

# **A small tour of optimization models**

## *Theory of games and markets with examples*

Jonas Mockus

Institute of mathematics and Informatics  
Kaunas technological university  
Lithuania

# The idea

"The best way to learn is to do;  
the worst way to teach is to talk.  
The best way to teach  
is to make students ask, and do.  
Do not preach facts - stimulate to act".

P. HALMOS, The problem of learning to teach,  
Amer. Math. Monthly 82  
1975, 750-758

# OPTIMIZATION AND APPLICATIONS

Consulting:

jonas2@optimum2.mii.lt

Web-sites:

*[http : //soften.ktu.lt/~mockus](http://soften.ktu.lt/~mockus)*

*[http : //pilis.if.ktu.lt/~jmockus](http://pilis.if.ktu.lt/~jmockus)*

*[http : //eta.ktl.mii.lt/ mockus](http://eta.ktl.mii.lt/mockus)*

*[http : //mockus.us/optimum](http://mockus.us/optimum) (short)*

Textbook (in Lithuanian):

A. Žilinskas

"Matematinis Programavimas"

Kaunas, VDU, 2000

# Optimality

## Objective function

$$f(x), \quad x = (x_1, \dots, x_m). \quad (1)$$

## Global minimum

$$f(x_A) \leq f(x) \quad x \in A, \quad (2)$$

or

$$x_A = \arg \min_A f(x). \quad (3)$$

Here  $A$  is a feasible region.

## Local minimum

$$f(x_\epsilon) \leq f(x) \quad x \in \epsilon. \quad (4)$$

Here  $\epsilon$  is a vicinity of  $x_\epsilon$ .

# Discrete and convex optimization

## Discrete optimization

if variables  $x_i$  are discrete.

## Linear programming

if

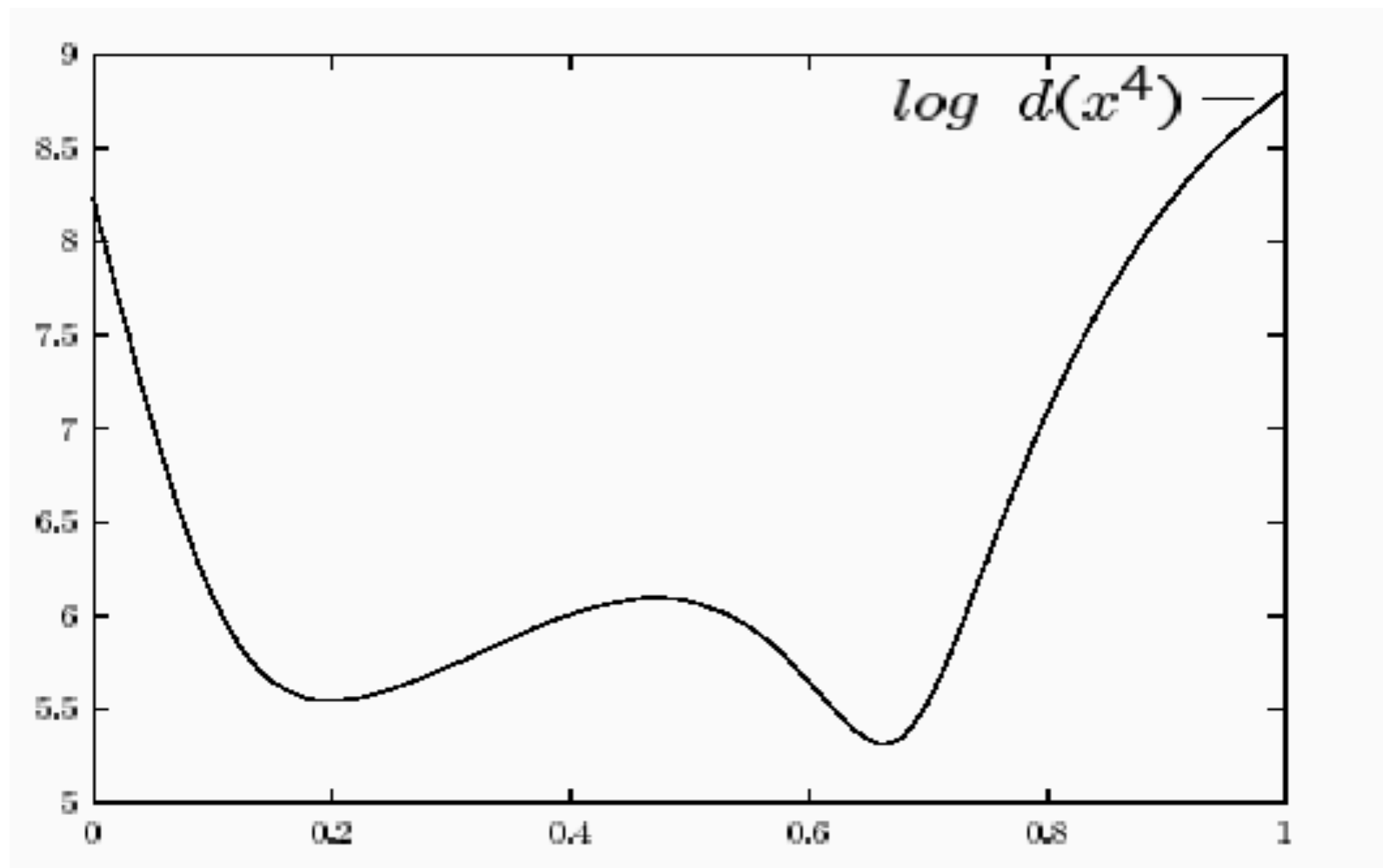
$$f(x) = \sum_{i=1}^m c_i x_i, \quad (5)$$

$$A : \sum_{i=1}^m a_{ij} x_i \geq b_j, \quad j = 1, \dots, m, \quad x_i \geq 0. \quad (6)$$

## Convex programming

if both objective  $f(x)$  and feasible region  $A$  are convex.

# Fig. 1: Global and local minima



# Global and local minima

Local minimum  $x = 0.2$ ,

global minimum  $x = 0.66$ .

Function  $f(x)$ : - multi-modal in  $[0.0, 1.0]$ ,

- convex in  $[0.0, 0.3]$ ,  $[0.6, 0.8]$ ,

- uni-modal in  $[0.0, 0.48]$ ,  $[0.48, 1.0]$ .

# Theory of games and markets

## **Search for equilibrium**

A contract with no incentives to brake is equilibrium.

Examples: models of competition, inspection and duel.

