates of the part j of the next period i using data up to the period $i-1$. The parameters of the expert model include scales s and delay and duration times d, τ. All special events, including DSE and PSE are regarded. We minimize the sum

$$\min_{s,d,\tau} \sum_{i=T_0}^{T} \sum_{j=1}^{J} (z_{ij} - \psi_{ij}(s, d, \tau))^2. \tag{16.84}$$

Here T is the end and T_0 is the beginning of the "learning" set. One minimizes (16.64) by corresponding global and local optimization methods.

Remarks

As usual, expert models predict the weekly and the seasonal graphs. The weekly graphs are given in a form of average calls for each of seven days. In the seasonal graphs, weeks are represented by weekly averages. Often "observed" values of the call rates z_i cannot be defined directly. Then their estimates are used (see Section 5). Missing or uncertain data is replaced by expert estimates. This is important for the first year of operation.

10.5 Time Series Model (ARMA)

Scalar Prediction

Here we consider how to apply the ARMA model to predict the errors ν_i of the expert scale model $\psi_i(s, d, \tau)$, where $\nu_i = \psi_i(s, d, \tau) - z_i$ (Mockus et al., 1997). The formal description of the ARMA model is in Chapter 15. The software is on the web-site (see Section 4). If predicting ν_i we do not know some values of $\nu_s,\ s = i-1, i-2, ...$ then we replace the missing data by the expected values of the unknown ν_s defined recurrently (see Section 10).

ARMA models predict only the difference between the expert models (EM) and the data. Therefore, in the first step, EM models are adapted to the data by defining the right scales. Only then, the parameters $a,\ b$ of ARMA models are optimized. This means that one minimizes a squared difference between the adapted EM model and the observed data. Predicting the data, the results of both EM and AH models are summed up.

Vector Prediction

The vector prediction of EM errors ν_i is performed using multi-step ARMA predictions described in Section 10 "Confidence cones" could be useful for preliminary evaluation of prediction errors. They estimate probable deviations from expected values with some "confidence level," say ninety percent or ninety nine percent. These cones are defined empirically as upper and lower limits of the results obtained by repetition of Monte Carlo procedures (see Section 10).

10.6 Application Examples

In the call center example, both the Expert and ARMA models were tested separately. Figure 15.18 shows daily call rates of this center. Table 15.1 illustrates some results of the ARMA model. Results of the expert model are not available, for publication.

11. Call Center Scheduling

There are "Of-the-Shelf" tools for the scheduling of single-skill agents. The scheduling of multi-skill agents is theoretically possible using Monte Carlo

simulation (see Section 7). However, the simulation time is too large for on-line scheduling. A convenient approximation to multi-skill scheduling is the reduction of multi-skill problems to the single-skill ones. One can do that by representing each multi-skill agent as a "weighted" single-skill one

$$N_{single} = v * N_{multi}. \tag{16.85}$$

Then, one estimates the unknown weight v by minimizing the sum of squared deviations. The deviations are differences between results of approximate single-skill model and the multi-skill one

$$\min_{v} 1/K \sum_{k=1}^{K} (Q_k(\lambda, N_{multi}) - q_k(\lambda, N_{single}))^2. \tag{16.86}$$

Here Q_k denotes the results of the k iteration of a Monte Carlo simulation of the multi-skill system. λ is the call rate. N_{multi} is the number of multi-skill agents. q_k is the result of the k iteration of a Monte Carlo simulation of a single-skill system. N_{single} is the number of single-skill agents, which replace the multi-skill ones.

Another way is to extend the single-skill simulation (see Section 11) to the multi-skill case. Here one needs more computing time to estimate the "optimal" weights v for different call rates. For example, the method $Exkor$ is used to obtain the best values of v (see Section 4).

12. Software Example

12.1 ARMA Model

There are two immlementations of the ARMA model, both are on the web-sites n the "Software Systems". The Java version is marked as $ARMA-GMJ$, the C++ version is denoted by $ARMA - C$.

12.2 Scales Model

To start the program (Jankauskas, 2000), one goes to one of web-sites (see Section 4) and selects the problem "Call Center-SCALES-GMJ" in the section "Global Optimization ". Figure 16.3 shows the operating page. Here is an illustrative example of data file.

```
1.2 1.1 0.72 1.37 0.96
```

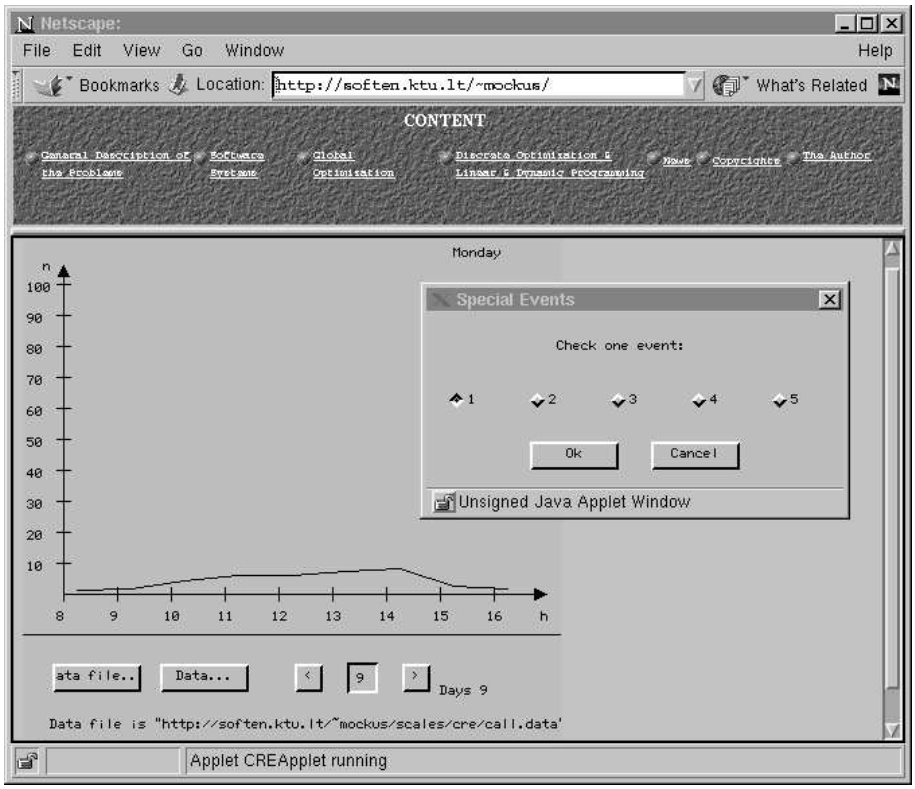


Figure 16.3. Operating page.

```
7
1 1 24.3 26.3 32.5 37.2 40.1 60.2 58.9 56.3 54.2 0 0 0 0 0
2 2 30.3 40.1 40.5 71.7 72.3 70.3 54.1 50.1 45.7 0 0 0 0 0
3 3 32.5 28.9 30.2 35.6 32.5 37.2 40.1 60.2 58.9 0 0 0 0 0
4 4 56.3 54.2 30.3 40.1 46.6 68.9 65.1 54.2 50.1 0 0 0 0 0
5 5 16.3 22.2 27.1 11.8 12.5 10.2 13.1 17.2 16.9 0 0 0 0 0
6 6 18.3 19.2 24.3 40.1 45.5 44.2 30.1 27   20.1 0 0 0 0 0
7 7 2.3  3    7.3  10.2 9.8  12   13.2 4.5  3.1  0 0 0 0 0
```

Chapter 17

OPTIMAL SCHEDULING

1. Introduction

Industrial, financial, commercial or any kinds of activity have at least one common feature: the better organized they are, the higher the profit, the better the quality or the lower the cost. Everyone makes a schedule to organize his or her own life. Making a schedule for any organized activities, one considers a sequence of tasks and a list of resources. Resources include tools, machines, materials, work force e.t.c. Of course, making a schedule for organization is far more difficult than making a schedule for our everyday life.

A failure to make a schedule or devising a wrong schedule can result in delay of a deadline and can cost much of the money (Ahuja et al., 1994, Hajdu, 1997).

First we consider a simple flow-shop scheduling problem. Here the sequence of tools is fixed by technology. One minimizes the make-span, which is the time from the beginning of the first task until the end of the last one. That is a well known simplest model presented just to illustrate how the Bayesian Heuristic Approach (BHA) (Mockus, 2000) could be applied in scheduling problems.

The second example is scheduling of traditional school that is closer to real life tasks. Here the sequence of teaching subjects, regarded as tools, can be changed. One needs to reduce the sum of gaps ("empty" hours) in the teacher schedules. There should be no gaps for students. Different classes are considered as different tasks. The classrooms, including the computer and physics rooms and studies, are the limited resources.

The most difficult example is scheduling of profiled school. Here eleventh and twelfth grade students are choosing , for example, 14 subjects from 64. This means that each student works by his own schedule. We search for the most convenient feasible schedule. The inconveniences are evaluated by penalty points.

All the software is available at the web-site $http://mockus.org/optimum$ and can be run by a standard browser with Java support.

2. Flow-Shop Problem

The flow-shop problem is a simple case of the large and important family of scheduling problems We denote the sets of jobs and machines by J and S. Denote by $\tau_{j,s}$ the duration of operation (j, s). Here $j \in J$ denotes a job and $s \in S$ denotes a machine.

Suppose that the sequence of machines s is fixed for each job j. One machine can do only one job at a time. Several machines cannot do the same job at the same moment.

The objective function v is the make-span T, the time to complete all the jobs.

2.1 Permutation Schedule

The number of feasible decisions for the flow-shop can be very large. One reduces this number, if one considers only the so-called permutation schedules that are subsets of all feasible schedules. The permutation schedule is a schedule with the same job order on all machines. Here one defines a permutation schedule by fixing a sequence of job indices, for example, $1, 2, ..., n$. The schedule is transformed by a single permutation of job indices. As usual, permutation schedules describe the optimal decision well and are easier to implement (Baker, 1974).

2.2 Heuristics

A simple priority rule is defined by the Longer-Job heuristics

$$h_i(j) = \frac{\tau_j - A_i}{A^i - A_i} + a, \tag{17.1}$$

were

$$A_i = \min_j \tau_j,\ A^i = \max_j \tau_j,\ a > 0. \tag{17.2}$$

Here

$$\tau_j = \sum_s \tau_{j,s}$$

where τ_j is the length of the job j.

2.3 Results

Table 17.1 illustrates results of Bayesian Heuristic Approach (BHA) (Mockus, 2000) after 100 iterations using the Longer-Job heuristic (17.1) and different randomization procedures. The objective is a stochastic function. This function defines the shortest make-span found after K repetitions.

In the example, $J^* = S^* = O^* = 10$, where $J^*,\ S^*,\ O^*$ are numbers of jobs, machines, and operations, respectively. Lengths and sequences of operations are generated as random numbers uniformly distributed from zero to ninety-nine. The expectations and standard deviations are estimated by repeating forty times the optimization of a randomly generated problem.

Table 17.1. Results of BHA using Longer-Job Heuristics.

$R = 100$, $K = 1$, $J^* = 10$, $S^* = 10$, and $O^* = 10$					
Algorithm	f_B	d_B	x_0	x_1	x_2
BHA	6.183	0.133	0.283	0.451	0.266
CPLEX	12.234	0.00	—	—	—

In Table 17.1 the symbol f_B denotes a mean, and d_B denotes a standard deviation of the make-span.

"BHA" denotes the Bayesian Heuristic Approach optimizing a "mixture" of heuristic algorithms.
x_0 defines the optimal ratio of Monte Carlo heuristics (algorithm 0) meaning equal probabilities,
x_1 shows the optimal proportion of linear randomized heuristics (algorithm 1) where probabilities are proportional to heuristics h_i,
x_2 is the optimal ratio of greedy heuristics (algorithm 2) when one selects the longest available job.
Note that randomization is applied in two stages. First time the randomization is to select the algorithm $i = 0, 1, 2$ with probabilities x_i. Second time, the randomization is to chose the next job using the algorithm i selected in the first stage.
The objective is to minimize the average make-span that is a stochastic multimodal function of variables $x_i,\ i = 0, 1, 2$ therefore, the Bayesian search is applied (Mockus, 2000). This explains the term Bayesian Heuristic Approach (BHA).

"CPLEX" denotes the the well known general discrete optimization software after twenty hundred iterations (one CPLEX iteration is comparable to a Bayesian observation). Weak results of CPLEX show that the standard MILP technique is not efficient in solving this specific problem of discrete optimization. It is not yet clear how much one improves the results using specifically tailored B&B.

To optimize BHA parameters for a given Flow-Shop problem, one applies the Java optimization system GMJ2 that reduce the average deviation (Greicius and Luksys, 1999).

The optimal proportions of three heuristics: the Monte Carlo, the linear randomization and the "Select-the-Best" are on the output page (see Figure 17.1). The convergence line is there, too. These are results of one hundred iterations by the 'Bayes' method.

3. Traditional School Scheduling

School scheduling is an example of the general scheduling problem. We consider the case where the objective is the number of "empty" hours when the teachers wait for the next scheduled lectures. One calls these hours as "teacher gaps, " or just "gaps" for short. We search for such schedules that reduce the sum of teacher gaps considering the schedules of fifth to twelfth class of the

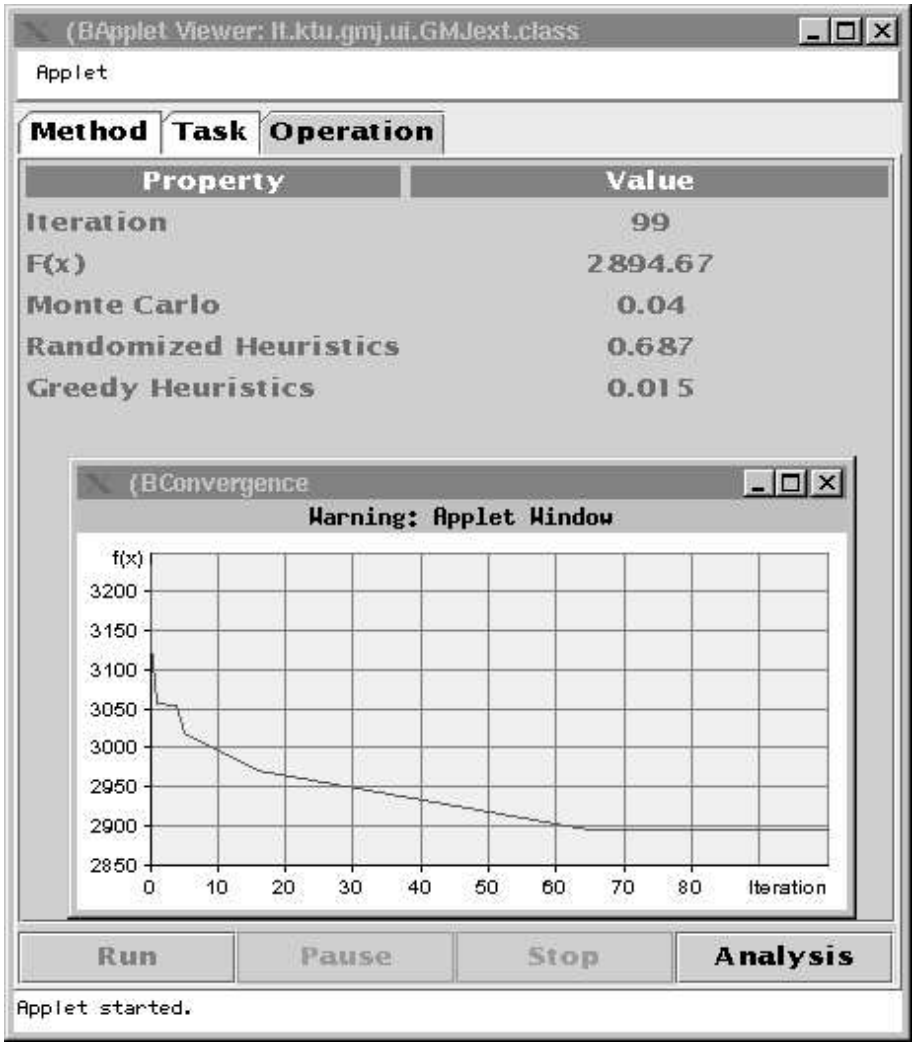


Figure 17.1. Numerical results and convergence line, the 'FlowShop' task.

public high school. Other factors are school-specific and should be included adapting the software to specific schools.

3.1 Constraints

There are five constraints:

- at any given moment a teacher can deliver only one lesson,
- at any given moment a student can take part at only one lesson,
- no "student gaps"are allowed,
- the "double" lectures (the sequence of two lectures of the same subject) are not allowed (there are some exceptions),
- the number of lecture hours per day is limited.

3.2 Data Structure

Figure 17.2 shows a fragment of the existing weakly schedule of some Lithuanian school.

