A small tour of optimization models

Theory of games and markets with examples

Jonas Mockus

Institute of mathematics and Informatics

Kaunas technological university

Lithuania

Introduction

- Press on CTRL-L to go to/leave full screen view.
- Curious? Want to go directly to the last page? Push here.

OPTIMIZATION AND APPLICATIONS

Consulting:
jonas2@optimum2.mii.lt
Web-sites:
$http://soften.ktu.lt/\tilde{}mockus$
$http://pilis.if.ktu.lt/\tilde{}jmockus$
$http://eta.ktl.mii.lt/\ mockus$
$http://mockus.us/optimum$ (short)

Optimality

Objective function

$$f(x),\ x = (x_1, ..., x_m). \tag{1}$$

Global minimum

$$f(x_A) \leq f(x)\ x \in A, \tag{2}$$

or

$$x_A = \arg\min_A f(x). \tag{3}$$

Here A is a feasible region.

Local minimum

$$f(x_\epsilon) \leq f(x)\ x \in \epsilon. \tag{4}$$

Here ϵ is a vicinity of x_ϵ.

Simplex algorithm

Simple example:

$$min_x z \tag{5}$$

$$z = x_1 - x_3 \tag{6}$$

$$x_1 + x_2 + x_3 = 1, x_i \geq 0 \tag{7}$$

Here x_2 base variables,
$x_1,\ x_3$ free variables. $x_1 = 1$ base solution obtained when free variables are equal to zero
$x_1 = x_3 = 0$, then $z = 0$.
This base solution is improved by increasing x_3, because $c_3 = -1 < 0$.
New base solution $x_3 = 1, x_1 = x_2 = 0$, where $z = -1$, can't be improved,
since both free variables are non-negative $c_1 = 1, c_2 = 0$.

Knapsack problem

Integer Linear Programming

$$\max_x \sum_{i=1}^{m} c_i x_i, \tag{8}$$

$$\sum_{i=1}^{m} g_i x_i \leq g, \tag{9}$$

$$x_i = \{0, 1\}. \tag{10}$$

Here c_i is price of the object i and g_i is weight of object i, g is weight limit.
$x_i = 1$ means to take object i, $x_i = 0$ means to leave it.
For small scale problems Branch-and-Bounds are used, for large scale heuristics are applied, as usual.

Method of Branch-and-Bounds

Example is the knapsack problem:

$$\max_x \sum_{i=1}^{m} c_i x_i \tag{11}$$

$$\sum_{i=1}^{m} g_i x_i \leq g, \; x_i = \{0, 1\}. \tag{12}$$

Calculating Bounds

Bounds are obtained by the auxiliary LP problems:

$$C_1 = c_1 + \max_x \sum_{i=2}^{m} c_i x_i \tag{13}$$

$$g_1 + \sum_{i=1}^{m} g_i x_i \leq g \tag{14}$$

$$C_0 = \max_x \sum_{i=2}^{m} c_i x_i \tag{15}$$

$$\sum_{i=2}^{m} g_i x_i \leq g \tag{16}$$

Branch 1- take the object, branch 0- keep the object.
If $C_0 < c_1$- cut branch 0, if $C_0 > c_1$- keep branch 0.

Worst case

Worst case is when
$c_i = c,\ g_i = g_i,\ i = 1, ..., m$.
Here no branch is cut and all $N = 2^m$ knapsacks are regarded.
Difficult case is when
$h_i = c_i/g_i = h,\ i = 1, ..., m$.
Here just a few branches are cut.
Favorable case is when h_i differ.
Here many branches are cut.

Heuristic methods

$$\max_x \sum_{i=1}^{m} c_i x_i, \tag{17}$$

$$\sum_{i=1}^{m} g_i x_i \leq g, \; x_i = \{0, 1\}. \tag{18}$$

Here heuristics

$$h_i = c_i / g_i. \tag{19}$$

Greedy heuristics is to take the best object

$$i^* = \arg\max_i h_i. \tag{20}$$

Here $N \leq m^2$ if h_i differ. If $h_i = const$- the greedy heuristic don't work, randomization is applied.

Mixing heuristics

Example is the knapsack problem.
Randomized heuristic means taking object i with probability r_i.
Traditional randomization

$$r_i = h_i / \sum_j h_j. \tag{21}$$

works well if h_i differ. If all h_i are equal
this means uniform random search.
Example of mixed randomization:

$$r_i^0 = 1/m, \tag{22}$$

$$r_i^1 = h_i / \sum_j h_j, \tag{23}$$

$$r_i^\infty = 1, \ jei \ h_i = \max_j h_j. \tag{24}$$

Optimizing mixture of heuristics

Example is a lottery of three heuristics: $x = (x_0, x_1, x_2)$.
Here
x_0 is a probability to "win" the greedy heuristic,
x_1 is a probability to "win" the randomized greedy heuristic,
x_2 is a probability to "win" the Monte Carlo search.
Lottery x is optimized using Bayesian algorithm searching for x providing best average results after K iterations.
That is a simple example of Bayesian Heuristic Approach (BHA).
Another example of BHA is school scheduling where parameters of Simulated Annealing (SA) are optimized.
Third example of BHA is flow-shop problem where mixture of three heuristics is optimized.

Optimization complexity

Algorithm is polynomial if the computing time $T = Cm^K$.
Here K is an integer and m is complexity.
In discrete problems m is number of variables.
In continuous problems m is accuracy,
meaning that error $\epsilon \leq 2^{-m}$.
Algorithm is exponential, if the computing time $T \geq C2^m$.
Important are NP-complete problems.
Simplest examples are knapsack and traveling salesman problems.
No polynomial algorithm is known for NP-complete problems.
However there is no proof that exponential algorithm is needed.

Complexity examples

Linear programming is polynomial:
simplex algorithm is exponential but interior point is polynomial.
Knapsack, flow-shop and traveling salesmen problems all are NP-complete.
Global optimization of continuous functions is exponential, in general.
Here computing time
$T \geq C2^{mn}$,
where
m is accuracy,
n is number of variables.