## **Prediction**

Examples:

- rates of currency and stocks,
- calls of call-center. **Optimal investment**

Examples:

- portfolio problem,
- optimal insurance.

# Discrete programming

## **Optimal scheduling**

Examples:

- flow-shop,
- school schedule.

## **Sequential decisions**

Examples:

- bride,
- buy-a-PC,
- buy-a-car.

# Nash equilibrium

## Competing servers, Nash version

- Profit of server  $i$ :

$$u_i = a_i y_i - x_i, \quad i = 1, \dots, m, \quad (7)$$

$$\sum_{i=0}^m a_i = a, \quad (8)$$

where  $a$  rate of customers.

- Customer goes to server  $i^*$ , if

$$h_{i^*} \leq h_i, \quad i = 0, \dots, m, \quad (9)$$

$$h_i = y_i + \gamma_i, \quad \gamma_i = n_i/x_i. \quad (10)$$

- $y_i$  service price,  $\gamma_i$  cost of waiting.

# Optimal contract

- A contract is optimal if no incentives to break.  
Denote the contract vector  $(\bar{y}_i, \bar{x}_i)$ . Then the "no-contract vector":

$$(\bar{y}_i, \bar{x}_i) = \arg \max_{(y_i, x_i)} u_i(y_i, x_i, \bar{y}_j, \bar{x}_j, j \neq i), \quad (11)$$

- Contract  $z = (\bar{y}_i, \bar{x}_i, i = 1, \dots, m)$  is stable if

$$\min_z f(z) = 0, \quad (12)$$

$$f(z) = \sum_{i=1}^m (u_i(\bar{y}_i, \bar{x}_i) - u_i(\bar{y}_i, \bar{x}_i)). \quad (13)$$

# Stable coalitions

Coalition is stable if no incentives to abandon,  $m = 3$ .

$S_1$  is "coalition" of single server,

$S_2$  is coalition of two servers,

$S_3$  is monopole- coalition of all three servers.

The profit is divided equally

$$u_i(S_3) = (u_i + u_j + u_k)/3,$$

$$u_i(S_2) = (u_i + u_j)/2,$$

$$u_i(S_1) = u_i.$$

Monopole is stable if

$$u_i(S_3) \geq \max\{u_i(S_2), u_i(S_1)\}.$$

Free individual competition is stable if

$$u_i(S_1) \geq \max\{u_i(S_2), u_i(S_3)\}.$$

# Walras problem

Denote  $y = (y_i, i = 1, \dots, m)$ ,  $p = (p_i, i = 1, \dots, m)$ ,  $x = (x_{ij}, i, j = 1, \dots, m)$ . Here  $x_{ii}$  are local resources,  $x_{ij}$  are server  $j$  resources used by  $i$ . Profit of  $i$

$$u_i(y, p, x) = a_i y_i + p_i \sum_{j \neq i} x_{ji} - \sum_{j \neq i} p_j x_{ij}, \quad (14)$$

Customer expences

$$h_i = y_i + \gamma_i, \quad \gamma_i = n_i / w_i, \quad w_i = c_{0i} (1 - e^{-c_{ii} x_{ii} - c_{ij} x_{ij}}), \quad (15)$$

where  $w_i$  is capacity of server  $i$ ,  $b_i$  is resource of server  $i$ ,  $c_{ij}$  defines efficiency of resources, here local resources  $x_{ii}$  are defined by balance condition:

$$x_{ii} + \sum_j x_{ij} = b_i, \quad i = 1, \dots, m, \quad (16)$$

# Optimal contract in resources $x$ , $m=2$

Denote the contract resources as  $x_{12}$  and  $x_{21}$ . Then "no-contract" resources:

$$x_{12}^{\bar{\bar{}}} (y, p) = \arg \max_{x_{12}} u_1(y, p, x_{12}, x_{21}^{\bar{\bar{}}}) \quad (17)$$

$$x_{21}^{\bar{\bar{}}} (y, p) = \arg \max_{x_{21}} u_2(y, p, x_{21}, x_{12}^{\bar{\bar{}}}) \quad (18)$$

Denote

$$\begin{aligned} f_x(y, p, x_{12}^{\bar{\bar{}}}, x_{21}^{\bar{\bar{}}}) &= u_1(y, p, x_{12}^{\bar{\bar{}}}, x_{21}^{\bar{\bar{}}}) - u_1(y, p, x_{12}, x_{21}^{\bar{\bar{}}}) \\ &\quad + u_2(y, p, x_{21}^{\bar{\bar{}}}, x_{12}^{\bar{\bar{}}}) - u_2(y, p, x_{21}, x_{12}^{\bar{\bar{}}}) \end{aligned} \quad (19)$$

Contract  $(x_{12}^{\bar{\bar{}}}(y, p), x_{21}^{\bar{\bar{}}}(y, p))$  is stable if

$$\min_{x_{12}, x_{21}} f_x(y, p, x_{12}^{\bar{\bar{}}}, x_{21}^{\bar{\bar{}}}) = 0, \quad (20)$$

Optimal resources:  $x_{12}^{\star} = x_{12}(y, p)$  and  $x_{21}^{\star} = x_{21}(y, p)$ .

# Optimal contract in prices (y,p), m=2

Denote the contract prices  $(\bar{y}_1, \bar{p}_1, \bar{y}_2, \bar{p}_2)$ . Then the "no-contract" prices:

$$(\bar{y}_1, \bar{p}_1) = \arg \max_{(y_1, p_1)} u_1(y_1, \bar{y}_2, p_1, \bar{p}_2, x_{12}(\bar{y}_2, \bar{p}_2), x_{21}(y_1, p_1)), \quad (21)$$

$$(\bar{y}_2, \bar{p}_2) = \arg \max_{(y_2, p_2)} u_2(y_2, \bar{y}_1, p_2, \bar{p}_1, x_{12}(y_2, p_2), x_{21}(\bar{y}_1, \bar{p}_1)) \quad (22)$$

$$\min_{\bar{y}_1, \bar{p}_1, \bar{y}_2, \bar{p}_2} f(\bar{y}_1, \bar{p}_1, \bar{y}_2, \bar{p}_2) = 0, \quad (23)$$

$$\begin{aligned} f(\bar{y}_1, \bar{p}_1, \bar{y}_2, \bar{p}_2) = & \quad (24) \\ & u_1(\bar{y}_1, \bar{y}_2, \bar{p}_1, \bar{p}_2, x_{12}(\bar{y}_2, \bar{p}_2), x_{21}(\bar{y}_1, \bar{p}_1)) - \\ & u_1(\bar{y}_1, \bar{y}_2, \bar{p}_1, \bar{p}_2, x_{12}(\bar{y}_2, \bar{p}_2), x_{21}(\bar{y}_1, \bar{p}_1)) + \\ & u_2(\bar{y}_2, \bar{y}_1, \bar{p}_2, \bar{p}_1, x_{12}(\bar{y}_2, \bar{p}_2), x_{21}(\bar{y}_1, \bar{p}_1)) - \\ & u_2(\bar{y}_2, \bar{y}_1, \bar{p}_2, \bar{p}_1, x_{12}(\bar{y}_2, \bar{p}_2), x_{21}(\bar{y}_1, \bar{p}_1)). \end{aligned}$$