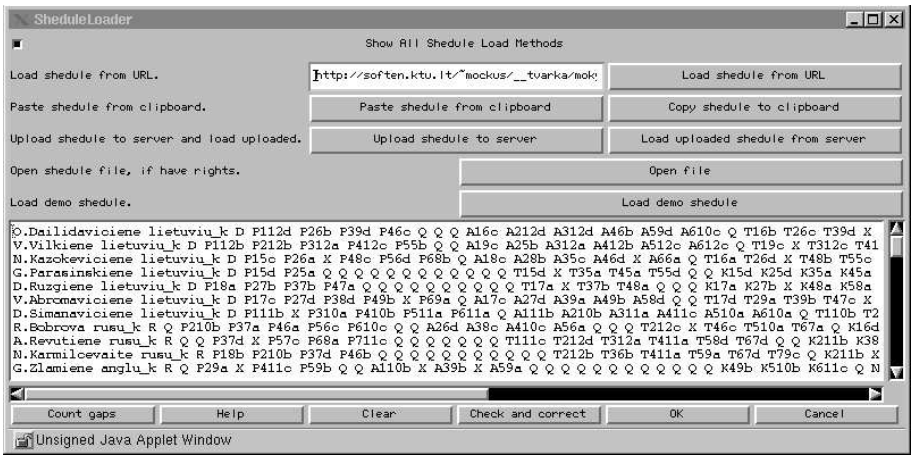


Figure 17.2. Uploaded Schedule-Row Data

- the schedule is defined as a string array $Mokytojai[64][38]$,
- the string $Mokytojai[i][1]$ defines the name of the i-th teacher, for example, $O.Dilidaviciene$, $V.Vilkiene$, e.t.c.
- the string $Mokytojai[i][2]$ defines the subject of the i-th teacher, for example, $lietuviu\_k$ means Lithuanian, $rusu\_k$ means Russian, $anglu\_k$ means English, $matematika$ means Mathematics, $informatika$ means Informatics, e.t.c.

- the string $Mokytojai[i][3]$ defines the code of the classroom of the i-th teacher, for example, 0 means no special study, A means a study for Informatics, e.t.c.
- the sequence of strings $Mokytojai[i][j],\ j = 4, ..., 38$ define the class codes of the i-th teacher for all weak, for example, in the class code $P19c$, the first symbol P means Monday[1], the second symbol 1 defines the first lesson, and the third symbol $9c$ means the class $9c$, teacher gaps are denoted by symbols X, symbols Q denote "open" lessons, a lesson is called "open", if a teacher is free before or after this lesson.

3.3 Permutation and Evaluation Algorithm

The process of checking and correcting illustrate Figure 17.3. The corrected

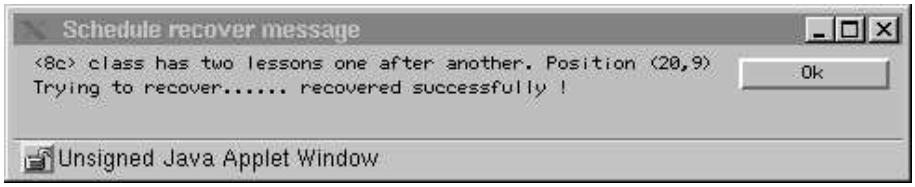


Figure 17.3. Correcting "Double Lecture"

schedule is optimized using the permutation algorithm. The algorithm follows a general pattern of permutation algorithms related to the Bayesian Heuristic Approach (Mockus, 2000)

1 record the current schedule $Mokytojai0$
by reading the data file $Mokytojai[64][38]$,

2 record the current number of teacher gaps[2],

3 set the initial iteration number $it = 0$,

4 set the current iteration number $it := it + 1,\ it < K$,

5 if $it = K$, stop and print the current schedule,

6 set the initial teacher number $i = -1$,

[1]The complete list is P for Monday, A for Tuesday, T for Wednesday, K for Thursday, and N for Friday.
[2]A number of gaps in the current schedule $Mokytojai0$.

7 set the current teacher number $i := i + 1$,

8 if $i > 63$ go to the step 4,

9 generate uniformly distributed random number $\xi \in [0, 1]$,

10 go to the step 7, if $\xi < x$,

11 select a gap of the teacher i,
go to step 7, if there are no gaps,

12 select an open lesson of the i-th teacher,
go to step 7, if there are no open lessons,

13 make a permutation by exchanging the "gap class"[3] with the open one[4],

14 test the feasibility conditions listed in section 3.1,
go to the step 16, if the permutation is feasible,

15 restore the feasibility by restoring the teacher gap and go to the step 7,

16 count the teacher gaps in the permuted schedule,

17 compare the number of the teacher gaps in the current schedule with the permuted one,

18 replace the current schedule by the permuted one,
if the current number of teacher gaps is less then that in the permuted schedule,
go to step 7

The algorithm is very simple and natural. It mimics, in a sense, the usual ways to improve existing schedules. However, replacing a "gap" class by the open one, defined in the step thirteen, a new gap for another teacher is opened, as usual. The algorithm does not satisfy the convergence conditions (Mockus et al., 1997). To satisfy these conditions the probability to reach any schedule should be positive.

3.4 Running Software

The GMJ1 and GMJ2 systems described in (Mockus, 2000) are used in (Perlibakas, 1999) for optimization of BHA parameters following these steps:

[3] A gap class is the class in the teacher gap.
[4] An open class is the first or the last class for this teacher.

- select the method and its parameters,
- select the task 'Tvarka' and its parameters including the iteration number K, the lower and upper bounds for the randomization parameter x, the default value of x.

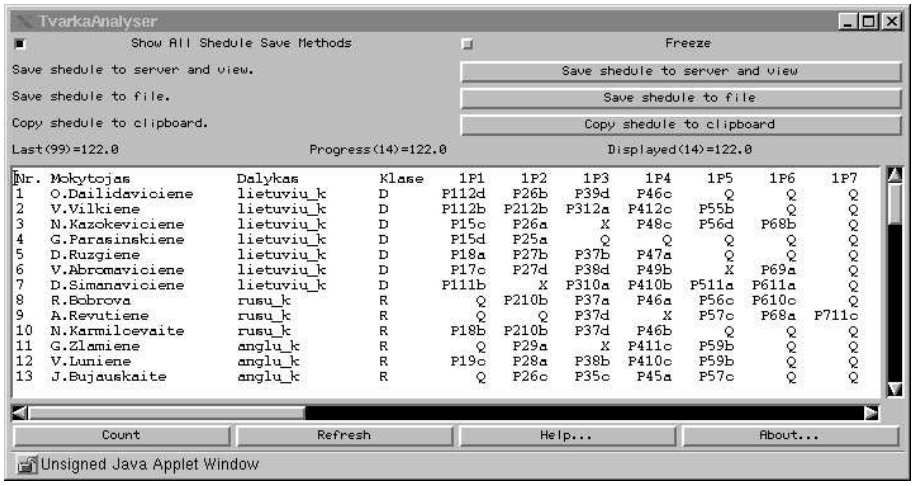


Figure 17.4. Fragment of the Optimized Schedule

To see the optimal schedule click the button $TvarkaAnalyser$ (see Figure 17.5). The software is available at the web-site $http://mockus.org/optimum$,

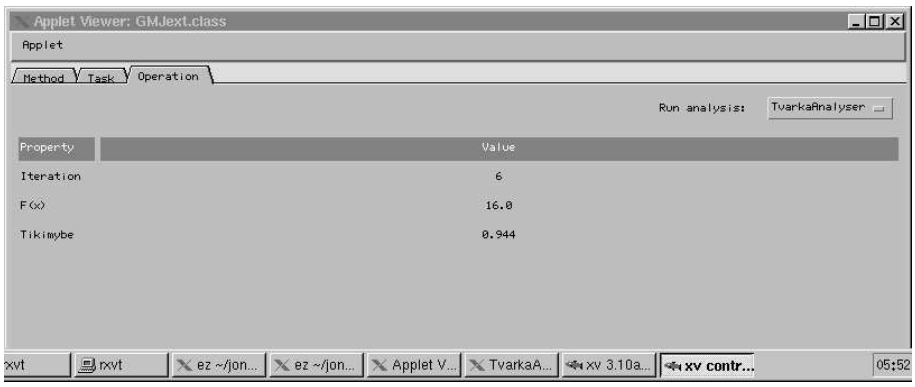


Figure 17.5. Results of GMJ1.

the section "Discrete Optimization", examples "School-COMPLETE" and can be run by a browser with Java support.

4. Profiled Schools

4.1 Introduction

The algorithm described in the previous section is for scheduling of traditional schools. In these schools the study schedules are fixed for each grade.

Application of this algorithm is more difficult, if classes are divided depending on subjects of choice. For example, some students may choose religion, some others ethics, e.t.c. Here the number of "classes" is greater because the schedules of divided classes of the same grade are different. In addition, classes are divided for some subjects and united for others, as usual.

In so-called "Profiled Schools" the students of eleventh and twelfth year select, for example, fourteen subjects from sixty available. That means different individual schedules for most of the students. There are no stable student classes anymore. They are replaced by changing "interest groups". The changes happen each hour.

Personal choices of eleventh and twelfth grade students are defined as sets of subjects. Sequences of these subjects and the corresponding class-rooms are defined by the general school schedule.

In this case a new approach is needed for optimal scheduling. An example is the commercial system $MIMOSA$. The $MIMOSA$ helps to produce feasible schedules by performing corrections, for example, by closing some gaps, and informing about violated constraints and inconveniences. About 50% of the schedules produced by $MIMOSA$ depends on the human scheduler operating $MIMOSA$ system. Therefore, comparison with other systems is very difficult. In this sense, the $MIMOSA$ and other similar systems can be regarded mainly as "Support Systems."

The objective of this research is to design workable optimization system for profiled school scheduling. In the genuine optimization system a human operator specifies just general objectives and constraints. The schedules, both general and personal, are produced automatically by an optimization system. These schedules may be corrected by an operator considering some additional factors not included in the general objectives and constraints. The human oper-

ator can influence the outcome of optimization by choosing an initial schedules, too [5].

4.2 Schedule Representation

A way to define the complete schedule of profiled school is to represent it as a four-dimensional binary array

$$schedc[M][V][G][K]. \tag{17.3}$$

Here
M is a number of teachers[6],
V denotes the total number of weekly working hours,
G is a number of different interest groups,
K is a number of school rooms,

$$schedc[i][j][k][l] = 1 \tag{17.4}$$

means that the teacher i at the hour j teaches the interest group k in the class room l,

$$schedc[i][j][k][l] = 0, \tag{17.5}$$

means no lesson.

The general schedule is represented as an three-dimensional binary array

$$schedg[M][V][K]. \tag{17.6}$$

To represent the general schedule in two dimensions the two-dimensional array is used

$$schedg2[M][VK]. \tag{17.7}$$

where VK is a string defining both the hour and the class room (see for example 17.2).

Personal schedules are defined for each interest group k as three-dimensional binary arrays

$$sched_k[M][V][K]. \tag{17.8}$$

[5]The optimization results depend on the initial schedules because there is no possibility to obtain exact solutions in a limited time.
[6]If a teacher covers several subjects then he/she is represented as several different "virtual" teachers, feasibility conditions prevents cases with two or more lessons by the same "physical" teachers at the same time.

To represent this schedule in two dimensions the two-dimensional array is used

$$sched2_k[M][VK]. \tag{17.9}$$

The binary representation is simple and convenient for understanding and defining constraints but requires large memory and is not convenient for schedules of real profiled schools.

4.3 Objective Function

There are no schedules that satisfy all restrictions and personal preferences because they contradict each other as usual. For example, the ministry of education defines a number of rules to be observed, the school teachers and students both prefer schedules without gaps, etc. Thus, the objective is to find the best compromise of conflicting interests.

A compromise solution is reached by defining penalties for violation of constraints and disregarding inconveniences. We search for such schedule that minimizes the total penalty function. Only the "physical" constraint may not be violated:
a person cannot be in two places at the same time.
We assume, that all other constraints may be violated, at a price. Both normative and convenience constraints are considered in this research.

4.4 Normative Constraints

The standard restrictions are

1 for each group k the total number of weekly hours $v_{sk} \leq V_{sk} < 168$,

2 for each group k the total number of daily hours $v_{dk} \leq V_{dk} < 24$,
for eleventh and twelfth grades $V_{sk} < 5V_{sd}$,
for the rest $V_{sk} = 5V_{sd}$

3 for each teacher i the total number of weekly hours is V_i,

4 for each group k the total number of weekly hours for a subject n[7] is V_{kn},

5 not more than one lesson at a time in a class [8],

[7] If the subject n is covered by just one teacher i, then $n = 1$.
[8] This is a normative constraint, because several lessons may be performed in one class room, physically.

6 for some subjects and groups "double" lessons [9] are needed,

7 for some subjects and groups "double" lessons are forbidden,

8 for some subjects and groups "double" lessons are allowed,

9 no gaps for students from the first to tenth grade.

4.5 Convenience Constraints

The usual factors of inconvenience are

1 teacher gaps

2 student gaps (for eleventh and twelfth grades)

3 inconvenient hours

4 inconvenient days

5 inconvenient sequences of lessons

4.6 Penalty Function

Normative Penalties

Formally, the normative constraints cannot be violated. Practically, some minor violations could be tolerated, if this improves other parameters considerably. Thus the normative penalties c_r must be high.

$$C_n = \sum_r c_r N_r \tag{17.10}$$

where
c_r is the penalty for violation of the normative r,
N_r is the number of corresponding violations.
In this case $r = 1, .., 9$.
The actual numbers c_r are expert estimates and depends on general situation of the school. Therefore, they are parameters defined by users using graphical interfaces, as usual.

[9] This means two lessons in a raw.

Penalties of Inconvenience

Penalties of inconvenience are defined by users for each inconvenience factor:

1 c_i is the penalty for the teacher i gap,

2 c_k is the penalty for the group k gap,

3 c_{ji} is the penalty of the hour j that is inconvenient for the teacher i

4 c_{li} is the penalty of the day l that is inconvenient for the teacher i

5 c_{jk} is the penalty of the hour j that is inconvenient for the group k

6 c_s is the penalty for inconvenient sequence s of lessons,

The general inconvenience penalty

$$C_c = \sum_i c_i L_i + \sum_k c_k L_k + \sum_i \sum_j c_{ji} L_i^j + \sum_i \sum_l c_{li} L_i^l + \sum_k \sum_j c_{jk} L_k^j + \sum_s c_s L_s \tag{17.11}$$

where
L_i is the number of gaps of teacher i,
L_k is the number of gaps of group k,
L_i^j is the number of hours j that are inconvenient

for the teacher i,
L_i^l is the number of days l that are inconvenient for the teacher i,
L_k^j is the number of hours that are inconvenient for

the group k,
L_s is the number of inconvenient sequences of lessons.

The objective function is the total penalty

$$C = C_n + C_c \tag{17.12}$$

Then the optimization problem is

$$\min_{\tau \in \Theta} C(\tau), \tag{17.13}$$

where $C(\tau)$ is the total penalty of some schedule τ,
Θ is the set of schedules satisfying the physical constraints. The penalties $C(\tau)$ depend on expert evaluations, therefore we regard them as heuristics.

4.7 Defining the Initial Schedule

In some cases the initial schedule is defined by users. Otherwise it is build by some greedy heuristics.

Using the Greedy Heuristics techniques one builds the schedule from scratch. Here the results depend on the sequence of "building" operations. This way, one obtains a schedule that is convenient for the

first teacher but may be very inconvenient for the last teacher, e.t.c.

As usual, greedy heuristics define priorities. That means that subjects, classes and persons that are higher on the priority list obtain better schedules. The schedule build by pure greedy heuristics often does not satisfy some important restrictions. Therefore randomization is introduced. It helps to build better initial schedule by random mixing of schedule lines and so extending the range of search.

4.8 Optimization by Permutations

The initial schedule is improved by permutations. The best obtained schedule is recorded after each iteration. Changes to worse schedules are performed with some probabilities. That is for improving convergence conditions.

A natural first step while trying to provide convergence is the Simulated Annealing (SA):
One moves from the current schedule i to the permuted schedule $i+1$ with probability

$$r_{i+1} = \begin{cases} e^{\frac{-h_{i+1}}{x/\ln(1+N)}}, & \text{if } h_{i+1} > 0, \\ 1, & \text{otherwise.} \end{cases} \tag{17.14}$$

Here N is the iteration number and x is the "initial temperature." The logarithmic "cooling schedule" $\ln(1+N)$ follows from convergence conditions (Cohn and Fielding, 1999).

The difference from the traditional SA is that we optimize the parameter x for some fixed number of iterations $N = K$. In this case the cooling schedule should be optimized, too. A natural way to do it is by introducing the second

parameter x_2. This transforms expression (17.14)

$$r_{i+1} = \begin{cases} e^{\frac{-h_{i+1}}{x_1/\ln(1+x_2 N)}}, & \text{if } h_{i+1} > 0, \\ 1, & \text{otherwise,} \end{cases} \tag{17.15}$$

where $x_1 \geq 0$ defines an "initial temperature" of SA and $x_s \geq 0$ describes its "cooling rate".

SA and other discrete optimization techniques are explained in (Mockus, 2000) by the Knapsack example .

4.9 Software Example

The profiled school scheduling software implements permutations similar to these of the traditional school considered in (Mockus, 2000). That is only similarity. Important new ideas are developed and implemented while considering profiled school scheduling.

The fist one is that the schedule should be evaluated including both objective and subjective factors in a balanced way. The theoretical framework of this is the Pareto optimum. Simlest way to obtain Pareto optimal schedule is by assigning some penalty points for deviations from the desirable or feasible conditions. The total penalty is minimized. In such a case the penalty points represent subjective opinion of people responsible for schedules. The conditions reflects objective legal and physical factors.

The second new idea is optimization of heuristics by selecting their best parameters. That is achieved in three stages. In the first stage the same trivial permutation heuristics is used as in the traditional schedule. That works if one starts from nearly optimal schedule what is true in traditional schools as usual. In the profiled schools some "globalization" of permutation heuristics is needed.

That is achieved in the second stage by Simulated Annealing (SA) schedules applied with same fixed parameters such as initial "temperature" and the "cooling" rate The third fixed parameter is the probability to "miss" some teacher waiting in the "improvement" queue. In this case one accepts with some small probability schedules with higher penalties so improving convergence. It is easy to notice that the results of SA depend on all of three arbitrarily fixed parameters.

In the third stage these parameters are optimized using methods of stochastic global optimization developed in the framework of the Bayesian Heuristic

Approach (BHA) (Mockus, 2000). Each stage can be involved separately, that makes comparisons of their results much simpler.

There are two software versions: applet and servlet. The graphics of the applet version look better as usual, because graphical programs run at the user site. The servlet version runs using server resources. That is an advantage if user computing resources are limited.

Applet version

This version implements only the first stage where the total penalty function is minimized locally using the simplest permutation heuristics That improves the initial schedules as usual. The advanced SA and BHA methods are implemented in the servlet version because servlet mode is more convenient providing schools with servers computing resources and better facilities for data exchange.

To run the applet one visits the web-site $http://mockus.org/optimum$, enters the problem named "School-PROFILED" in the section "Discrete Optimization" and starts $applet.html$ by a browser that supports Java.

User Data

The data file defines the list of subjects and the students choices. File format is txt, one record isl for one subject.

Structure of the record is as follows:
1. Number of grade.
2. Subject code.
3. Teacher code.
4. Number of classrooms.
5. Number of classes per week.
6. List of attending students.

```
Records are  not longer then 11 symbols.
Fields are separated by the symbol |.
The end of the line is marked by the symbol |^
 For example,
11AngLg | ZenMar | 30 | 3 | 11LeonUla | 11KuprMin |^
12MatLg | BirCer | 21 | 4 | 12GerdAbis | 12GirdSaru |
```

```
12GedLag |^
```

Here the number 11 denotes eleventh grade, the code $AngLg$ means the Lg version of English, $ZenMar$ is the abbreviation of teacher name, 30 is the classroom number, 3 is the number of weekly lessons, $11LeonUla$ defines a name of student of eleventh grade, e.t.c.
A fragment of user data file is in Figure 17.6.
Figure 17.7 shows the opening window.

```
11BiolprdMD|MDbio|K35|1|11AleksVR|11BinkyGR|11DociûlR|11LoèysÞR|11ÈalneVR|11ÈepënRR|A
11BraizybNL|NLmat|K27|1|11BabicDR|11BarauRR|11BinkyGR|11JokubAR|11JuozaMR|11JusaiVR|11
11Chemij3AR|ARche|K33|2|11AleksVR|11BelevlH|11KamarAH|11LaukaLR|11LoèysÞR|11MilþiEH|11
11Lietuv3Gl|Gllie|K25|5|11AleksVR|11BarauRR|11BinkyGR|11DociûlR|11JokubAR|11JuozaMR|11Jur
11Dizstu1JN|JNdai|K18|1|11BabicDR|11BagdoER|11BarauRR|11JankûlR|11JuozaMR|11JurevlR|11Jus
11Dizstu2JN|JNdai|K18|1|11AleksVR|11BelevlH|11BinkyGR|11ButkuSH|11ChlynEH|11DociûlR|11Jak
11Dizteo1JN|JNdai|K18|1|11BabicDR|11BagdoER|11BarauRR|11JankûlR|11JuozaMR|11JurevlR|11Jus
11Etikaa2KJ|KJeti|K10|1|11CinytLH|11JakniRH|11JanuðlH|11JusaiVH|11KarveGH|11KauðaGH|11Matu
11FilosofCL|CList|K26|1|11KamarAH|11KarveGH|11LemanlH|11PalakKH|11SadauNH|11SteliRH|11Sv
11Fizika1LL|LLfiz|K36|3|11DociûlR|11JankûlR|11JokubAR|11KenulDR|11KuzmiVR|11LaukaLR|11Sir
11Fizika2KS|KSfiz|K36|3|11AleksVR|11BagdoER|11ButkuSH|11JakniRH|11JuozaMR|11JurevlR|11Jusa
11Fizika3KS|KSfiz|K36|2|11BabicDR|11BarauRR|11BinkyGR|11LemanlH|11LubaiSR|11NenenAR|11P
11GeogPolUl|Ulgeo|K24|1|11BarauRR|11BinkyGR|11JuozaMR|11LoèysÞR|11LubaiSR|11UleckMR|11Z
11Inform1PV|PVinf|K15|2|11BagdoER|11BarauRR|11JankûlR|11JuozaMR|11JurevlR|11JusaiVR|11Ker
```

Figure 17.6. User Data

User data is uploaded by CGI interface clicking the $Browse$ button. The scheduling starts by the $Read\ data$ button.
User preferences are expressed defining restrictions and preferences and assigning penalty points for violating these restrictions and preferences.

Initial Schedule

The initial schedule is built by priority rules and randomization. For example, continuous lessions are included first, starting from Monday. Then all the lessions are listed that can be performed at the same time without violation of physical restrictions and day-off requests. One lession per day is scheduled. Both student and teacher gaps are ignored while building the initial schedule. The main difficulty is to list all the subjects without exceeding the default limit of 9 daily hours. To achieve this the random mixing of schedule lines is used up to 100 times. After that the limit is increased by 1 (first time from the default 9 to 10 hours per day). The initial schedule is completed by moving the largest classes to early hours.

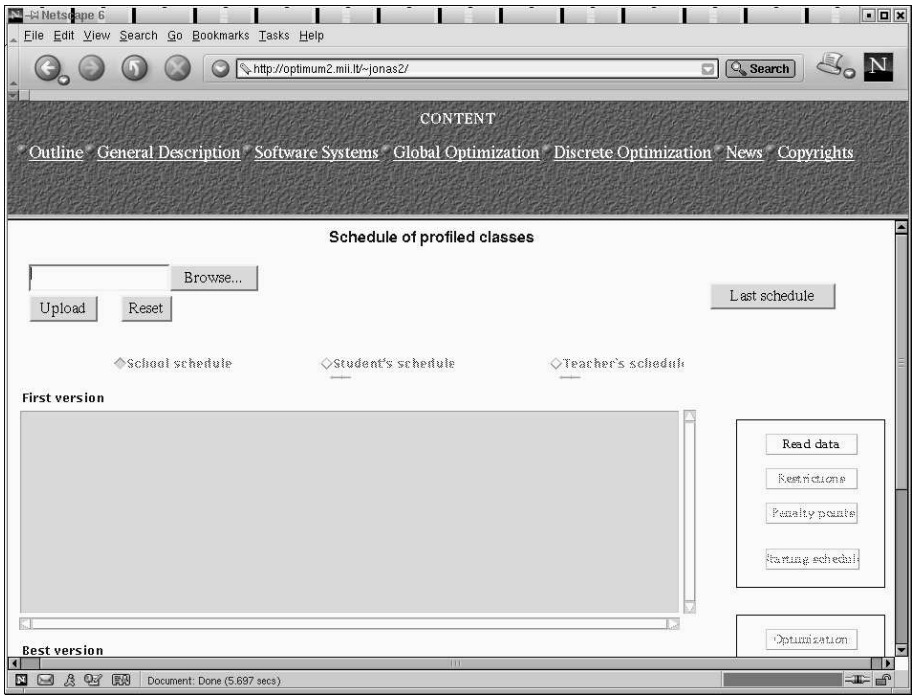


Figure 17.7. Opening Window

Optimization by Permutations

First a window opens for starting optimization and defining randomization parameter x. The permutations are similar to those in traditional school. The difference is that both the teacher and the student gaps are considered and the probability x to pass a person i is fixed. This reduces optimization possibilities as compared with the traditional school scheduling software. Note that the advanced servlet version includes optimization of all the heuristic parameters.

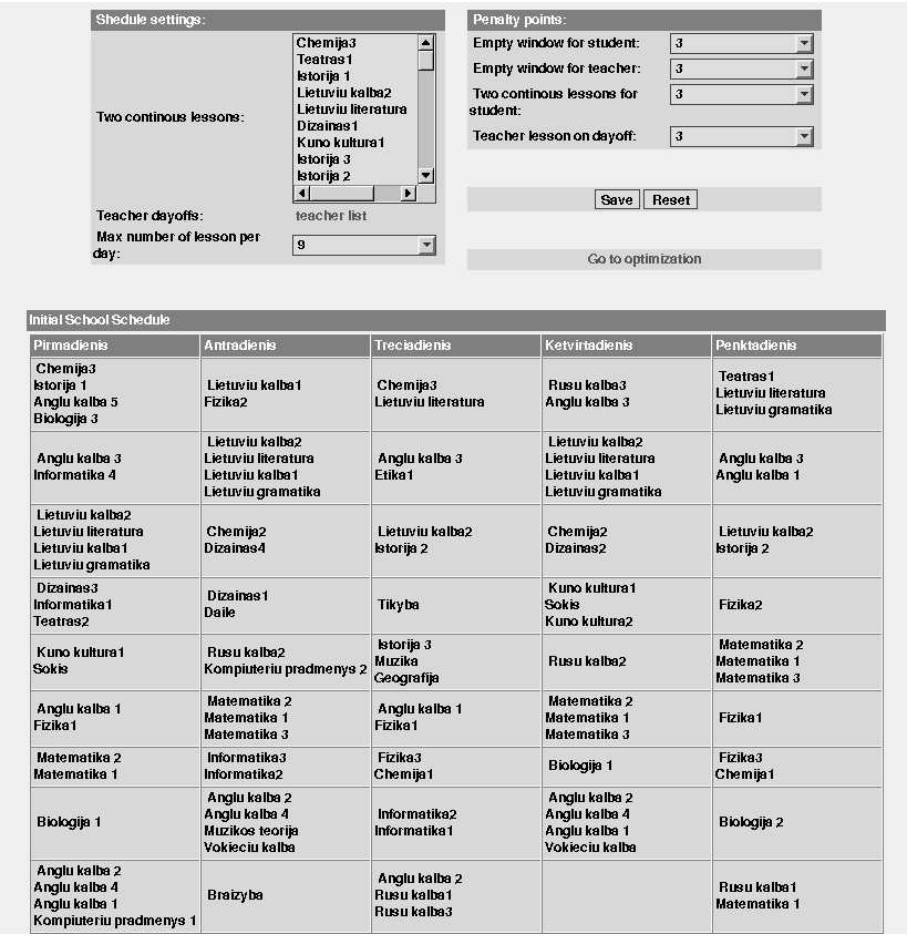
Servlet versions

There are three servlet versions. The simplest one is direct implementation of the applet version. The updated version adds optimization of heuristics by Simulated Annealing (SA) with fixed parameters. The advanced servlet version optimizes parameters of SA completing software implementation of the main theoretical ideas of the Bayesian Heuristic Approach (BHA) (see Figure 17.8).

Figure 17.9 shows how data is uploaded, preferences and restrictions defined.

Figure 17.8. Opening the program.

In the upper-left corner is a list of topics. Marking some of them introduces



Initial School Schedule

Pirmadienis	Antradienis	Treciadienis	Ketvirtadienis	Penktadienis
Chemija3 Istorija 1 Anglu kalba 5 Biologija 3	Lietuviu kalba1 Fizika2	Chemija3 Lietuviu literatura	Rusu kalba3 Anglu kalba 3	Teatras1 Lietuviu literatura Lietuviu gramatika
Anglu kalba 3 Informatika 4	Lietuviu kalba2 Lietuviu literatura Lietuviu kalba1 Lietuviu gramatika	Anglu kalba 3 Etika1	Lietuviu kalba2 Lietuviu literatura Lietuviu kalba1 Lietuviu gramatika	Anglu kalba 3 Anglu kalba 1
Lietuviu kalba2 Lietuviu literatura Lietuviu kalba1 Lietuviu gramatika	Chemija2 Dizainas4	Lietuviu kalba2 Istorija 2	Chemija2 Dizainas2	Lietuviu kalba2 Istorija 2
Dizainas3 Informatika1 Teatras2	Dizainas1 Daile	Tikyba	Kuno kultura1 Sokis Kuno kultura2	Fizika2
Kuno kultura1 Sokis	Rusu kalba2 Kompiuteriu pradmenys 2	Istorija 3 Muzika Geografija	Rusu kalba2	Matematika 2 Matematika 1 Matematika 3
Anglu kalba 1 Fizika1	Matematika 2 Matematika 1 Matematika 3	Anglu kalba 1 Fizika1	Matematika 2 Matematika 1 Matematika 3	Fizika1
Matematika 2 Matematika 1	Informatika3 Informatika2	Fizika3 Chemija1	Biologija 1	Fizika3 Chemija1
Biologija 1	Anglu kalba 2 Anglu kalba 4 Muzikos teorija Vokieciu kalba	Informatika2 Informatika1	Anglu kalba 2 Anglu kalba 4 Anglu kalba 1 Vokieciu kalba	Biologija 2
Anglu kalba 2 Anglu kalba 4 Anglu kalba 1 Kompiuteriu pradmenys 1	Braizyba	Anglu kalba 2 Rusu kalba1 Rusu kalba3		Rusu kalba1 Matematika 1

Figure 17.9. Uploading data.

"double time" preferences. Important restriction is maximal number of hours per day. The default value 9 is shown. Clicking "teacher list" one introduces teacher "days-off" requests.

In the upper-right corner penalty points are defined. That is the subjective part of scheduling defined by persons responsible for scheduling reflecting informal local conditions and preferences.

The upper window in Figure 17.10 is for defining optimization parameters. Here the number of iterations is 10, the initial probability to "miss" a teacher is 0.5, both the initial "temperature" $X1$ and the "cooling" rate $X2$ of SA are open.

The middle window shows the difference between the initial schedule (2532 penalty points) and the optimized one (1743 penalty points). To see the complete schedules one clicks on corresponding words shown in the lower window of Figure 17.10. The initial school schedule is in the lower part of Figure 17.9. The optimized school schedule is in Figure 17.11.
Most of notations are similar to those in the user data. There are some Lithuanian terms: $Mokyklos\ tvarkarastis$ means school schedule, $Pirmadienis$ is Monday, $Antradienis$ is Tuesday, $klaida$ means that user data is not correct here.
The optimized schedule of teacher Jancius is in Figure 17.12. The optimized schedule of student Butkus is in Figure 17.13

4.10 Conclusions

The profiled school scheduling algorithms and software implements permutations similar to these of the traditional school considered in (Mockus, 2000). That is only similarity. Important new ideas are developed and implemented considering profiled school scheduling.

The fist one is that the schedules should be evaluated including both objective and subjective factors in a balanced way. The theoretical framework of this is the Pareto optimum. Simlest way to obtain Pareto optimal schedule is by assigning some penalty points for deviations from the desirable or feasible conditions. The total penalty is minimized. In such a case the penalty points represent subjective opinion of people responsible for schedules. Desirable and feasible conditions reflects objective legal and physical factors.