# Walras problem, graph error

Relation of profit  $u_1(p_1)$  to resource price  $p_1$   
(other variables are fixed as the contract prices  $(\bar{y}_1, \bar{y}_2, \bar{p}_2)$ ):

**The correct relation:**

$$u_1(p_1) = a_1\bar{y}_1 + p_1x_{21}(\bar{y}_1, \bar{y}_2, p_1, \bar{p}_2) - p_2x_{12}(\bar{y}_1, \bar{y}_2, p_1, \bar{p}_2), \quad (25)$$

**The observed error:**

$$u_1(p_1) = a_1\bar{y}_1 + p_1x_{21}(\bar{y}_1, \bar{y}_2, \bar{p}_1, \bar{p}_2) - p_2x_{12}(\bar{y}_1, \bar{y}_2, \bar{p}_1, \bar{p}_2), \quad (26)$$

(the same error is in  $u_2(p_2)$ ).

# Walras problem, optimization error

Denote the contract prices  $(\bar{y}_1, \bar{p}_1)$ .

**The correct "no-contract" prices:**

$$\begin{aligned} & (\bar{y}_1, \bar{p}_1) = \\ \arg \max_{(y_1, p_1)} & (a_1 y_1 + p_1 x_{21}(y_1, \bar{y}_2, p_1, \bar{p}_2) - \bar{p}_2 x_{12}(y_1, \bar{y}_2, p_1, \bar{p}_2)), \quad (27) \end{aligned}$$

**The suspected error:**

$$\begin{aligned} & (\bar{y}_1, \bar{p}_1) = \\ \arg \max_{(y_1, p_1)} & (a_1 y_1 + p_1 x_{21}(y_1, \bar{y}_2, \bar{p}_1, \bar{p}_2) - \bar{p}_2 x_{12}(y_1, \bar{y}_2, \bar{p}_1, \bar{p}_2)), \quad (28) \end{aligned}$$

(the same error is suspected in  $(\bar{y}_2, \bar{p}_2)$ ).

# Inspector's problem, simple

Inspector's payoff

$$u(i, j) = \begin{cases} p_i g_i q_j, & \text{if } i = j, \\ 0, & \text{if } i \neq j. \end{cases} \quad (29)$$

Poacher payoff

$$v(i, j) = \begin{cases} -p_i g_j q_j + (1 - p_i) g_j q_j, & \text{if } i = j, \\ g_j q_j, & \text{if } i \neq j. \end{cases} \quad (30)$$

where  $p_i$  is probability to meet in forrest  $i$ ,  
 $q_i$  , is probability to kill a pray,  
 $g_i$  is the utility of pray.

# Average payoffs

Average payoffs of inspector and poacher

$$U(x, y) = \sum_{i,j} x_i u(i, j) y_j, \quad (31)$$

$$V(x, y) = \sum_{i,j} x_i v(i, j) y_j. \quad (32)$$

Here  $x_i, y_i$  are visiting probabilities of inspector and poacher where  $i$  denotes a forrest.

# Optimal inspection

Conditions of equal average payoffs

$$\sum_{j=1}^m u(i, j)y_j^0 = U, \quad i = 1, \dots, m$$

$$\sum_{i=1}^m v(i, j)x_i^0 = V, \quad j = 1, \dots, m$$

$$\sum_{i=1}^m x_i = 1, \quad \sum_{i=1}^m y_i = 1 \quad (33)$$

If there is a feasible solution  $x_i \geq 0$ ,  $y_i \geq 0$ ,  $i = 1, \dots, m$ , that is the equilibrium, if not then additional testing of equilibrium conditions is made, or the additional inequalities are introduced:  $x_i \geq \epsilon$ ,  $y_j \geq \epsilon$ .  $\epsilon > 0$ .

# Inspector's problem, extended

Inspector's payoff

$$u(i, j) = \begin{cases} p_i g_i q_j, & \text{if } i = j, \\ 0, & \text{if } i \neq j, \\ 0, & \text{if } i = \emptyset. \end{cases} \quad (34)$$

Poacher payoff

$$v(i, j) = \begin{cases} -p_i g_j q_j + (1 - p_i) g_j q_j, & \text{if } i = j, \\ g_j q_j, & \text{if } i \neq j, \\ 0, & \text{if } j = \emptyset. \end{cases} \quad (35)$$

where  $p_i$  is probability to meet in forrest  $i$ ,  
 $q_i$  is probability to kill a pray,  
 $g_i$  is the utility of pray,  
 $\emptyset$  means staying at home.

# Average payoffs, extended

Average payoffs of inspector and poacher

$$U(x, y) = \sum_{i,j} x_i u(i, j) y_j, \quad (36)$$

$$V(x, y) = \sum_{i,j} x_i v(i, j) y_j. \quad (37)$$

Here  $i, j = 1, 2, \dots, m, \emptyset$ ,

$x_i, y_i$  are probabilities of inspector and poacher actions, where  $i$  means going to forrest  $i$  or 'staying at home'.

Note that in the extended version, the payoffs do not satisfy the Nash equilibrium conditions, since they are not convex functions of strategies  $x_\emptyset, y_\emptyset$ , due to jumps at the points  $x_\emptyset = 0, y_\emptyset = 0$ .

# Optimal inspection, extended

Conditions of equal average payoffs

$$\sum_{j=1}^{m, \emptyset} u(i, j) y_j^0 = U, \quad i = 1, \dots, m, \emptyset$$

$$\sum_{i=1}^{m, \emptyset} v(i, j) x_i^0 = V, \quad j = 1, \dots, m, \emptyset$$

$$\sum_{i=1}^{m, \emptyset} x_i = 1, \quad \sum_{i=1}^{m, \emptyset} y_i = 1 \quad (38)$$

If there is a feasible solution  $x_i \geq 0$ ,  $y_i \geq 0$ ,  $i = 1, \dots, m$ , that is the equilibrium, if not then additional testing of equilibrium conditions is made, or the additional inequalities are introduced:  $x_i \geq \epsilon$ ,  $y_j \geq \epsilon$ .  $\epsilon > 0$ .