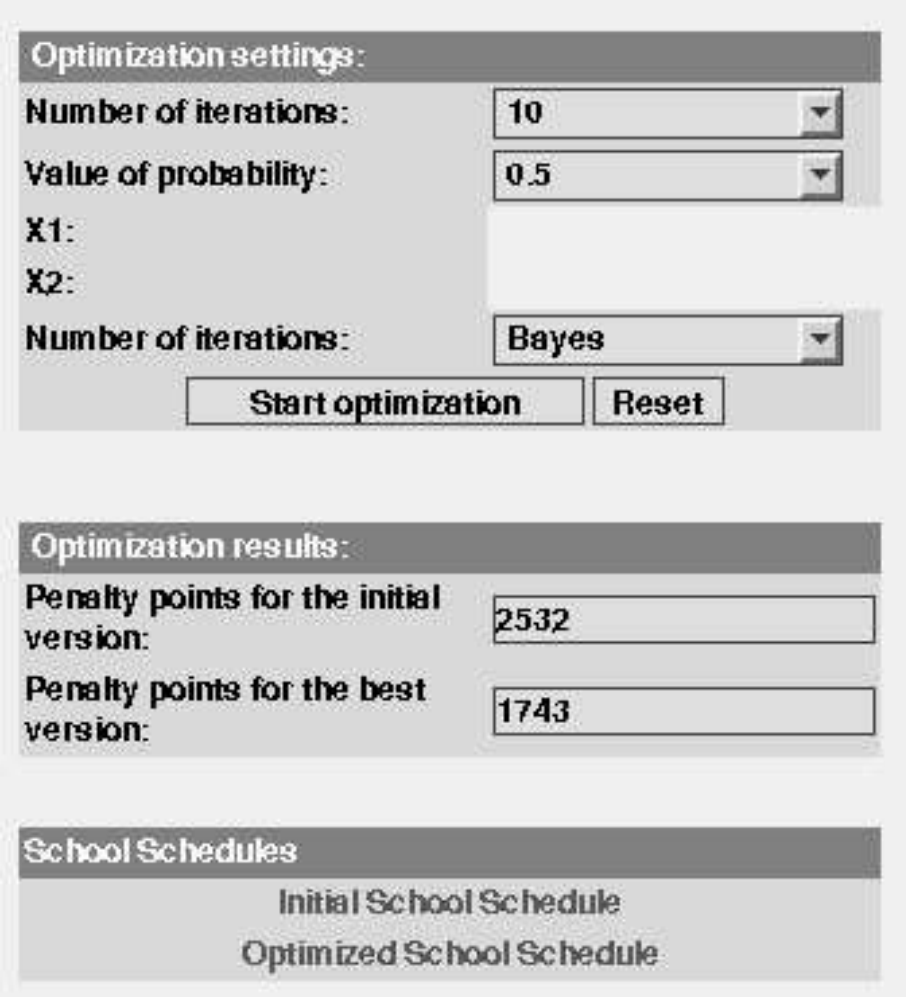
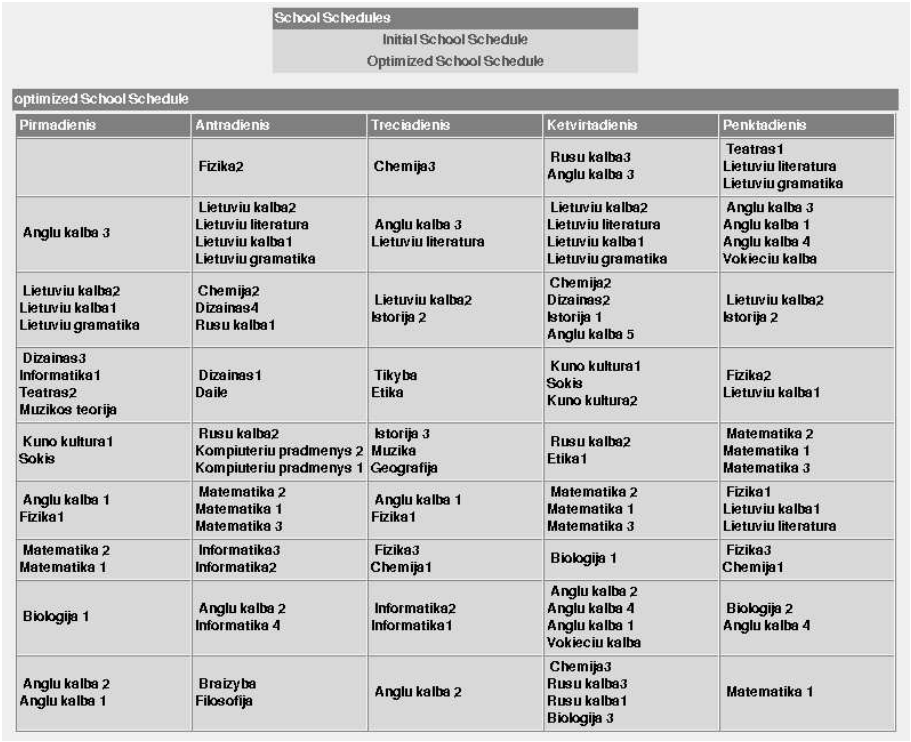


Figure 17.10. Optimization window

The second new idea is optimization of heuristics by selecting their best parameters. That is achieved in three stages. In the first stage the same trivial permutation heuristics is used as in the traditional schedule. That works if one starts from nearly optimal schedule what is true in traditional schools as usual. In the profiled schools some "globalization" of permutation heuristics is needed.

That is achieved in the second stage by the Simulated Annealing (SA) schedules applied with same fixed parameters such as initial "temperature" and the "cooling" rate The third fixed parameter is the probability to delete the name of some teacher waiting in the "improvement" queue. In this case one accepts



optimized School Schedule

Pirmadienis	Antradienis	Treciadienis	Ketvirtadienis	Penktadienis
	Fizika2	Chemija3	Rusu kalba3 Anglu kalba 3	Teatras1 Lietuviu literatura Lietuviu gramatika
Anglu kalba 3	Lietuviu kalba2 Lietuviu literatura Lietuviu kalba1 Lietuviu gramatika	Anglu kalba 3 Lietuviu literatura	Lietuviu kalba2 Lietuviu literatura Lietuviu kalba1 Lietuviu gramatika	Anglu kalba 3 Anglu kalba 1 Anglu kalba 4 Vokieciu kalba
Lietuviu kalba2 Lietuviu kalba1 Lietuviu gramatika	Chemija2 Dizainas4 Rusu kalba1	Lietuviu kalba2 Istorija 2	Chemija2 Dizainas2 Istorija 1 Anglu kalba 5	Lietuviu kalba2 Istorija 2
Dizainas3 Informatika1 Teatras2 Muzikos teorija	Dizainas1 Daile	Tikyba Etika	Kuno kultura1 Sokis Kuno kultura2	Fizika2 Lietuviu kalba1
Kuno kultura1 Sokis	Rusu kalba2 Kompiuteriu pradmenys 2 Kompiuteriu pradmenys 1	Istorija 3 Muzika Geografija	Rusu kalba2 Etika1	Matematika 2 Matematika 1 Matematika 3
Anglu kalba 1 Fizika1	Matematika 2 Matematika 1 Matematika 3	Anglu kalba 1 Fizika1	Matematika 2 Matematika 1 Matematika 3	Fizika1 Lietuviu kalba1 Lietuviu literatura
Matematika 2 Matematika 1	Informatika3 Informatika2	Fizika3 Chemija1	Biologija 1	Fizika3 Chemija1
Biologija 1	Anglu kalba 2 Informatika 4	Informatika2 Informatika1	Anglu kalba 2 Anglu kalba 4 Anglu kalba 1 Vokieciu kalba	Biologija 2 Anglu kalba 4
Anglu kalba 2 Anglu kalba 1	Braizyba Filosofija	Anglu kalba 2	Chemija3 Rusu kalba3 Rusu kalba1 Biologija 3	Matematika 1

Figure 17.11. Optimized school schedule

with some small probability schedules with higher penalties so improving convergence. It is easy to notice that the results of SA approach depend on all of three arbitrarily fixed parameters.

In the final stage these parameters are optimized using methods of stochastic global optimization developed in the framework of the Bayesian Heuristic Approach (BHA) (Mockus, 2000).

Each stage can be involved separately, that makes comparison of their results much simpler. Java implementation presents excellent opportunities to compare different scheduling versions for any interested person. The minimal user guide type information is presented in this paper to make easy the task of testing and applying the presented scheduling models.

The first few trials did show the better results as compared with the commercial software "MIMOSA" that now is used while scheduling Lithuanian profiled schools. However the statistical analysis of different school schedul-

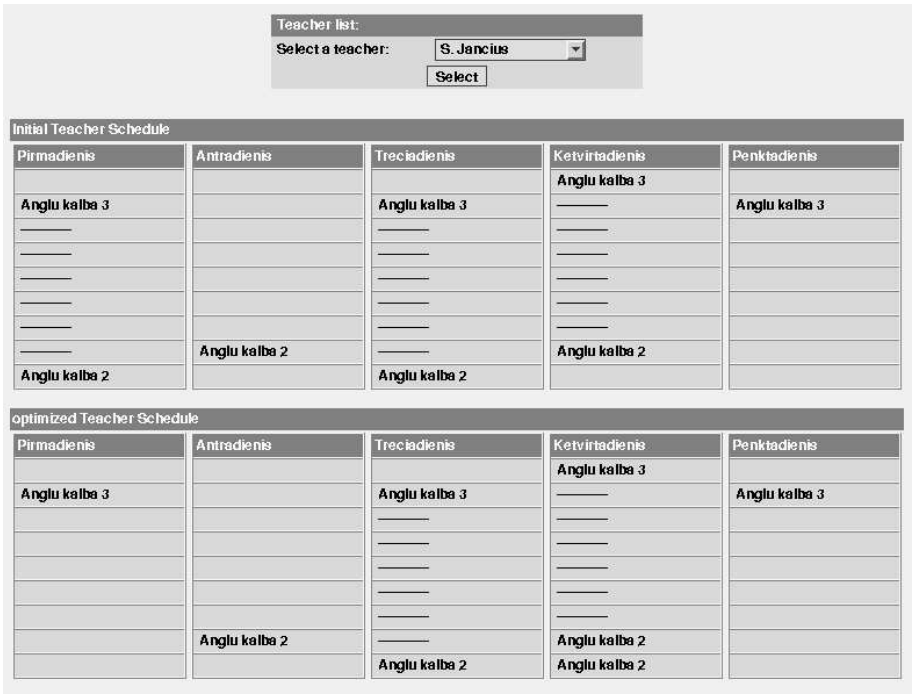


Figure 17.12. Optimized schedule of teacher Jancius

ing methods, algorithms and software systems is a separate important problem to be considered in future.

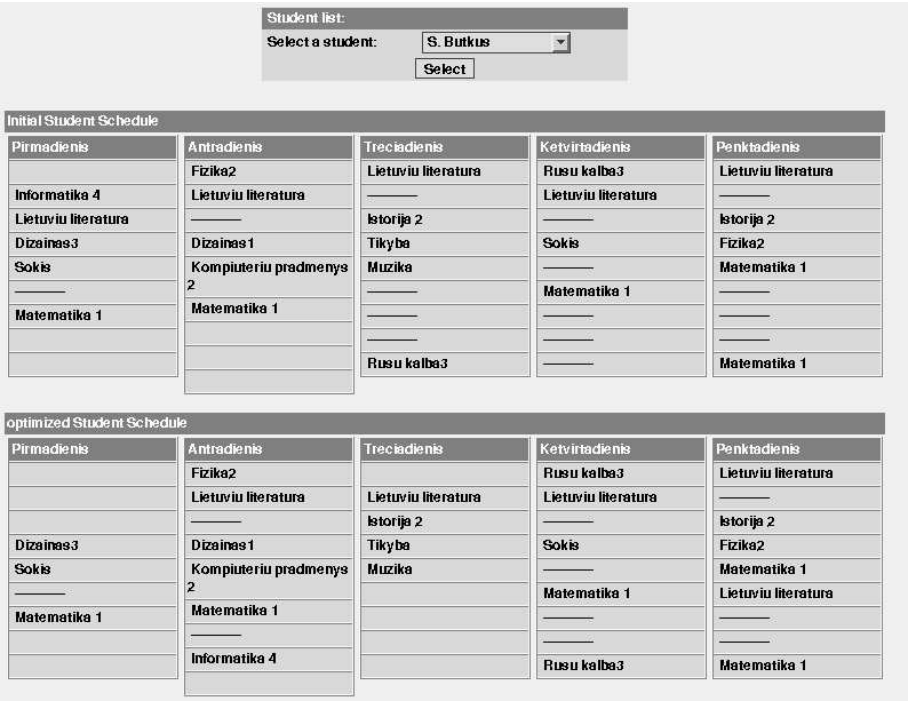


Figure 17.13. Optimized schedule of student Butkus

Chapter 18

SEQUENTIAL STATISTICAL DECISIONS MODEL, "BRIDE PROBLEM"

1. Introduction

Most of the decisions in a personal life and in organizations are made sequentially. That means that, at any given moment, one either makes the final decision, or postpones it hoping for better times. Statistical part of sequential decisions represents uncertainties of future events and a limited reliability of our observations.

The Bride problem is a good example of sequential statistical decisions (Wald, 1947, Wald, 1950). The dynamic programming is a conventional technique to optimize sequential decisions (Bellman, 1957). Applying the dynamic programming to specific problems, one develops specific algorithms, as usual. The algorithms, similar to these of bride's decision making, can be applied choosing the optimal time to buy durable goods, such as cars, houses, computers, to start new business, e.t.c..

Typical industrial applications of sequential decisions are the optimal stopping rules. For example, one can consider stopping of some nuclear plant as a marriage. One can evaluate stopping cost directly. Risks involved in running the plant, when one observes some deviations from the normal routine, can be considered as risks of waiting for the next proposal. That means that one hopes for something better while risking a bad outcome.

2. Average Utility

The Bride problem is to maximize the average utility of marriage by the optimal choice of a groom. Denote the actual goodness of a groom i by ω_i. Denote by s_i the bride's impression about the groom i. $p(\omega_i$ is a prior probability density of goodness ω_i. $p_s(s_i|\omega_i)$ is a probability density of impression s_i. Assume that goodness of different grooms are independent and identically distributed. This means that a prior probability density of goodness is

$$\begin{aligned} p(\omega_i, \omega_j) &= p(\omega_i)p(\omega_j), \\ p(\omega_i) &= p(\omega_j) = p(\omega). \end{aligned} \tag{18.1}$$

Suppose that impressions about the goodness of grooms are independent and identically distributed. Then the probability density of an impression s_i, given the goodness ω_i, is

$$\begin{aligned} p_s(s_i, s_j|\omega) &= p_s(s_i|\omega)p_s(s_j|\omega), \\ p_s(s_i|\omega) &= p_s(s_j|\omega) = p(s|\omega). \end{aligned} \tag{18.2}$$

Assume the Gaussian prior probability density of goodness

$$p(\omega) = \frac{1}{\sqrt{2\pi}\sigma_0} e^{-1/2(\frac{\omega-\alpha_0}{\sigma_0})^2}. \tag{18.3}$$

Here σ_0 is a prior variance and α_0 is a prior mean of goodness. For example, $\alpha_0 > 0$ shows an optimistic bride. $\alpha_0 < 0$ shows that a bride is pessimistic. Suppose, that a prior probability density of bride impressions is

$$p(s|\omega) = \frac{1}{\sqrt{2\pi}\sigma} e^{-1/2(\frac{s-\omega}{\sigma})^2}. \tag{18.4}$$

Here σ^2 is a variance of impressions around the true goodness ω.

Assume that both the groom goodness and the bride impression are random variables depending on many independent factors. This explains the Gaussian distributions (18.3) and (18.4). One defines a posterior probability density of goodness ω, given the impression s, by the Bayesian formula (Bayes, 1783)

$$p(\omega|s) = \frac{p(s|\omega)p(\omega)}{p_s(s)}. \tag{18.5}$$

Here

$$p_s(s) = \int_{-\infty}^{\infty} p(s|\omega)p(\omega)d\omega. \tag{18.6}$$

3. Single-Marriage Case

Denote by d_i the bride's decision about the groom i

$$d_i = \begin{cases} 1, & \text{if } bride\ marry\ the\ groom\ i, \\ 0, & \text{otherwise.} \end{cases} \tag{18.7}$$

Suppose that

$$\sum_{i=1}^{m} d_i = 1. \tag{18.8}$$

The last condition means that brides marry, and marry only once. Formally condition (18.8) defines the set of feasible decisions when the n-th groom proposes

$$D_{N-n} = \begin{cases} 0\ and\ 1, & \text{if } g_{N-n} = 0 \\ 0, & \text{if } g_{N-n} = 1, \end{cases} \tag{18.9}$$

Here g_{N-n} is the marriage index

$$g_{N-n} = 1 - \sum_{i=1}^{N-n-1} \tag{18.10}$$

The marriage index is zero, if the bride is marred. This prevents repeated marriages.

3.1 Bellman's Equations

The expected utility function is $u(s)$. Here s is the impression made by the successful groom[1].

$$u(s) = \int_{-\infty}^{\infty} \omega p(\omega|s) d\omega. \tag{18.11}$$

Denote by $u_N(s)$ the expected utility function, if the impression of the last groom is s

$$u_N(s) = \int_{-\infty}^{\infty} \omega p(\omega|s) d\omega. \tag{18.12}$$

Comparing (18.11) and (18.12) one observes that

$$u(s) = u_N(s). \tag{18.13}$$

[1]By the "successful groom" we mean the groom that a bride marries.

This follows from independence assumptions (18.1) and (18.2). Denote by u_{N-1} the expected utility, if the impression of the $(N-1)$ th groom is s and a bride is making the optimal decision $d = d_{N-1}(s) \in D_{N-1}$

$$u_{N-1}(s) = \max_d (du(s) + (1-d)u_N), \tag{18.14}$$

$$d_{N-1}(s) = arg\max_d (du(s) + (1-d)u_N). \tag{18.15}$$

Here

$$u_N = \int_{-\infty}^{\infty} u_N(s)p_s(s)ds. \tag{18.16}$$

Following the same pattern, we define the expected utility, if the impression of the $(N-n)$-th groom is s and the bride is making the optimal decision $d = d_{N-n}(s) \in D_{N-n}$

$$u_{N-n}(s) = \max_d (du(s) + (1-d)u_{N-n+1}), \tag{18.17}$$

$$d_{N-n}(s) = arg\max_d (du(s) + (1-d)u_{N-n+1}). \tag{18.18}$$

Here

$$u_{N-n+1} = \int_{-\infty}^{\infty} u_{N-n+1}(s)p_s(s)ds. \tag{18.19}$$

Note, that the utility $u(s)$ of accepting a proposal in expressions (18.17) and (18.18) is a function only of impression s. It does not depend on the proposal number $N-n$. That follows from independence assumptions (18.1) and (18.2). Solving these recurrent equations one defines the sequence of optimal decision functions $d_{N-n}(s) \in D_{N-n}$ and the expected utilities $u_{N-n}(s)$. This should be done for all possible impressions $s \in (-\infty, \infty)$ and for all numbers $n = 1, ..., N-1$. One cannot do that in continuous case. Therefore, one uses a discrete approximation

3.2 Discrete Approximation

From expressions (18.11) and (18.12), replacing the integrals by sums one obtains that

$$u_N(s) = u(s) = 2M/K \sum_{k=1}^{K} \omega_k p(\omega_k|s). \tag{18.20}$$

From expression (18.16)

$$u_N = 2M/K \sum_{k=1}^{K} u_N(s_k)p_s(s_k). \tag{18.21}$$

From expression (18.19)

$$u_{N-n+1} = 2M/K \sum_{k=1}^{K} u_{N-n+1}(s_k) p_s(s_k). \tag{18.22}$$

Here $\omega_k \in [-M, M]$, $\omega_1 = -M$, $\omega_K = M$ and $s_k \in [-M, M]$, $s_1 = -M$, $s_K = M$. That is a discrete approximation of the recurrent equations. All possible impressions $s_k \in [-M, M]$ and all numbers $n = 1, ..., N-1$ are considered. One sets the number of iterations K by the accuracy needed.

The results are sequences of optimal decision functions $d_{N-n}(s_k)$ and the expected utilities $u_{N-n}(s_k)$. These sequences are stored in a set of arrays which define how the optimal decisions d and the expected utilities u depend on the possible impressions s_k, $k = 1, ..., K$. Doing this, one avoids the repeated calculations. Large arrays is a disadvantage of this procedure.

3.3 Including the Waiting Cost

The waiting losses are important in many real-life sequential decision problems. Denote by c the loss of waiting for the next groom. Including this parameter into Bellman equations one obtains

$$u_{N-1}(s) = \max_d (du_N(s) + (1-d)(u_N - c)), \tag{18.23}$$

$$d_{N-1}(s) = arg \max_d (du_N(s) + (1-d)(u_N - c)). \tag{18.24}$$

In a similar way one defines the expected utility if the impression of the $(N-n)$-th groom is s and the bride is making the optimal decision $d_{N-n}(s)$

$$u_{N-n}(s) = \max_d (du_N(s) + (1-d)(u_{N-n+1} - c)), \tag{18.25}$$

$$d_{N-n}(s) = arg \max_d (du_N(s) + (1-d)(u_{N-n+1} - c)). \tag{18.26}$$

The other expressions remains the same.

3.4 Non-Linear Case

Expression (18.12) was defined assuming the linear bride's utility function. It was supposed that bride's utility is equal to the goodness of groom $u(\omega) = \omega$. In the real life, utility functions are nonlinear (see chapter 14) and non-convex, as usual. Then expression (18.12) is replaced by this integral

$$u_N(s) = \int_{-\infty}^{\infty} u(\omega) p(\omega|s) d\omega. \tag{18.27}$$

4. Multi-Marriage Case

Condition (18.8) implies that brides marry and marry only once. No divorce is permitted. That is wrong, if the "marriage" represents buying a Personal Computer (PC). We illustrate this by a simple "Buy-a-PC" example.

4.1 " Buy-a-PC" Example

Define PC parameters by a vector $g = (g_1, g_2, g_3)$. Here g_1 is the speed of CPU in MHz, g_2 is the volume of RAM in MB, and g_3 is the volume of HD in GB. Express a subjective utility of PC by the weighted sum $\omega = a_1 g_1 + a_2 g_2 + a_3 g_3$. Here a_i are expressed in \$ per unit.

Assume that a prior probability density of PC utilities is Gaussian

$$p_t(\omega) = \frac{1}{\sqrt{2\pi}\sigma_0} e^{-1/2(\frac{\omega-\alpha_t}{\sigma_0})^2}. \tag{18.28}$$

Here σ_0 is a prior variance and α_t is a prior mean of PC utilities. Suppose that $\sigma_0 = constant$ and that $\alpha_t = \alpha_0 + \alpha_1 t$. This means that expected PC utility is increasing linearly. The expected diversity remains the same.

The PC price in \$ is denoted by l. Suppose that the price of PC depends linearly on the weighted sum

$$l = b_1 g_1 + b_2 g_2 + b_3 g_3. \tag{18.29}$$

Here parameters b_i are defined in \$ per unit and reflect the market prices of CPU, RAM, and HD. Expressing the price l as a function of utility ω:

$$l = h_0 \omega. \tag{18.30}$$

Here

$$h_0 = \frac{b_1 g_1 + b_2 g_2 + b_3 g_3}{a_1 g_1 + a_2 g_2 + a_3 g_3}. \tag{18.31}$$

Assume that one observes the utility ω exactly. This means that the impression $s = \omega$ and that impression errors $\sigma = 0$ in expression (18.2). That simplifies the problem.

4.2 Bellman's Equations

The expected utility function is $u(\omega, q)$. Here ω is the utility of a new PC. q is the utility of the old PC, to be replaced by the new one. Consider a "horizon" of N years. During a year one can change PC only once, if one wishes.

Denote by $u_N(\omega, q)$ the maximal expected utility in the year N

$$u_N(\omega, q) = \max_d (d\omega + (1-d)(q_N - c_N)). \tag{18.32}$$

There are two possible decisions $d = \{0, 1\}$. The decision $d = 1$ means to buy a new PC. The utility of the new PC is ω. The utility of the old one is q.

The utility of the decision $d = 0$, to keep the old PC in the last year N, is $q_N + c_N$. Here $c_N = \tau_N - l_N$ is the price of refusing to buy a new PC. This includes the waiting losses τ minus the price l_N of new PC in the year N defined by (18.30). It is assumed that we abandon the old PC as soon as we obtain the new one. Therefore, one "wins" the price l of the new PC by using the old PC. The optimal decision in the last year N

$$d_N^* = \begin{cases} 1, & \text{if } \omega_N > q_N - c_N, \\ 0, & \text{if } \omega_N \le q_N - c_N. \end{cases} \tag{18.33}$$

Denote by $u_{N-1}(\omega, q)$ the maximal expected utility in the year $N-1$.

$$u_{N-1}(\omega, q) = \max_d (d\omega + (1-d)(u_N(q_N) - c_{N-1})). \tag{18.34}$$

Here $u_N(q)$ is the maximal expected utility in the year N, if the utility of the old PC is q.

$$u_N(q) = \int_{-\infty}^{\infty} u_N(\omega, q) p_N(\omega) d\omega. \tag{18.35}$$

$p_N(\omega)$ is a prior probability density of ω defined by expression (18.28) at a time $t = N$.

$$q_N = \begin{cases} \omega_{N-1}, & \text{if } q_N < q_{N-1}^*, \\ q_{N-1}, & \text{if } q_N \ge q_{N-1}^*. \end{cases} \tag{18.36}$$

Here q_{N-1}^* is obtained from this equation

$$\omega_{N-1} = u_N(q_{N-1}^*) - c_{N-1} \tag{18.37}$$

The optimal decision in the year $N-1$

$$d_{N-1}^* = \begin{cases} 1, & \text{if } \omega_{N-1} > u_N(q_N) - c_{N-1}, \\ 0, & \text{if } \omega_{N-1} \le u_N(q_N) - c_{N-1}. \end{cases} \tag{18.38}$$

Denote by $u_{N-i}(\omega, q)$ the maximal expected utility in the year $N-i$, $i = 1, ..., N-1$. Then

$$u_{N-i}(\omega, q) = \max_d (d\omega + (1-d)(u_{N-i+1}(q_{N-i+1}) - c_{N-i})). \quad (18.39)$$

Here $u_{N-i+1}(q)$ is the maximal expected utility in the year $N-i+1$, if the utility of the old PC is q.

$$u_{N-i+1}(q) = \int_{-\infty}^{\infty} u_{N-i+1}(\omega, q) p_{N-i+1}(\omega) d\omega. \quad (18.40)$$

$p_{N-i+1}(\omega)$ is a prior probability density of ω defined by expression (18.28) at a time $t = N-i+1$.

$$q_{N-i+1} = \begin{cases} \omega_{N-i}, & \text{if } q_{N-i+1} < q^*_{N-i}, \\ q_{N-i}, & \text{if } q_{N-i+1} \geq q^*_{N-i}. \end{cases} \quad (18.41)$$

q^*_{N-i} is obtained from the equation

$$\omega_{N-i} = u_{N-i+1}(q^*_{N-i}) - c_{N-i} \quad (18.42)$$

The optimal decision in the year $N-i$

$$d^*_{N-i} = \begin{cases} 1, & \text{if } \omega_{N-i} > u_{N-i+1}(q_N - i + 1) - c_{N-i}, \\ 0, & \text{if } \omega_{N-i} \leq u_{N-i+1}(q_{N-i+1}) - c_{N-i}. \end{cases} \quad (18.43)$$

Here the decision functions d^*_{N-i} depend on two variables ω, q defining the corresponding utilities of the new and the old computer. That is difficult for both the calculations and grafical representations.

One applies the discrete approximation such as in Section 3.2. The optimal decision functions depend on two variables ω and q. Therefore, one needs K times more calculations to obtain the same accuracy as in the single marriage case (see Section 3.2).

To solve real life problems the "directly unobservable" factors should be included into the "Buy-a-PC" model. For example , one can estimate the expected life-span $T(s)$ of PC, given the impression s, by the Bayesian formula

$$T = \int_{-\infty}^{\infty} \tau p(\tau|s) d\tau, \quad (18.44)$$

$$p(\tau|s) = \frac{p(s|\tau)p(\tau)}{p_s(s)}. \quad (18.45)$$

Here τ is the life-span of PC and s is the user's impression about its reliability. The impression s depends on the warranty, the reputation of the manufacturer and on some subjective factors, too.

One-Step-Ahead Approximation

To simplify the calculations, the "One-Step-Ahead" approximation is suggested. Here one assumes that the next step is the last one and introduces the parameter $r = \omega_{N-1} - q_{N-1} + c_{N-1}$.

The one-step-optimal decision

$$d_{N-1}(r) = \begin{cases} 1, & \text{if } r \geq 0, \\ 0, & \text{if } r < 0. \end{cases} \tag{18.46}$$

Following the same pattern, one defines the maximal expected utility in $(N-n)$-th year

$$u_{N-n}(\omega, q) = \max_{d}(d\omega_{N-n} + (1-d)u_{N-n+1}(q)), \tag{18.47}$$

where

$$u_{N-n+1}(q) = \int_{-\infty}^{\infty} u_{N-n+1}(\omega, q)p_{N-n+1}(\omega)d\omega. \tag{18.48}$$

Here

$$q = \begin{cases} \omega_{N-1}, & \text{if } r \geq 0, \\ q_{N-1}, & \text{if } r < 0. \end{cases} \tag{18.49}$$

The one-step-optimal decision

$$d_{N-n}(r) = \begin{cases} 1, & \text{if } r \geq 0, \\ 0, & \text{if } r < 0. \end{cases} \tag{18.50}$$

The one-step-optimal decision $d_n(r)$ is a function of difference $r = \omega_{N-n} - u_{N-n+1}(q)$ that depends on two parameters ω and q. Therefore, one defines it as a function of critical pairs (ω, q) depending on time $t = n$.

Solving these recurrent equations one defines the sequence of one-step-optimal decision functions $d_{N-n}(r)$ and the expected utilities $u_{N-n}(\omega, q)$. This should be done for all possible values of ω and q.

The One-Step-Ahead approximation, assuming that $r = \omega_{N-n} - q_{N-n} - l_{N-n}$, reduces the computing time to a level of single marriage case. However, one needs to estimate the approximation error.

5. Optimal Ordering of Supplies

The important application a of sequential statistical decisions is optimal ordering of supplies. Consider, for example, same computer shop that orders d

units of RAM (Random Access Memory) at fixed times $t = 1, ..., N$ dividing the time into N stages of unit length each.

5.1 Uncertain Demand

One optimizes the supply d when the demand ω is unknown. Denote by ω a number of RAM units demanded during one stage. Denote the retail price of one RAM unit by c_r and the wholesale price by $c_w \leq c_r$. The utility function-the profit-at the $(N-i)$-th stage

$$v_{N-i}(\omega, d, q) = \begin{cases} c_r\omega - c_w d, & \text{if } b \geq \omega, \\ c_r\omega - c_w d + L(b-\omega), & \text{if } b < \omega. \end{cases} \quad (18.51)$$

Here $c = c_r - c_w$ and $b = d + q$, q is the number of unsold units.
L is the loss due to short supply including fines for breaking supply contracts and/or damages to reputation as an reliable retailer.
All these parameters correspond to the stage $N-i,\ i = 0, 1, ..., N-1$, meaning that
$\omega = \omega_{N-i},\ d = d_{N-i}, q = q_{N-i}$
where ω_{N-i} is the demand at the stage $N - i$,
$d = d_{N-i} \in D$ is the supply belonging to a feasible set $D = D_{N-i}$ at this stage, and
$q = q_{N-i-1} = d_{N-i-1} + q_{N-i-2} - \omega_{N-i-1}$
is the number of unsold RAM units, left from the previous stage $N - i - 1$.
Here and later we omit the lower indices to make expressions shorter.

Denote by $p_{N-i}(\omega)$ the probability of demand $\omega_{N-i} = \omega$ at the stage $N - i$. Usually the demand $\omega = 0, 1, 2, ...$ during the time τ is described by the Poisson distribution (Tijms, 1994)

$$p_{N-i}(\omega) = e^{-\lambda\tau}\frac{(\lambda\tau)^\omega}{\omega!}. \quad (18.52)$$

Considering stages of unit time $\tau = 1$ and

$$p_{N-i}(\omega) = e^{-\lambda}\frac{\lambda^\omega}{\omega!}, \quad (18.53)$$

where $\lambda = \lambda_{N-i}$ is the expected demand at the stage $N - i$. Then the expected utility at the stage $N - i,\ i = 0, ..., N - 1$

$$u_{N-i}(d, q) = \sum_\omega v_{N-i}(\omega, d, q)\ p_{N-i}(\omega) \quad (18.54)$$

The maximal expected utility at the last stage N

$$u_N(q) = \max_{d \in D_N} \sum_{\omega} v_N(\omega, d, q)\, p_N(\omega) \tag{18.55}$$

where $q = q_{N-1}$ is a number of unsold RAM units available at the beginning of the N-th stage[2],
$d = d_N$ is a number of RAM units ordered at the beginning of the N-th stage,
$\omega = \omega_N$ is the demand at the stage N, D_N is a set of feasible supplies at the stage N.

Denote by $d_N(q)$ the optimal decision function maximizing the expected utility (18.55) at any fixed q.

Denote by $u_{N-1}(q)$ the maximal expected utility at the $(N-1)$-th stage

$$u_{N-1}(q) = \max_{d \in D_{N-1}} (u_{N-1}(d, q) + \sum_{\omega} u_N(q)\, p_N(\omega)). \tag{18.56}$$

Note, that the meaning of parameter q depends on the index of the function u, for example, $q = q_{N-1}$, if the index is N, and $q = q_{N-2}$, if this index is $N-1$, etc.

The maximum of (18.56) is obtained by the optimal decision function $d_{N-1}(q)$ where $q = q_{N-2}$.

Extending (18.56) to $i = 0, ..., N-1$ one obtains

$$u_{N-i}(q) = \max_{d \in D_{N-i}} (u_{N-i}(d, q) + \sum_{\omega} u_N(q)\, p_{N-i+1}(\omega)) \tag{18.57}$$

Solving recurrent equations (18.55)-(18.57) one defines the sequence of optimal decision functions
$d_{N-i}(q),\ i = 0, ..., N-1$
and the optimal expected utility functions $u_{N-i}(q)$.
This is done for all possible remaining RAM units q and for all stages $i = 1, ..., N-1$.

In the software terms that means calculating recurrently the sequence $i = 0, 1, ..., N-1$ of arrays that shows how the optimal supply $d = d_{N-i}(q)$ and the maximal profit $u = u_{N-i}(q)$ depend on the unsold RAM units q, left from the previous stages. This way one reduces computing time considerably, because

[2]The meaning of the argument q of the function $u_N(q)$ is defined by the index N, for example, $q = q_{N-1}$, if this index is N, etc.

array access operations need much less time as compared to multi-dimensional optimization.

5.2 Uncertain Wholesale Price

Denote by $p^c_{N-i}(c_w) = P\{c_w(N-i) = c_w\}$ a probability that the wholesale price $c_w(N-i)$ at the stage $N-i$ is c_w. Denote by $C_w(N-i)$ the expected wholesale price c_w at the stage $N-i$. Assuming the Gaussian distribution of the wholesale price logarithm, one obtains the lognormal density function (Tijms, 1994)

$$p^c_{N-i}(c_w) = \frac{1}{\alpha c_w \sqrt{2\pi}} e^{-\frac{(ln(c_w)-\gamma)^2}{2\alpha^2}} \tag{18.58}$$

where c_w means $c_w(N-i)$, $\alpha > 0$ is the shape parameter, and γ is the scale parameter. The expected value $C_w(N-i)$ and the standard $S_w(N-i)$ of the lognormal distribution

$$C_w(N-i) = e^{\gamma+\alpha^2/2} \tag{18.59}$$

$$S_w(N-i) = e^{\alpha^2} - 1 \tag{18.60}$$

The lognormal density is unimodal with maximum at $c_w = exp(\gamma - \alpha^2)$.