# Inspection example, simple

If  $q_i = g_i = 1$  then from (29)(30)

$$u(i, j) = \begin{cases} p_j, & \text{if } i = j \\ 0, & \text{otherwise,} \end{cases} \quad (39)$$

and

$$v(i, j) = \begin{cases} -p_i + (1 - p_i), & \text{if } i = j \\ 1, & \text{otherwise.} \end{cases} \quad (40)$$

From here

$$p_j y_j = U, \quad j = 1, \dots, m \quad (41)$$

$$\sum_{i \neq j} x_i + (1 - 2p_j)x_j = V, \quad j = 1, \dots, m \quad (42)$$

$$\sum_j y_j = 1, \quad \sum_i x_i = 1, \quad y_j \geq 0, \quad x_i \geq 0$$

# Two forests

If  $m = 2$  then from (41)

$$y_1 = p_2 / (p_1 + p_2), \quad (43)$$

$$y_2 = p_1 / (p_1 + p_2), \quad (44)$$

and

$$x_1 = p_2 / (p_1 + p_2), \quad (45)$$

$$x_2 = p_1 / (p_1 + p_2). \quad (46)$$

No 'staying at home' possibility, in these examples.

# $m \geq 2$ forrests

$$\begin{aligned}u^* &= U = p_1 p_2 / (p_1 + p_2), \\v^* &= V = p_1 + p_2 - 2p_1 p_2 / (p_1 + p_2).\end{aligned}\tag{47}$$

If  $p_1 = 1/3, p_2 = 2/3$

$$x_1^* = y_1^* = 2/3, \quad x_2^* = y_2^* = 1/3, \quad u^* = 2/9, \quad v^* = 5/9.\tag{48}$$

For any  $m \geq 2$

$$y_i = x_i = \prod_{k \neq i} p_k / \left( \sum_i \prod_{k \neq i} p_k \right),\tag{49}$$

(50)

where  $\prod_{k \neq i} p_k$  is a product of all  $p_k$  except  $p_i$ .

# Duel

Two flying objects are fighting.  
The trajectories are

$$dz(t)/dt = az(t), \quad (51)$$

$$dw(\tau)/d\tau = bw(\tau), \tau = 2 - t, \quad (52)$$

Thus

$$z(t) = z_0 e^{at}, \quad w(\tau) = w_0 e^{b\tau}. \quad (53)$$

Hitting probability

$$p(t) = 1 - d(t)/D. \quad (54)$$

Here  $d(t)$  distance between objects,  
firing time is  $t$ , maximal distance is  $D$ .

# Payoff functions

Payoff function of the first object:

$$U(t_1, t_2) = \begin{cases} p(t_1) - (1 - p(t_1)), & \text{if } t_1 < t_2, \\ -p(t_2) + (1 - p(t_2)), & \text{if } t_2 < t_1, \\ 0, & \text{if } t_2 = t_1, \end{cases}$$

Payoff function of the second object

$$V(t_1, t_2) = \begin{cases} p(t_2) - (1 - p(t_2)), & \text{if } t_2 < t_1, \\ -p(t_1) + (1 - p(t_1)), & \text{if } t_1 < t_2, \\ 0, & \text{if } t_2 = t_1. \end{cases}$$

# One-dimensional duel, "Two Knights"

Payoff function of the first knight:

$$U(t_1, t_2) = \begin{cases} t_1 - (1 - t_1), & \text{if } t_1 < t_2, \\ -t_2 + (1 - t_2), & \text{if } t_2 < t_1, \\ 0, & \text{if } t_2 = t_1, \end{cases}$$

Payoff function of the second knight

$$V(t_1, t_2) = \begin{cases} p(t_2) - (1 - t_2), & \text{if } t_2 < t_1, \\ -pt_1 + (1 - t_1), & \text{if } t_1 < t_2, \\ 0, & \text{if } t_2 = t_1. \end{cases}$$

Equilibrium:  $t_1 = t_2 = 0.5$ . Here  $D = 2$ , speed = 1,  
Therefore  $p(t_1) = t_1$ ,  $p(t_2) = t_2$ .

# Optimal duel

Optimal firing time:

$$p(t_1) = p(t_2) = 0.5. \quad (55)$$

Optimal initial heights  $z_0, w_0$   
and optimal ascend rates  $a, b$   
are calculated by equilibrium conditions  
using mixed strategies defined by linear programming.

# Economic duel

## Dynamical competition of two servers

Profit functions:

$$U_i(T) = \int_{t_0}^T u_i(t) dt, \quad (56)$$

$$u_i(t) = a_i(t)y_i(t) - x_i(t), \quad (57)$$

where

$u_i(t)$  is profit of server  $i$ , moment  $t$ ,

$a(t) = \sum_i a_i(t)$  is customer arrival rate.

Service price is  $y_i(t)$ , service expenses are  $x_i(t)$

Trajectories are from:

$$dy_i(t)/dt = ay_i(t), \quad (58)$$

$$dx_i(t)/dt = bw_i(t). \quad (59)$$

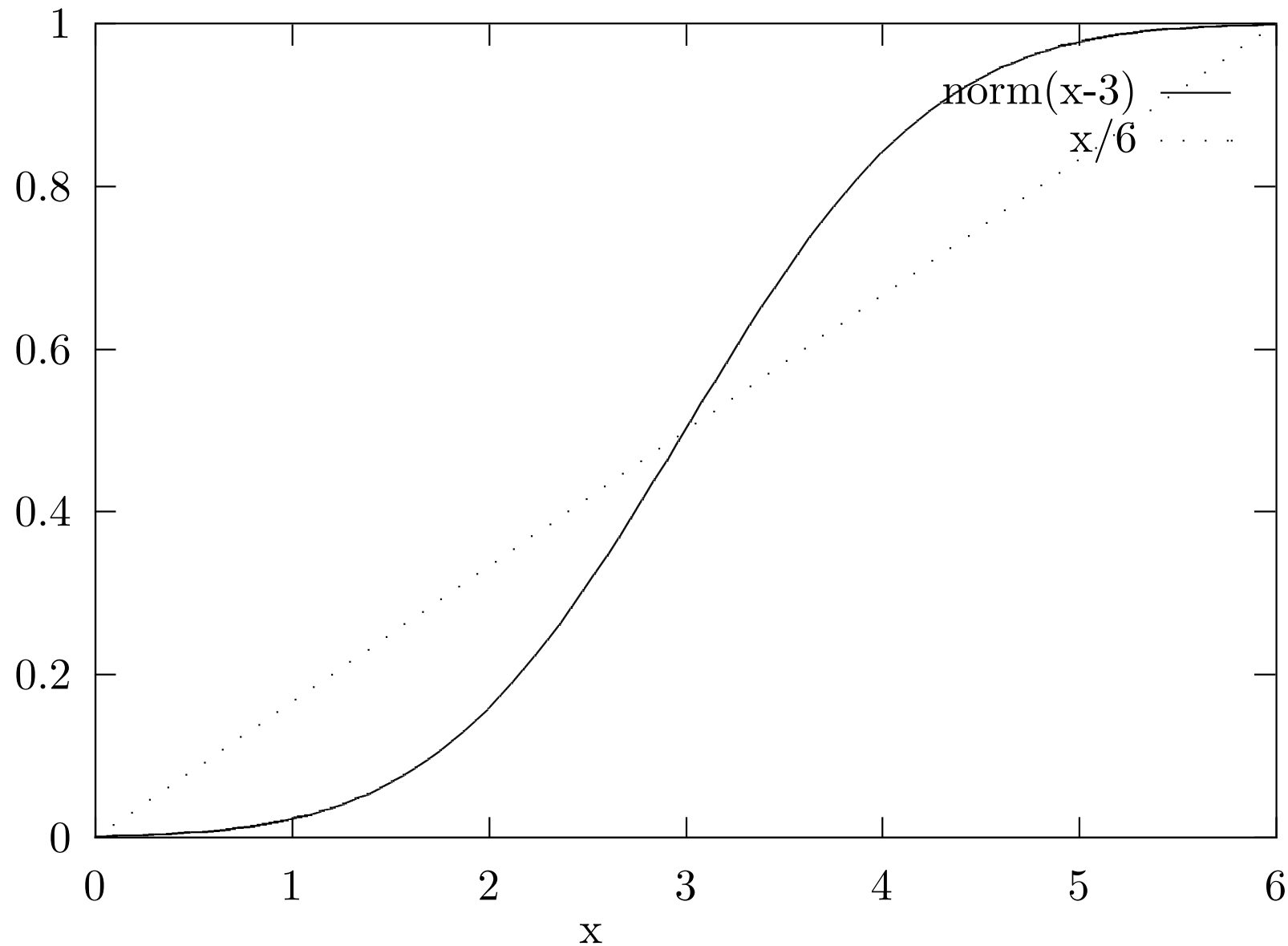
# Optimal economic duel

We search for initial values  $y_i(0), x_i(0)$ ,  
and change rates  $a, b$ ,  
that provide equilibrium.

Server  $i$  gets broken at moment  $t^*$  if  
 $U_i(t^*) < -U^*$ .

The surviving server enjoys monopolistic profit.

# Fig. 2. Utility function



# Explaining Fig. 2

Utility function of

- rich person is denoted by dots,
- average person is denoted by continuous line,
- risk area is [0.0, 3.0],
- risk aversion area is [3.0,6.0],
- investment is  $x$ ,
- total amount is 6.0

# Lottery

Utility of event  $0 \leq C \leq 1$  is denoted by  $u(C)$  and is defined using the lottery:

$$C \sim \{pA + (1 - p)B\}. \quad (60)$$

Here utility of keeping  $C$  is

$$u(C) = p,$$

where  $p$  is probability to win  $A = 1$ ,

$1 - p$  is probability to win nothing  $B = 0$ .

Symbol  $\sim$  denotes the "hesitation"

when a player don't know what is better:

- to keep the  $C$ ,
- or to risk loosing  $C$  while trying to win better  $A$ .

# Expected utility

Expected utility of investment  $x$ :

$$U(x) = \sum_{k=1}^M u(y^k) p(y^k). \quad (61)$$

Here  $y^k = \sum_{i=1}^m \delta_i c_i x_i$  is returned wealth,

$u(y^k)$  is utility of wealth  $y^k$ ,

$x_i$  is capital invested in the object  $i$ ,

$c_i = 1 + \alpha_i$ ,  $\alpha_i$  is the yield of  $i$ ,

$\delta_i = 1$  if  $i$  survives,

$\delta_i = 0$  if  $i$  gets broken,

$p(y^k)$  probability to get wealth  $y^k$ .

For example:

$$p(y^1) = p_1 \prod_{i \neq 1} (1 - p_i).$$

Here  $y^1 = c_1 x_1$ , and  $p_i$  is survival probability of  $i$ .

# Optimal insurance

Expected utility of investment  $x$

$$U(x) = \sum_{k=1}^M u(y^k) p(y^k).$$

Here  $p(y^k)$  is probability to get wealth

$y^k = \sum_{i=1}^m c_i(x_i)$ , and  $u(y^k)$  is utility of wealth  $y^k$ .

$$c_i(x_i) = \begin{cases} -a_i x_i, & \text{if } \delta_i = 1 \\ -z_i + (1 - a_i)x_i, & \text{if } \delta_i = 0. \end{cases} \quad (62)$$

# Explaining insurance variables

In the equation

$z_i$  is price of the object  $i$ ,

$a_i x_i$  is insurance cost of object  $i$ ,

$x_i \leq z_i$  is insurance sum of object  $i$ .

$\delta_i = 1$  if  $i$  survives,

$\delta_i = 0$  otherwise,

$p_i = P\{\delta_i = 1\}$  survival probability of object  $i$ .

For example:

$$p(y^1) = p_1 \prod_{i \neq 1} (1 - p_i),$$

$$y^1 = c_1(x_1) + \sum_{i=2}^m c_i(x_i), \text{ where}$$

$$c_1(x_1) = z_1 - a_1 x_1, c_i(x_i) = (1 - a_i) x_i, \quad i = 2, \dots, m.$$

# Prediction

ARMA is auto-regression moving-average model

$$w_t = \sum_{i=1}^p a_i w_{t-i} + \sum_{j=1}^q b_j \epsilon_{t-j} + \epsilon_t. \quad (63)$$

For example,

$w_t$  is stock rate tomorrow

$w_{t-1}$  is stock rate today,

$\epsilon_t$  is a random component tomorrow,

# ARMA optimization

$a_i, b_j$  are ARMA parameters defined by minimization of

$$f(x) = \sum_{t=1}^T \epsilon_t^2, \quad (64)$$

where  $x = (x_1, \dots, x_{p+q})$ ,  $x_i = a_i$ ,  $i = 1, \dots, p$ ,  $x_i = b_{i-p}$ ,  $i = p + 1, \dots, p + q$ .