In the case of random wholesale prices $c_w = c_w(N-i)$, the profit at the $(N-i)$-th stage

$$v_{N-i}(\omega, d, q) = \begin{cases} c_r\omega - c_w d, & \text{if } b \geq \omega, \\ c_r\omega - c_w d + L(b-\omega), & \text{if } b < \omega. \end{cases} \tag{18.61}$$

Here $c = c_r - c_w(N-i)$, $q = q_w(N-i-1)$ is the number of unsold units. Then the expected utility at the stage $N-i,\ i = 0, ..., N-1$

$$u_{N-i}(d, q) = \sum_{\omega} \int_0^{\infty} v_{N-i}(\omega, d, q)\ p^c_{N-i}(c_w) dc_w\ p_{N-i}(\omega) \tag{18.62}$$

The maximal expected utility at the last stage N

$$u_N(q) = \max_{d \in D_N} \sum_{\omega} \int_0^{\infty} v_N(\omega, d, q)\ p^c_N(c_w) dc_w\ p_N(\omega) \tag{18.63}$$

where $q = q_{N-1}$ is a number of unsold RAM units left at the beginning of N-th stage,
$c_w = c_w(N)$ is the wholesale price at the stage N.

Denote by $d_N(q)$ the optimal decision function maximizing the expected utility (18.63) at any fixed q.

Denote by $u_{N-1}(q)$ the maximal expected utility at the $(N-1)$-th stage

$$u_{N-1}(q) = \max_{d \in D_{N-1}} (u_{N-1}(d,q) + \sum_{\omega} \int_0^{\infty} u_N(q)\, p_N^c(c_w) dc_w\, p_N(\omega)). \tag{18.64}$$

That is achieved by the optimal decision function $d_{N-1}(q)$
where $q = q_{N-2}$.

Extending to $i = 0, ..., N-1$ one obtains

$$u_{N-i}(q) = \max_{d \in D_{N-i}} (u_{N-i}(d,q) + \sum_{\omega} \int_0^{\infty} u_N(q)\, p_{N-i+1}^c(c_w) dc_w\, p_{N-i+1}(\omega)) \tag{18.65}$$

Solving recurrent equations (18.63)-(18.65) one defines the sequence of optimal decision functions
$d_{N-i}(q),\ i = 0, ..., N-1$
and the optimal expected utilities $u_{N-i}(q)$.
This is done for all possible numbers of RAM units q available at the beginning of stages $i = 1, ..., N-1$.

5.3 Market Elasticity

We call by Market Elasticity the relation $\Omega = \Omega(c_r)$ of expected demand Ω to the retail price $c_r \in C$, where C is a set of feasible retail prices. A simple approximation is the exponential function

$$\Omega(c_r) = be^{-\beta c_r} \tag{18.66}$$

where $b = b_{N-i}$ is the saturation demand, $\beta = \beta_{N-i}$ is the elasticity parameter, $c_r = c_r(N-i) \in C_{N-i}$ is the retail price, and C_{N-i} is a set of feasible retail prices, all at the stage $N-i$.

In this case the profit function at the $(N-i)$-th stage

$$v_{N-i}(\omega, d, q, c_r) = \begin{cases} c_r\omega - c_w d, & \text{if } b \geq \omega, \\ c_r\omega - c_w d + L(b-\omega), & \text{if } b < \omega. \end{cases} \tag{18.67}$$

Here c denotes the difference $c = c_r(N-i) - c_w(N-i)$ of retail and wholesale prices at the stage $N - i$.
The expected utility at the stage $N - i,\ i = 0, ..., N - 1$

$$u_{N-i}(d, q, c_r) = \sum_{\omega} \int_0^{\infty} v_{N-i}(\omega, d, q)\ p^c_{N-i}(c_w) dc_w\ p_{N-i}(\omega) \quad (18.68)$$

where $c_r = c_r(N - i)$.
The maximal expected utility at the last stage N

$$u_N(q) = \max_{d \in D_N, c_r \in C_N} \sum_{\omega} \int_0^{\infty} v_N(\omega, d, q, c_r)\ p^c_N(c_w) dc_w\ p_N(\omega) \quad (18.69)$$

Denote by $d_N(q) = (d_N(q), c_N(q))$, where q denotes q_{N-1}, the two-dimensional optimal decision function maximizing the expected utility (18.69) at any fixed q.

Denote by $u_{N-1}(q)$ the maximal expected utility at the $(N - 1)$-th stage

$$u_{N-1}(q) = \max_{d \in D_{N-1}, c_r \in C_{N-1}} (u_{N-1}(d, q, c_r) + \int_0^{\infty} \sum_{\omega} u_N(q)\ p^c_N(c_w) dc_w\ p_N(\omega)). \quad (18.70)$$

That is achieved by the two-dimensional optimal decision function $d_{N-1}(q) = (d_{N-1}(q), c_{N-1}(q))$
where q means q_{N-2}.

Extending (18.70) to $i = 0, ..., N - 1$ one obtains

$$u_{N-i}(q) = \max_{d \in D_{N-i}, c_r \in C_{N-i}} (u_{N-i}(d, c_r, q) + \sum_{\omega} \int_0^{\infty} u_N(q)\ p^c_{N-i+1}(c_w) dc_w\ p_{N-i+1}(\omega)) \quad (18.71)$$

Solving recurrent equations (18.69)-(18.71) one defines the sequence of decision functions
$d_{N-i}(q) = (d_{N-i}(q), c_{N-i}(q))\ i = 0, ..., N - 1$
and the optimal expected profits $u_{N-i}(q)$.
This is done for all possible numbers q of RAM units q available at the beginning of stages $i = 1, ..., N - 1$.

In the case of market elasticity, to represent maximal profits $u_{N-i}(q)$ and two-dimensional optimal decisions $d_{N-i}(q) = (d_{N-i}(q), c_{N-i}(q))\ i = 0, ..., N-1$ as functions of remaining RAM units q one needs to calculate a sequence of

N arrays defining the sequence of two-dimensional decision functions. This is difficult, therefore, one starts from the simplest case of uncertain demand (see section 5.1), as usual.

Note that in the actual calculations of expressions (18.63)-(18.71) the integrals are replaced by sums and the densities by probabilities using expressions similar to those in the section 3.2.

In the demonstration version the "horizon" N means the estimated end of business. In the expert version one uses the "moving horizon" $N_o < N$, repeating the optimization after each stage. This way one simplifies calculations and updates data.

The recurrent equations (18.55)-(18.57) include Bride problems as special cases with only two feasible decisions $D = \{0, 1\}$ and different utility functions (18.51).

6. Software Examples

Several interactive versions of the Bride problem are implemented as Java applets. These, and the corresponding $C++$ software are on web-sites (see section 4).

6.1 Single "Marriage"

For this example one selects the section "Dicrete, Linear and Dynamic Programming" in some web-site and opens "Bride-1". Figure 18.1 shows the input window of the Bride problem (Kajokas, 1997)
Figure 18.2 shows simulated results of the "optimal marriage".
Figure 18.3 shows the optimal decision function defining how the critical "goodness" of a groom depends on his number.

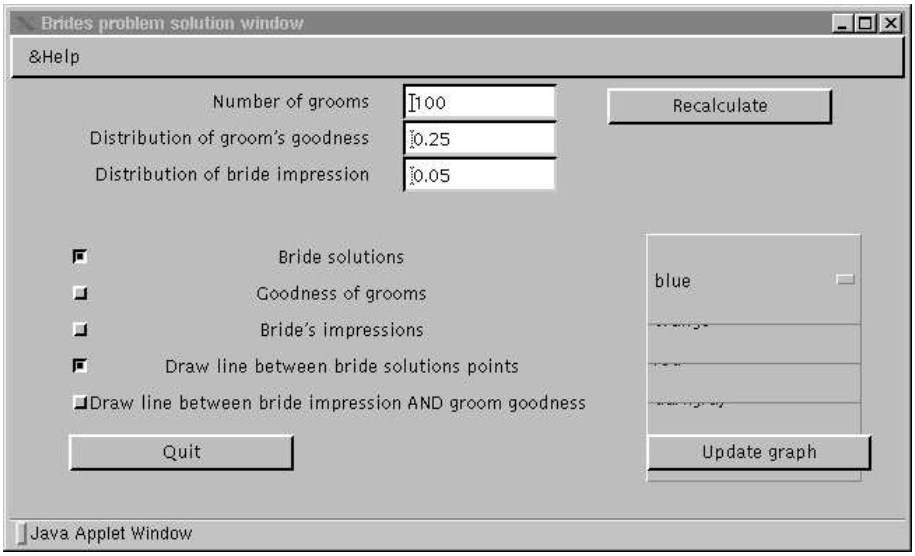


Figure 18.1. The input window of the Bride problem.

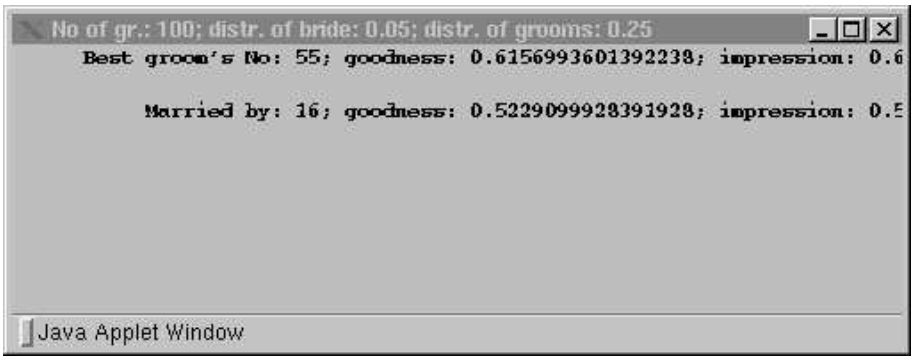


Figure 18.2. The result of statistical simulation of the Bride problem.

"Buy-a-Car" Version

The "Buy-a-Car" version applies modified "Single-Marriage" software. Figure 18.4 shows the help window of the "Buy-a-Car" version of the Bride problem (Mikenas, 1998).
Figure 18.5 shows the input window of the "Buy-a-Car" problem.
Figure 18.6 shows the output window of the "Buy-a-Car" problem.
Figure 18.7 shows the optimal decision function of the "Buy-a-Car" problem.

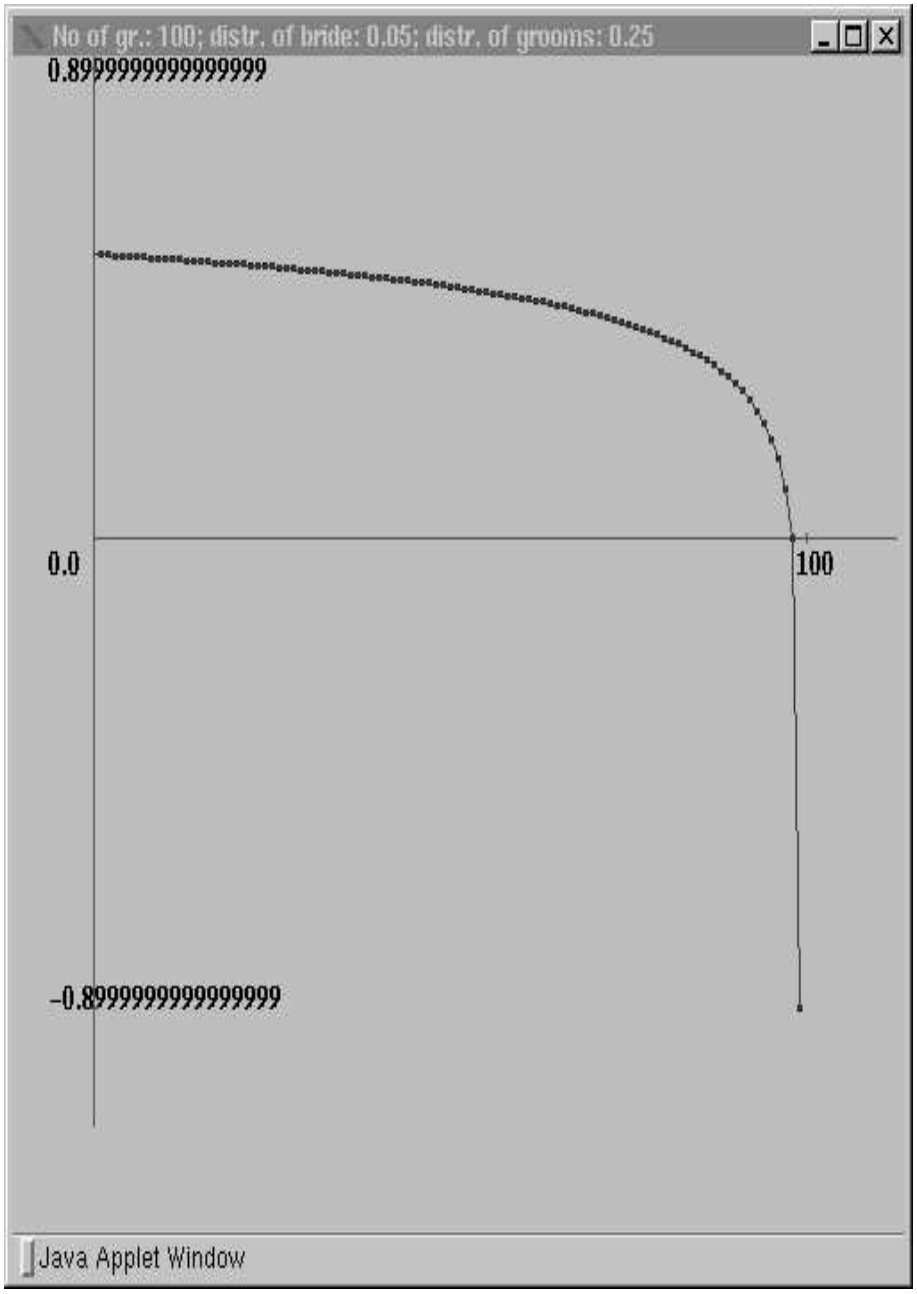


Figure 18.3. The optimal decision function of the Bride problem.

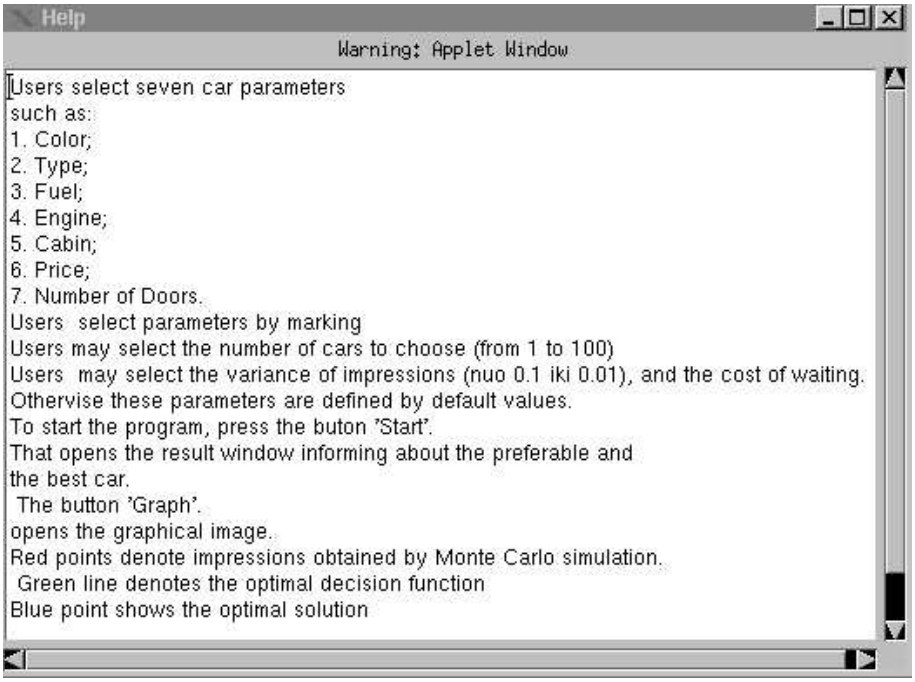


Figure 18.4. The help window solving the "Buy-a-Car" problem.

The "Buy-a-Car" algorithm is just a first approximation. It should be improved. For example, improving the By-a-PC model one should

- remove the "single marriage" condition (18.8),
- add "Change-a-Car" cost,
- express desirable car properties (see Figure 18.5), using conditions of Pareto optimality (Mockus, 1989b),
- extend the property list by including properties that are not directly observable, such as the reliability of a car,
- keep the "No-Return" rule.

"No-Return" means that one cannot return to a situation which he or she rejected previously. The "No-Return" rule is essential in the dynamic programming. One considers "Return" cases in the framework of discrete or non-linear programming. That requires more computing, as usual.

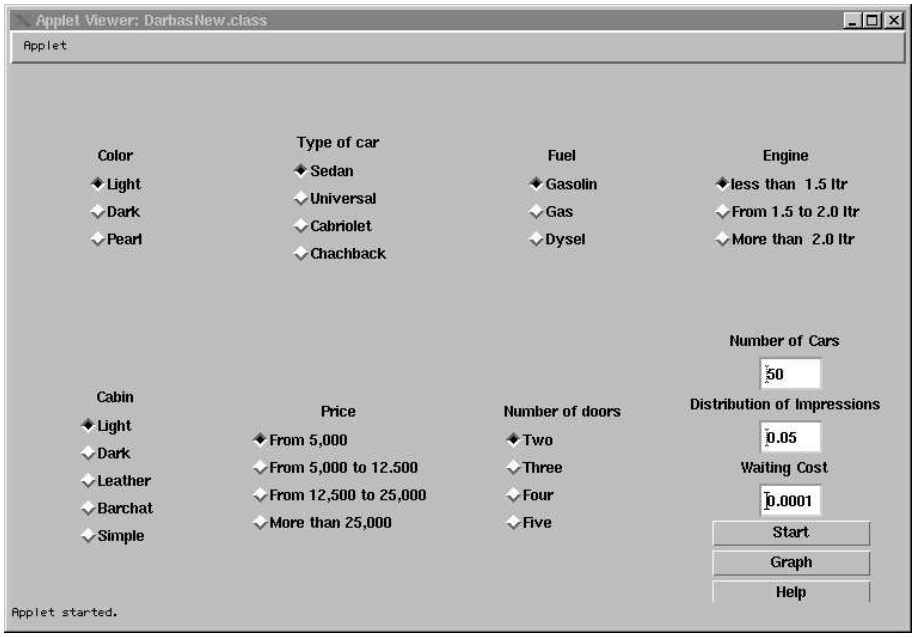


Figure 18.5. The input window solving the "Buy-a-Car" problem.

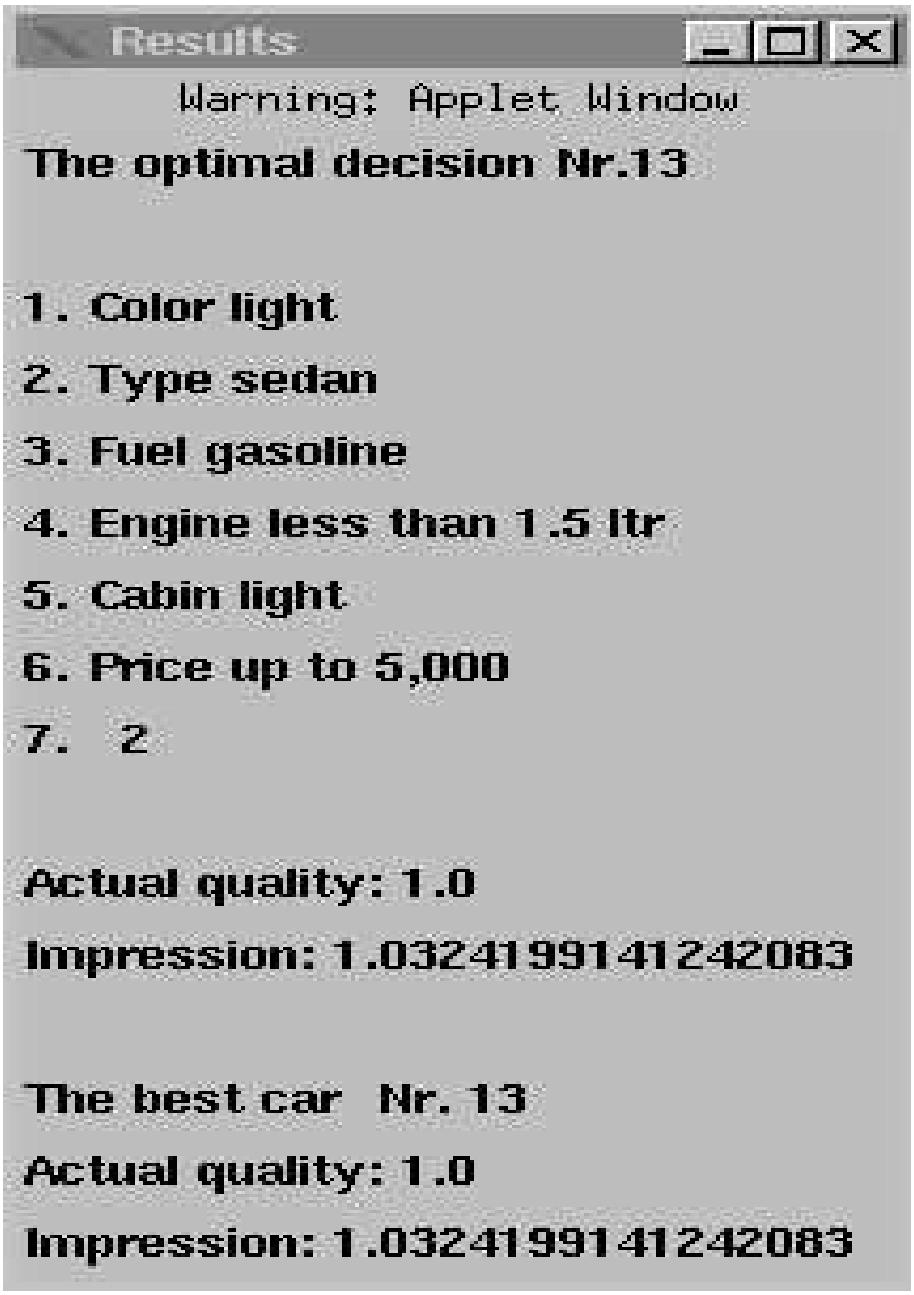


Figure 18.6. The output window of the "Buy-a-Car" problem.

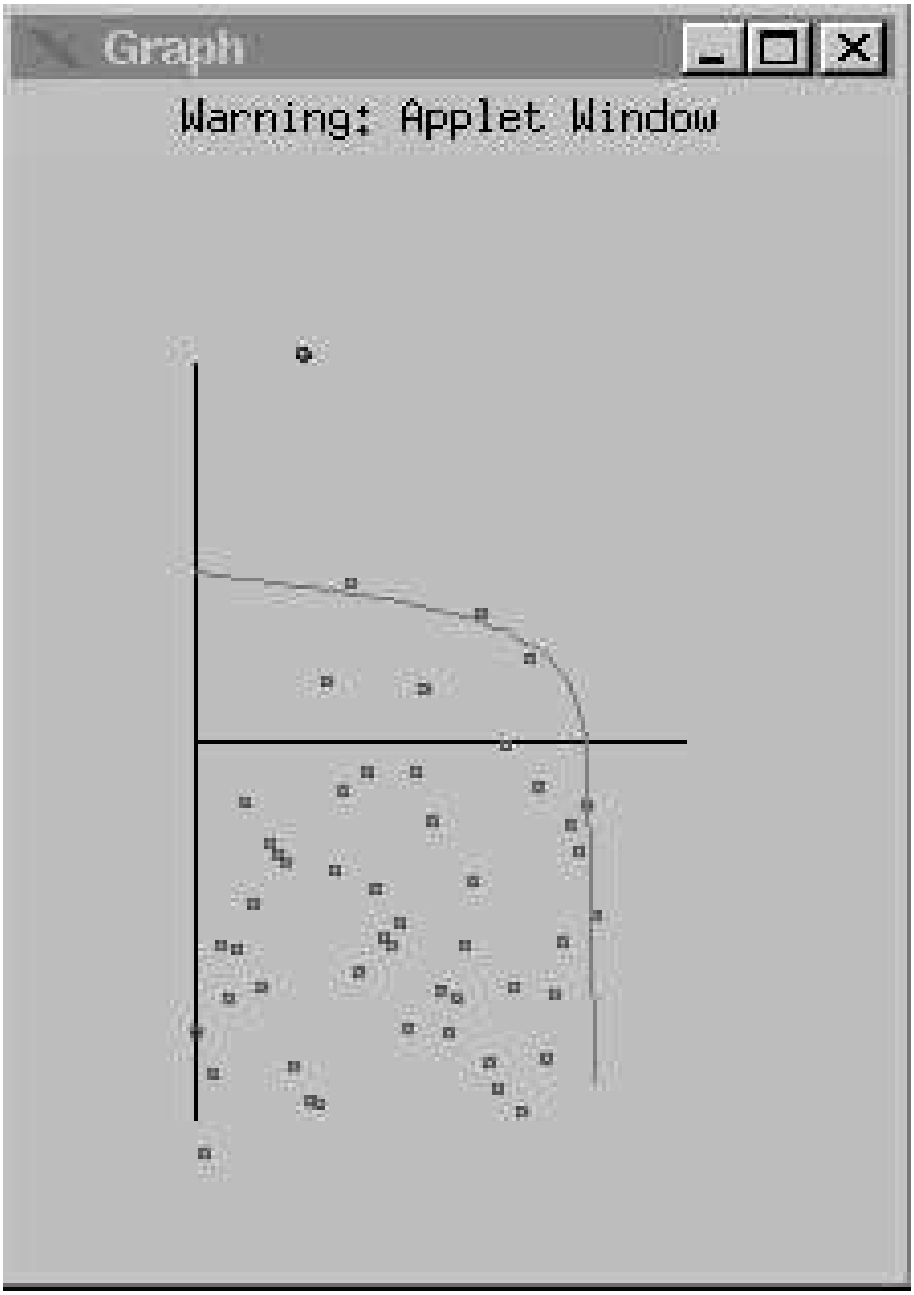


Figure 18.7. The optimal decision function of the "Buy-a-Car" problem.

6.2 Many "Marriages"

First one selects the problem "Bride-Multi" in the section "Discrete, Dynamic and Linear Programming ". The program (Damasevicius, 2000) starts by clicking the button $Start$ at the end of the $BrideProblem$ page. Data is entered using the operation page (see Figure 18.8). Optimization starts by clicking the "Calculate" button. The decision function (see Figure 18.9) is defined by

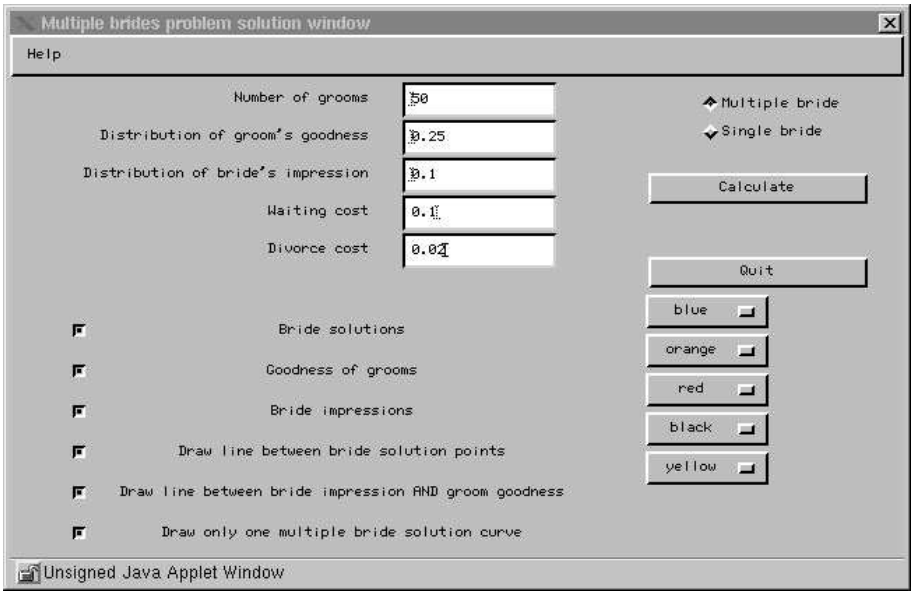


Figure 18.8. Input page.

Bellman equations (18.33)(18.50). The "State-of Nature" generated by Monte Carlo simulation of probability distributions (18.1) (18.2) is also shown in Figure 18.9. The jumps of the decision function correspond to two marriages. In addition, Figure 18.10 shows the best "State-of-Nature" .

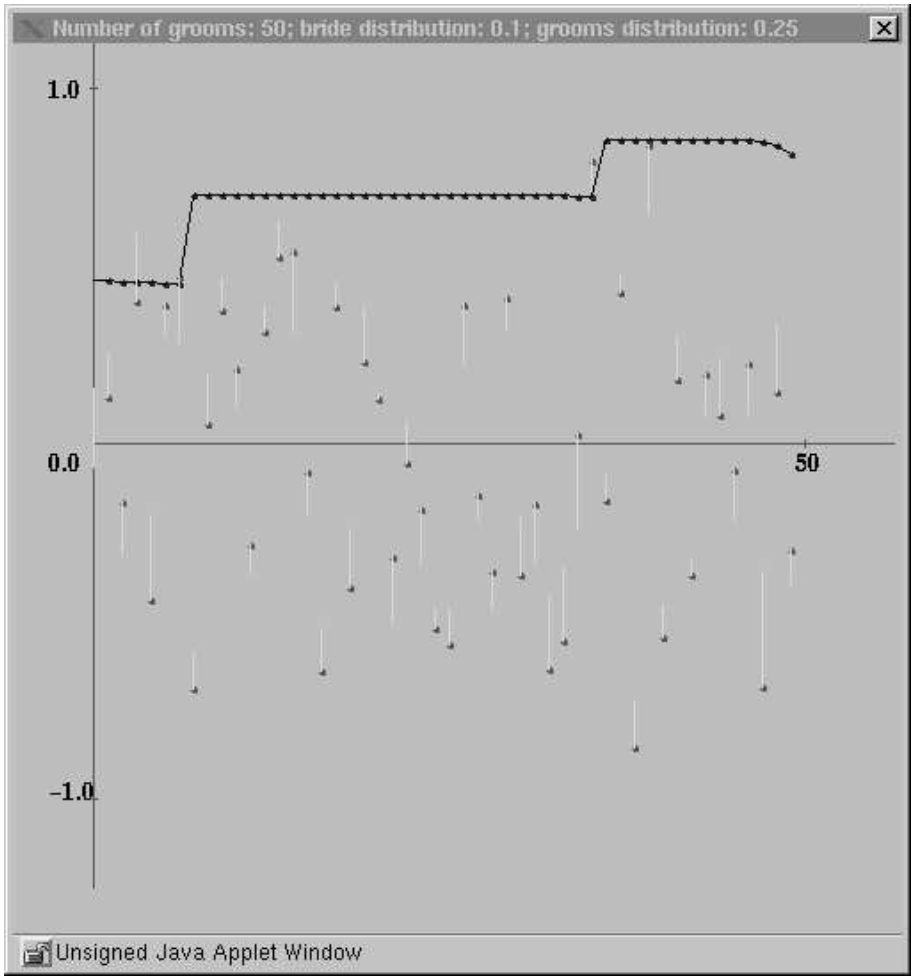


Figure 18.9. Example of a decision function.

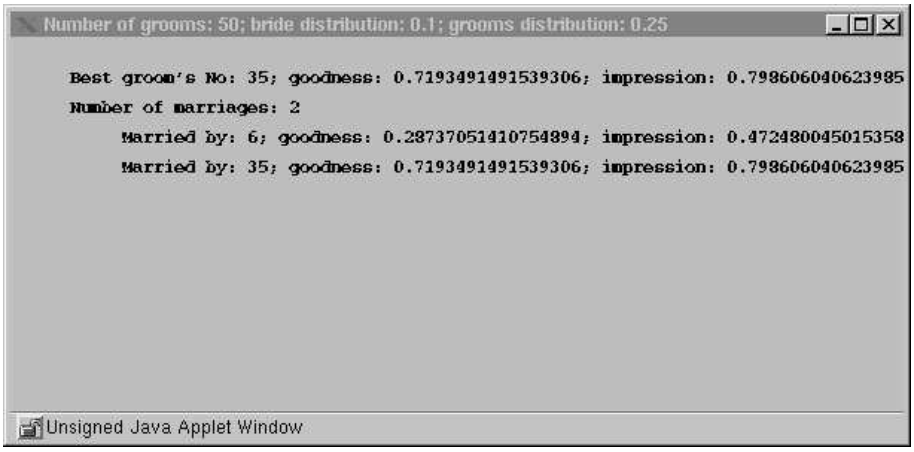


Figure 18.10. Monte Carlo simulation of the"State of Nature".

References

[Ahuja et al., 1994] Ahuja, N., Dozzi, S., and s. Abourizk (1994). *Project Management Techniques in Planning and Controlling Construction Projects*. John Wiley, New York.

[Andriulaitis and Varnagiris, 2000] Andriulaitis, A. and Varnagiris, V. (2000). Global minimizer for c# (gmc#) version 1.0b. Technical report, Institute of Mathematics and Informatics, Akademijos 4, Vilnius, Lithuania.

[Androulakis and Venkatasubramanian, 1991] Androulakis, I. and Venkatasubramanian, V. (1991). A genetic algorithm: Framework for process design and optimization. *Computers in Chemical Engineering*, 15:217–228.

[Baker, 1974] Baker, K. R. (1974). *Introduction to Sequencing and Scheduling*. John Wiley & Sons, New York.