# Stock-exchange game

One stock two major customers  $i = 1, 2$ ,  
Customer  $i$  is buying a stock at moment  $t$  if

$$z(t) \leq z_i^{min}(t), \quad (65)$$

where

$z(t)$  stock rate at  $t$ ,

$z_i^{min}(t)$  is a buying level.

Customer  $i$  is selling a stock at  $t$  if

$$z(t) \geq z_i^{max}(t), \quad (66)$$

where

$z_i^{max}(t)$  is a selling level.

# Stock exchange model

Stock rate at  $t + 1$  is defined  
by the price of previous deal at moment  $t$

$$z(t + 1) = \begin{cases} z(t) + \epsilon(t + 1), & \text{if } z^{\min}(t) < z(t) < z^{\max}(t), \\ z^{\min}(t) + \epsilon(t + 1), & \text{if } z(t) \leq z^{\min}(t), \\ z^{\max}(t) + \epsilon(t + 1), & \text{if } z(t) \geq z^{\max}(t). \end{cases}$$

Here

$$z^{\min}(t) = \max_i z_i^{\min}(t), \quad (67)$$

$$z^{\max}(t) = \min_i z_i^{\max}(t). \quad (68)$$

# Expected profit

Expected profit of player  $i$  at next moment  $t + 1$ :

$$\Delta_i(t + 1) = (\beta_i(t + 1) + \delta(t + 1) - \alpha(t + 1)) z(t). \quad (69)$$

Here

$$\beta_i(t + 1) = (z_i(t + 1) - z(t))/z(t), \quad (70)$$

where  $z_i(t + 1)$  is expected stock rate at moment  $t + 1$ .

$$\delta(t + 1) = d(t + 1)/z(t), \quad (71)$$

$d(t + 1)$  are expected dividends at moment  $t + 1$ .

$$\alpha(t + 1) = a(t + 1)/z(t), \quad (72)$$

$a(t + 1)$  is yield at moment  $t + 1$ .

# Buying and selling levels

Buying level

$$z_i^{min}(t) = k_{buy} \Delta_i(t), \quad k_{buy} > 1. \quad (73)$$

Selling level

$$z_i^{max}(t) = k_{sell} \Delta_i(t), \quad k_{sell} < 1. \quad (74)$$

Number of stocks own by customer  $i$  at time  $t$

$$N_i(t+1) = \begin{cases} N_i(t), & \text{if } z_i^{min}(t) < z(t) < z_i^{max}(t), \\ N_i(t) + 1, & \text{if } z(t) \leq z_i^{min}(t), \\ N_i(t) - 1, & \text{if } z(t) \geq z_i^{max}(t) \end{cases}$$

# Stock-exchange game, Model 2

One stock two major customers  $i = 1, 2$ ,  
Customer  $i$  is buying a stock at a moment  $t$  if

$$\Delta_i(t) \geq k_i^{buy}, \quad (75)$$

where

$\Delta_i(t)$  is expected profit rate at a moment  $t$ , see (80),  
Customer  $i$  is selling a stock at  $t$  if

$$\Delta_i(t) \leq k_i^{sell}, \quad (76)$$

where

$k_i^{buy} > 0$  is a buying level.

$k_i^{sell} < 0$  is a selling level.

# Stock-exchange game "Soros"

One additional customer Soros  $s$ ,  
Customer Soros  $s$  is buying a stock at a moment  $t$  if

$$z(t) \leq z^{buy}(t), \quad (77)$$

Customer Soros  $s$  is selling a stock at a moment  $t$  if

$$z(t) \geq z^{sell}(t), \quad (78)$$

Here

$z^{buy}(t)$  is a Soros buying price at a moment  $t$ .

$z^{sell}(t)$  is a Soros selling price at a moment  $t$ . More funds can be made available in this model.

# Stock exchange, 2

Stock rate at next moment  $t + 1$  is defined by the price of previous deal at this moment  $t$

$$z(t + 1) = \begin{cases} z(t) + \epsilon(t + 1), & \text{if } k_i^{sell} < \Delta_i(t) < k_i^{buy} \text{ for all } i, \\ z^{sell}(t) + \epsilon(t + 1), & \text{if } \Delta_i(t) \leq k_i^{sell} \text{ for some } i, \\ z^{buy}(t) + \epsilon(t + 1), & \text{if } \Delta_j(t) \geq k_j^{buy} \text{ for some } j, \\ z^{sell}(t) + \epsilon(t + 1), & \text{if } z(t) \geq z^{sell}(t), \\ z^{buy}(t) + \epsilon(t + 1), & \text{if } z(t) \leq z^{buy}(t). \end{cases}$$

(79)

# Expected profit rate, 2

Expected profit rate of player  $i$  at a moment  $t + 1$ :

$$\Delta_i(t + 1) = (\beta_i(t + 1) + \delta(t + 1) - \alpha(t + 1)). \quad (80)$$

Here

$$\beta_i(t + 1) = (z_i(t + 1) - z(t))/z(t), \quad (81)$$

where  $z_i(t + 1)$  is a stock rate at moment  $t + 1$  predicted by customer  $i$  using AR model.

$$\delta(t + 1) = d(t + 1)/z(t), \quad (82)$$

$d(t + 1)$  are expected dividends at moment  $t + 1$ .

$$\alpha(t + 1) = a(t + 1)/z(t), \quad (83)$$

$a(t + 1)$  is yield at moment  $t + 1$ .

# Buying and selling stocks, Single level

Number of stocks own by customer  $i$  at time  $t$

$$N_i(t+1) = \begin{cases} N_i(t), & \text{if } k_i^{sell} < \Delta_i(t) < k_i^{buy} \text{ for all } i, \\ N_i(t) + N_b, & \text{if } \Delta_i(t) \leq k_i^{sell} \text{ for some } i, \\ N_i(t) - N_s, & \text{if } \Delta_i(t) \geq k_i^{buy} \text{ for some } i. \end{cases} \quad (84)$$

Here  $N_b$  is a number of stocks to buy and  $N_s$  is a number to sell.

# Multi-level example, 2

Stock rate at  $t + 1$  is defined  
by the price of previous deal at a moment  $t$

$$z(t+1) = \begin{cases} z(t) + \epsilon(t + 1), & \text{if } k_i^{sell}(l) < \Delta_i(t) < k_i^{buy}(l) \text{ for all } i, \\ z^{sell}(t) + \epsilon(t + 1), & \text{if } \Delta_i(t) \leq k_i^{sell}(l) \text{ for some } i, l, \\ z^{buy}(t) + \epsilon(t + 1), & \text{if } \Delta_j(t) \geq k_j^{buy}(l) \text{ for some } j, l, \\ z^{sell}(t) + \epsilon(t + 1), & \text{if } z(t) \geq z^{sell}(t), \\ z^{buy}(t) + \epsilon(t + 1), & \text{if } z(t) \leq z^{buy}(t). \end{cases} \quad (85)$$

Funds and stocks of the customer Soros  $s$  are not limited.

# Buying and selling stocks, 2

Number of stocks own by customer  $i$  at time  $t$  in the multi-level mode

$$N_i(t+1) = \begin{cases} N_i(t), & \text{if } k_i^{sell}(l) < \Delta_i(t) < k_i^{buy}(l) \text{ for all } i, \\ N_i(t) + N_{i,b}(l), & \text{if } \Delta_i(t) \leq k_i^{sell}(l) \text{ for some } i, \\ N_i(t) - N_{i,s}(l), & \text{if } \Delta_i(t) \geq k_i^{buy}(l) \text{ for some } i. \end{cases} \quad (86)$$

Here

$$k_i^{buy}(l+1) > k_i^{buy}(l), \quad k_i^{sell}(l+1) < k_i^{buy}(l),$$

$$N_{i,b}(l+1) < N_{i,b}(l), \quad N_{i,s}(l+1) < N_{i,s}(l),$$

$$N_i(t+1) = N_i(t), \quad \text{if } \Delta_i(t) \leq k_i^{sell}(l) \text{ and no funds left for } i.$$

# Number of stocks for buying and selling, 2

Maximal number of stocks to buy at the time  $t$

$$N_{i,b}(t) = (C_i(t) + U_i(t, T_0))/z(t), \quad (87)$$

Here  $C_i(t)$  is credit available for customer  $i$  at time  $t$ ,  
 $U_i(t, T_0)$  is customers'  $i$  profit accumulated at time  $t$ .

Number of stocks to buy at the time  $t$  and buying level  $l$

$$N_{i,b}(t, l) = N_{i,b}/l, \quad (88)$$

Maximal number of stocks to sell at the time  $t$

$$N_{i,s}(t) = N_i(t), \quad (89)$$

Number of stocks to buy at the time  $t$  and buying level  $l$

$$N_{i,s}(t, l) = N_{i,s}/l, \quad (90)$$

$l = 1, 2, 3$  as usual.

# Customer profits, 2

Profit of customer  $i$  during time  $T_0 \leq t \leq T$ :

$$U_i(T, T_0) = \quad (91)$$

$$N_i(T)z_T - N(T_0)z_{T_0} - \sum_{t=T_0}^T (N_i(t+1) - N_i(t)) z(t).$$

Profit  $u_i$  depends on accuracy of prediction  $\beta_i(t+1)$  and random events  $\epsilon(t)$ .

If customer  $i$  predicts by AR model then

$$z_i(t+1) = \sum_{k=1}^{p_i} a_i^k z_{t-k} + \epsilon_i(t+1). \quad (92)$$

# Parameters of customer predictions

AR parameters  $a_k$  are defined by condition:

$$\min_{a_i} \sum_{s=t_0}^t \epsilon_i^2(s), \quad (93)$$

where

$$\epsilon_i(s) = z(s) - \sum_{k=1}^{p_i} a_i^k z(s - k). \quad (94)$$

This model is for stock exchange simulation by generating time series of virtual stock rates. The model is not intended for actual predictions.

# Stock exchange equilibrium

We search for AR parameters  $(p_1, p_2)$  satisfying Nash equilibrium using mixed strategies

$$x_{p_i}, \quad i = 1, 2, \quad p_i = 1, \dots, P_i, \quad \sum_{p_i=1}^{P_i} x_{p_i} = 1, \quad (95)$$
$$0 \leq x_{p_i} \leq 1.$$

Here  $x_{p_i}$  is a probability of parameter  $p_i$ .

Denote the "no-contract" vector

$$x_{p_1}^1 = \arg \max_{x_{p_1}} U_1^K(T_0, T, x_{p_1}, x_{p_2}^0), \quad (96)$$

$$x_{p_2}^1 = \arg \max_{x_{p_2}} U_1^K(T_0, T, x_{p_1}^0, x_{p_2}) \quad (97)$$

where  $x_{p_i}^0, \quad i = 1, 2$  is a "contract" vector.

# Search for Nash equilibrium

Nash equilibrium:

$$\arg \min_{x_{p_1}^0, x_{p_2}^0} \left\| (x_{p_1}^1, x_{p_2}^1) - (x_{p_1}^0, x_{p_2}^0) \right\|^2 = (x_{p_1}^*, x_{p_2}^*) \quad (98)$$

Optimizing by global stochastic methods.

If minimum not small enough

testing equilibrium conditions

by analysis of profit function convexity.

Final objective

is to define optimal AR parameters.

The results are unexpected- equilibrium is by Wiener model that means "no prediction".

# Call center model

Calls are random generated by distribution

$$F_a(t) = P\{\tau < t\}.$$

defining probability that time  $\tau$   
until next call is less than  $t$ .

Parameter  $a$  denotes call rate- average number  
of calls by a time unit.

# Generating arrival times

In the Poisson stream the arrival time of next customer is defined by condition

$$F_a(t) = 1 - e^{-1/at}, \quad (99)$$

$$\tau = -a \ln(1 - \xi). \quad (100)$$

where  $\xi$  random number uniform in  $[0,1]$ .

If service time is exponential

$$F_x(t) = 1 - e^{-1/mxt}, \quad (101)$$

$$\tau = -x \ln(1 - \xi), \quad (102)$$

where  $mx$  is service rate of  $m$  agents, then expected waiting time in Poisson stream is

$$\gamma = \frac{a}{mx(mx - a)}, \quad (103)$$

# Optimizing number of agents

If  $x$  is service rate of single agent,  
and  $m$  is number of agents  
then total service expenses are

$$c(a, m) = c_m m + c_\gamma(a)\gamma \quad (104)$$

where  $c_m$  expenses for one agent,  
 $c_\gamma(a)$  estimated cost of waiting time.  
Optimal number of agents

$$m(a) = \min_m c(a, m). \quad (105)$$

Optimizing  $m$  important is estimated rate  $a$ .

# Scale model

Predicting call rate

$$Z_i = z_{p(i)} s_{p(i)}. \quad (106)$$

where

$Z_i = (z_{ij}, j = 1, \dots, m)$  predicted graph for day  $i$ ,

$z_{ij}$  hour  $j$ ,

$z_{p(i)}$  average of day  $p(i)$ ,

$p(i)$  defines a previous day similar to the day  $i$ .