[Bayes, 1783] Bayes, T. (1783). An essay towards solving a problem in the doctrine of chances. *Phil. Transactions of Royal Society*, 53:370–418.

[Bellman, 1957] Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, Princeton, New Jersey.

[Biggs, 1975] Biggs, M. C. (1975). Constrained minimisation using recursive quadratic programming: some alternative subproblem formulations. In *Towards global optimisation*. North Holland, Amsterdam.

[Brouwer, 1912] Brouwer, L. (1912). Uber abbildung von mannigfaltigkeiten. *Math. Ann.*, 71:97–115.

[Browder, 1968] Browder, F. (1968). The fixed point theory of multivalued mappings in topological vector spaces. *Math. Ann.*, 177:283–302.

[Cheung, 1993] Cheung, Y.-W. (1993). Long memory in foreign exchange rates. *Journal of Business and Economic Statistics*, 1:93–101.

[Cheung and Lai, 1993] Cheung, Y.-W. and Lai, K. (1993). Fractional co-integration analysis of purchasing power parity. *Journal of Business and Economic Statistics*, 1:103–112.

[Cohn and Fielding, 1999] Cohn, H. and Fielding, M. (1999). Simulated annealing: Searching for an optimal temperature schedule. *SIAM J. Optim.*, 9:779–802.

[Damasevicius, 2000] Damasevicius, R. (2000). Multi-marriage bride model in java. Technical report, Institute of Mathematics and Informatics, Akademijos 4, Vilnius, Lithuania.

[DeGroot, 1970] DeGroot, M. (1970). *Optimal Statistical Decisions*. McGraw-Hill, New York.

[Diaconis, 1988] Diaconis, P. (1988). Bayesian numerical analysis. In *Statistical Decision Theory and Related Topics*, pages 163–175. Springer Verlag.

[Diebold and Rudebusch, 1989] Diebold, F. X. and Rudebusch, G. D. (1989). Long memory and persistence in aggregate output. *Journal of Monetary Economics*, 24:189–209.

[Dixon and Szego, 1978] Dixon, L. and Szego, G. (1978). *Towards global optimisation 2*. North Holland, Amsterdam.

[Dzemyda and Senkiene, 1990] Dzemyda, G. and Senkiene, E. (1990). Simulated annealing for parameter grouping. In *Transactions. Information Theory Statistical Decision Theory Random Processes*, pages 373–383, Praque.

[Dzemyda et al., 1984] Dzemyda, G., Vaitiekunas, F., Valincius, G., Vysniauskas, J., Juzefovic, D., Kurcerenko, V., Kuziakin, O., and Filatov, N. (1984). Solution of problems of optimal design and selection of model parameters using the package of applied programs minimum. *Optimal Decision Theory, Institute of Mathematics and Cybernetics, Vilnius, Lithuania*, 10:77–98. (in Russian).

[Ermoljev and Wets, 1988] Ermoljev, Y. and Wets, R.-B. (1988). *Numerical Techniques for Stochastic Optimization*. Springer-Verlag, Berlin- New York-London.

[Fishburn, 1964] Fishburn, P. (1964). *Decision and Value Theory*. Wiley, New York.

[Floudas et al., 1999] Floudas, C., Pardalos, P., Adjiman, C., Esposito, W., Z.H. Gu, u., Harding, S., Klepeis, J., Meyer, C., and Schweiger, C. (1999). *Handbook of test problems in local and global optimization*. Kluwer Academic Publishers, Dordrecht-Boston-London.

[Forgo et al., 1999] Forgo, F., Szep, J., and Szidarovszky, F. (1999). *Introduction to the Theory of Games*. Kluwer Academic Publishers.

[Fox and Taqqu, 1986] Fox, R. and Taqqu, M. (1986). Large-sample properties of parameter estimates for strongly dependent stationary gaussian time series. *Annals of Statistics*, 14:517–532.

[Friedman, 1971] Friedman, A. (1971). *Dfferential Games*. Wiley-Interscience, New York.

[Galperin and Zheng, 1991] Galperin, E. and Zheng, Q. (1991). *Global Solutions in Optimal Control and Games*. Presses de l'Universite du Quebec a Montreal, Montreal.

[Geweke and Porter-Hudak, 1983] Geweke, J. and Porter-Hudak, S. (1983). The estimation and application of long memory time series models. *Journal of Time Series Analysis*, 4:221–238.

[Glover, 1994] Glover, F. (1994). Tabu search: improved solution alternatives. In *Mathematical Programming. State of the Art 1994*, pages 64–92. University of Michigan.

[Gnedenko and Kovalenko, 1987] Gnedenko, B. and Kovalenko, I. (1987). *Introduction to the Theory of Queuing*. Nauka, Moscow. (in Russian).

[Goldberg, 1989] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA.

[Greicius and Luksys, 1999] Greicius, A. and Luksys, R. (1999). Conversion from c++ to java of the flow-shop model. Technical report, Vytautas Magnus University, Faculty of Informatics, Kaunas, Lithuania.

[Grybauskas, 1998] Grybauskas, M. (1998). Global minimizer for java (gmj) version 1.1. Technical report, Institute of Mathematics and Informatics, Akademijos 4, Vilnius, Lithuania.

[Hajdu, 1997] Hajdu, M. (1997). *Network Scheduling Techniques for Construction Project Management*. Kluwer Academic Publishers.

[Hajek, 1975] Hajek, O. (1975). *Pursuit Games*. Academic Press, New York.

[Helman et al., 1993] Helman, P., Moret, B., and Shapiro, H. (1993). An exact characterization of greedy structures. *SIAM Journal of Discrete Mathematics*, 6:274–283.

[Herings, 1994] Herings, P. (1994). A gobally and universally stable price ajustment process. Technical Report 9452, Center for Economic Research, Tilburg University, The Netherlands, Tilburg.

[Himmelblau, 1972] Himmelblau (1972). *Applied Nonlinear Programming*. McGraw-Hill.

[Holand, 1975] Holand, J. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, An Arbor, MI.

[Hosking, 1981] Hosking, J. (1981). Fractional differencing. *Biometrika*, 68:165–176.

[Hosking, 1984] Hosking, J. (1984). Modeling persistence in hydrological time series using fractional differencing. *Water Resources Research*, 20:1898–1908.

[Isaacs, 1965] Isaacs, R. (1965). *Dfferential Games*. Wiley, New York.

[Janacek, 1982] Janacek, G. (1982). Determining the degree of differencing for time series via the long spectrum. *Journal of Time Series Analysis*, 3:177–188.

[Jankauskas, 2000] Jankauskas, D. (2000). Scales model in java. Technical report, Institute of Mathematics and Informatics, Akademijos 4, Vilnius, Lithuania.

[Kajokas, 1997] Kajokas, V. (1997). Java1.1 software for the bride problem. Technical report, Kaunas Technological University, Faculty of Informatics, Kaunas, Lithuania. (in Lithuanian).

[Kakutani, 1941] Kakutani, S. (1941). A generalization of brouwer's fixed point theorem. *Duke Journal of Mathematics*, 8:457–459.

[Ko, 1991] Ko, K.-I. (1991). *Complexity Theory of Real Functions*. Birkhauser, Boston.

[Koop et al., 1994] Koop, G., Ley, E., Osiewalski, J., and Steel, M. F. (1994). Bayesian analysis of long memory and persistence using arfima models. Technical report, Department of Economics, University of Toronto.

[Krapauskas, 1997] Krapauskas, M. (1997). Java1.0 software for global optimization. Technical report, Kaunas Technological University, Faculty of Informatics, Kaunas, Lithuania. (in Lithuanian).

[Krasovski, 1970] Krasovski, N. (1970). *Game Problems on the encounter motions*. Nauka, Moscow. (in Russian).

[Kushner, 1964a] Kushner, H. (1964a). A new method of locating the maximum point of an arbitrary multi-peak curve in the presence of noise. *J. of Basic Engineering*, 86:97–100.

[Kushner, 1964b] Kushner, H. (1964b). A versatile stochastic model of a function of unknown and varying form. *J. of Mathematical Analysis and Applications*, 5:150–167.

[Kuzminaite, 1999] Kuzminaite, V. (1999). Conversion from c++ to java of the algorithm of the arma model. Technical report, Vytautas Magnus University, Faculty of Informatics, Kaunas, Lithuania.

[Lemke, 1965] Lemke, C. (1965). Bimatrix equilibrium points and mathematical programming. *Management Science*, 11:681–689.

[Li and McLeod, 1986] Li, W. and McLeod, A. I. (1986). Fractional time series modeling. *Biometrica*, 73:217–221.

[Liu, 1989] Liu, J. (1989). On the existence of general multiple bilinear time series. *Journal of Time Series Analysis*, 10:341–355.

[Malisauskas, 1998] Malisauskas, V. (1998). Software for the knapsack problem. Technical report, Kaunas Technological University, Faculty of Informatics, Kaunas, Lithuania. (in Lithuanian).

[Margelevicius, 1999] Margelevicius, M. (1999). Conversion from c++ to java of the algorithm of nonlinear programming. Technical report, Kaunas Technological University, Faculty of Informatics, Kaunas, Lithuania.

[Matulevicius, 1999] Matulevicius, R. (1999). Differential game model. Technical report, Vytautas Magnus University, Faculty of Informatics, Kaunas, Lithuania.

[Mavridou et al., 1998] Mavridou, T., Pardalos, P., Pitsoulis, L., and Resende, M. (1998). A GRASP for the biquadratic assignment problem. *European Journal of Operations Research*, 105:613–621.

[Michael, 1976] Michael, J. (1976). *The Computation of Fixed Points and Applications*. Springer-Verlag, Berlin.

[Mikenas, 1998] Mikenas, M. (1998). Java1.1 software for the bay-a-car version of the bride problem. Technical report, Kaunas Technological University, Faculty of Informatics, Kaunas, Lithuania. (in Lithuanian).

[Miller and Pekny, 1991] Miller, D. and Pekny, J. (1991). Exact solution of large asymmetric travelling salesman problems. *Science*, 251:754–761.

[Mockus and Mockus, 1990] Mockus, A. and Mockus, L. (1990). Design of software for global optimization. *Informatica*, 1:71–88.

[Mockus, 1967] Mockus, J. (1967). *Multi-modal Problems in Engineering Design*. Nauka, Moscow. (in Russian).

[Mockus, 1989a] Mockus, J. (1989a). *Bayesian approach to global optimization*. Kluwer Academic Publishers, Dordrecht-London-Boston.

[Mockus, 1989b] Mockus, J. (1989b). Bayesian approach to global optimization and application to constrained and multi-objective problems. In *Abstracts, the Fifth International Conference on Stochastic Programming in Ann Arbor, August 13-18, 1989*.

[Mockus, 1997] Mockus, J. (1997). A set of examples of global and discrete optimization: application of bayesian heuristic approach ii. *Informatica, ISSN 0868-4952*, 8(4):495–526.

[Mockus, 2000] Mockus, J. (2000). *A Set of Examples of Global and Discrete Optimization: Application of Bayesian Heuristic Approach*. Kluwer Academic Publishers. ISBN 0-7923-6359-0.

[Mockus et al., 1997] Mockus, J., Eddy, W., Mockus, A., Mockus, L., and Reklaitis, G. (1997). *Bayesian Heuristic Approach to Discrete and Global Optimization*. Kluwer Academic Publishers, ISBN 0-7923-4327-1, Dordrecht-London-Boston.

[Mockus and Soofi, 1995] Mockus, J. and Soofi, A. (1995). The long-run economic relationships: An optimization approach to fractional integrated and bilinear time series. *INFORMATICA*, 6:61–70.

[Nash, 1950] Nash, J. (1950). Equilibrium points in n-person games. *Proc. Nat. Acad. Sci. USA*, 36:48–49.

[Nash, 1951] Nash, J. (1951). Noncooperative games. *Annals of Mathematics*, 54:286–295.

[Nemajunas and Remeikis, 1999] Nemajunas, I. and Remeikis, N. (1999). Conversion from c++ to java of the school scheduling model. Technical report, Vytautas Magnus University, Faculty of Informatics, Kaunas, Lithuania.

[Neuman and Morgenstern, 1953] Neuman, J. and Morgenstern, O. (1953). *Theory of Games and Economic Behavior*. Princeton University Press, Princeton.

[Owen, 1968] Owen, G. (1968). *Game Theory*. W.B. Saunders, Philadelphia, PA.

[Perlibakas, 1999] Perlibakas, V. (1999). Software for the walras problem. Technical report, Kaunas Technological University, Faculty of Informatics, Kaunas, Lithuania. (in Lithuanian).

[Petrikis and Juodgalvyte, 1997] Petrikis, S. and Juodgalvyte, J. (1997). Java1.1 software for the portfolio problem. Technical report, Vytautas Nagnus University, Faculty of Informatics, Lithuania, Kaunas. (in Lithuanian).

[Petrosyan, 1985] Petrosyan, L. (1985). *Cooperative Dfferential Games*. Tomsk University Publishing, Tomsk, Russia. (in Russian).

[Petrosyan, 1993] Petrosyan, L. (1993). *Dfferential Games of Pursuit*. Worls Scientific Publishing Co., River Edge, NJ.

[Rao and Gabr, 1984] Rao, T. S. and Gabr, M. (1984). An introduction to bispectral analysis and bilinear time series models. In *Lecture Notes in Statistics*, number 24. Springer- Verlag, Berlin.

[Rastrigin, 1968] Rastrigin, L. (1968). *Statistical Methods of Search*. Nauka, Moscow. (in Russian).

[Raudys and Mockus, 1999] Raudys, A. and Mockus, J. (1999). Comparison of arma and multilayer perceptron based methods for economic time series. *Informatica, ISSN 0860-4952*, 10(2):231–243.

[Rodin, 1987] Rodin, E. Y. (1987). A pursuit-evasion bibliography. *International Journal of Computers and Mathematics with Applications*, 13:275–340.

[Rosenmuller, 1981] Rosenmuller, J. (1981). *The Theory of Games and Markets*. North-Holand, Amsterdam.

[Saltenis, 1971] Saltenis, V. (1971). On a method of multi-extremal optimization. *Automatics and Computers (Avtomatika i Vychislitelnayya Tekchnika)*, (3):33–38. (in Russian).

[Saltenis, 1989] Saltenis, V. (1989). *Analysis of Structure of Multi-modal problems*. Mokslas, Vilnius, Lithuania. (in Russian).

[Schittkowski, 1985] Schittkowski, K. (1985). A fortran subroutine solving constrained nonlinear programming problems. *Annals of Operations Research*, 5:485–500.

[Senkiene, 1980] Senkiene, E. (1980). Properties of conditional mean and variance of the wiener process in the presence of noise, and the convergence of bayesian optimization algorithms. In *Theory of Optimal Decision*. Institute of Mathematics and Cybernetics, Akademia Nauk Lithuanian SSR, Vilnius, Lithuania. (in Russian).

[Sobolj, 1967] Sobolj, I. (1967). On a systematic search in a hypercube. *SIAM Journal on Numerical Analysis*, 16:790–793.

[Sowel, 1992] Sowel, F. (1992). Maximum likelihood estimation of stationary univariate fractionally integrated models. *Journal of Econometrics*, 53:165–188.

[Sviderskaite, 2003] Sviderskaite, A. (2003). Software for the walras problem, walras profit analysis. Technical report, Kaunas Technological University, Faculty of Informatics, Kaunas, Lithuania. (in Lithuanian).

[Tiesis, 1975] Tiesis, V. (1975). The method of variable metrics for local optimization of functions of many variables with rectangular constraints. In *Proceedings of the Conference on Computers*, pages 111–114, Kaunas, Lithuania. (in Russian).

[Tijms, 1994] Tijms, H. (1994). *Stochastic Models. An Algorithmic Approach*. Wiley, New York.

[Torn and Zilinskas, 1989] Torn, A. and Zilinskas, A. (1989). *Global optimization*. Springer-Verlag, Berlin.

[Torn, 1990] Torn, A. A. (1990). Topographical global optimization. Technical report, Abo Akademi University, Dept. of Computer Science, Turku, Finland.

[Treigys, 2003] Treigys, P. (2003). Software for the walras problem. Technical report, Vilnius Gedimino Technical University, Vilnius, Lithuania. (in Lithuanian).

[Vaisnora, 2001] Vaisnora, E. (2001). Investigation of the inspection model. Technical report, Kaunas Technological University, Kaunas, Lithuania.

[Vitek and Jensen, 2000] Vitek, J. and Jensen, C. (2000). *Secure Internet Programing*. Springer, Berlin, Heidelberg.

[von Neumann and Morgenstern, 1953] von Neumann, J. and Morgenstern, O. (1953). *Theory of Games and Economic Behaviour*. Princeton University Press, Princeton, New Jersey.

[Wald, 1947] Wald, A. (1947). *Sequential Analysis*. J. Wiley, New York.

[Wald, 1950] Wald, A. (1950). *Statistical Decision Functions*. J. Wiley, New York.

[Walras, 1874] Walras, L. (1874). *Elements d'Economie Politique Pure*. LKorbaz and Company, Lousanne.

[Yin-Wong and Lai., 1993] Yin-Wong, C. and Lai., K. (1993). Long-run purchasing power parity during the recent float. *Journal of International Economics*, 34:181–192.

Index