# Similarity conditions

For example,  
similar are days such as Sunday, Saturday, working day.  
suPanašūs būna sekmadieniai, šeštadieniai.  
Thus assumption that the hourly graph of next Sunday  
will be similar to previous Sunday,  
the scale is defined by

$$s_{p(i)} = \frac{z_{i-1}}{z_{p(i-1)}}. \quad (107)$$

Here  $z_{i-1}$  average of this day,  
 $z_{p(i-1)}$  average of similar previous day.

# Optimal scheduling

## Flow-shop problem

Here the sequence of tools is defined by technology for example in tailors shop: scissors, needle, and iron.

Operation times  $\tau_{ij}$  define time to perform a task  $i$  by a tool  $j$

For example, in tailor shop  $\tau_{ij}$  is a time to cut a suit  $i$  using scissors  $j$ .

Objective is make-span minimization

In small scale flow-shop problems

we can optimize by comparing all task sequences, for example: 2, 1, 4, 6, 5, 3, 7.

Optimizing large scale flow-shop problems heuristic methods are applied, as usual.

Convergence can be provided by randomization.

# School scheduling

Here tasks represent academic classes. Tools are topics.

School resources are teachers and class-rooms.

Sequence of tools is free.

Objective is minimization of "penalty points". Penalty points define deviation from the "perfect schedule".

In the web-site examples the initial school schedule is improved by permutations. Optimization is made by Simulated Annealing with parameters optimized using Bayesian approach.

# Sequential decisions

**Dynamic programming**

**"Single bride" problem**

No divorce. Number of proposals is  $N$ .

Goodness of grooms  $\omega$  is defined by Gaussian distribution:

$$p(\omega) = \frac{1}{\sqrt{2\pi}\sigma_0} e^{-1/2\left(\frac{\omega-\alpha_0}{\sigma_0}\right)^2}, \quad (108)$$

where  $\alpha_0$  is average goodness ,

$\sigma_0^2$  is goodness variance.

Brides impression is Gaussian, too:

$$p(s|\omega) = \frac{1}{\sqrt{2\pi}\sigma} e^{-1/2\left(\frac{s-\omega}{\sigma}\right)^2}. \quad (109)$$

here  $\sigma$  error of brides judgment.

# Posterior goodness

Posterior goodness of grooms regarding impression  $s$

$$p(\omega|s) = \frac{p(s|\omega)p(\omega)}{\int_{-\infty}^{\infty} p(s|\omega)p(\omega)d\omega}. \quad (110)$$

Expected goodness of grooms making impression  $s$

$$u(s) = \int_{-\infty}^{\infty} \omega p(\omega|s)d\omega, \quad (111)$$

where  $p(\omega)$  is density of probability. If  $\omega$  and  $s$  discrete

$$u(s_j) = \sum_i \omega_i P(\omega_i|s_j), \quad (112)$$

where  $P(\omega_i|s_j)$  is probability of goodness given  $s = s_j$ .

# Example of posterior probability

Prior probability of rain  $p(r) = 0.5$ , that of clear  $p(c) = 0.5$ .  
Observed probabilities of wrong/right predictions:

$$p(s = c|r) = 0.1, p(s = r|r) = 0.9,$$
$$p(s = r|c) = 0.2, p(s = c|c) = 0.8.$$

The posterior probability of wrong rain prediction

$$p(r|s = c) = \frac{p(s = c|r)p(r)}{p(s = c|r)p(r) + p(s = c|c)p(c)} = \quad (113)$$
$$0.1 * 0.5 / (0.1 * 0.5 + 0.8 * 0.5) = 0.109.$$

The posterior probability of wrong clear prediction

$$p(c|s = r) = \frac{p(s = r|c)p(c)}{p(s = r|c)p(c) + p(s = r|r)p(r)} = \quad (114)$$
$$0.2 * 0.5 / (0.2 * 0.5 + 0.9 * 0.5) = 0.182.$$

# Bride's decision function

Expected goodness of the last proposal  $N$

$$u_N(s) = u(s), \quad (115)$$

because bride must marry at least once  
by problem definition.

Optimal decision  $d_{N-1}(s)$  for  $(N - 1)$ -th proposal:

$$u_{N-1}(s) = \max_d (du(s) + (1 - d)u_N), \quad (116)$$

$$d_{N-1}(s) = \operatorname{arg} \max_d (du(s) + (1 - d)u_N). \quad (117)$$

Optimal decision for  $(N - n)$ -th proposal  
is defined in a similarly.

This way bride's decision function is build.

This function shows how the optimal decision  
depends on the impression  $s$  made by the proposal  $n$ .

# Simple example-1

Probability density  $p$  of goodness  $\omega$

$$p(\omega) = \begin{cases} 0.5, & \text{if } -1 < \omega < 1, \\ 0, & \text{otherwise} \end{cases} \quad (118)$$

Brides' observation  $s = \omega$  Then expected goodness of the last proposal  $N$

$$u_N(s) = u(s) = s, \quad (119)$$

Optimal decision  $d_{N-1}(s)$  for  $(N - 1)$ -th proposal:

$$u_{N-1}(s) = \max_d (d * s + (1 - d) * 0), \quad (120)$$

$$d_{N-1}(s) = \mathit{arg} \max_d (d * s + (1 - d) * 0). \quad (121)$$

# Simple example-2

From here the optimal decision

$$d_{N-1}^*(s) = \begin{cases} 1, & \text{if } s > 0, \\ 0, & \text{if } s < 0 \end{cases} \quad (122)$$

and the optimal goodness

$$u_{N-1}^* = \max(0, s)$$

the expected optimal goodness at  $N - 1$

$$Eu_{N-1}^* = \int_{-1}^1 \max(0, s)p(s)ds = \quad (123)$$

$$0.5 \int_0^1 s ds = 0.25 \quad (124)$$

# Simple example-3

Optimal decision for  $(N - 2)$ -th proposal

$$u_{N-2}(s) = \max_d (d * s + (1 - d) * 0.25), \quad (125)$$

$$d_{N-2}(s) = \operatorname{arg} \max_d (d * s + (1 - d) * 0.25). \quad (126)$$

From here

$$d_{N-1}(s) = \begin{cases} 1, & \text{if } s > 0.25, \\ 0, & \text{if } s < 0.25 \end{cases} \quad (127)$$

The remaining steps are defined similarly.

This way bride's decision function is build.

This function shows how the optimal decision

depends on the impression  $s$  made by the current proposal.

# "Free" bride (Buy-a-PC)

## Decision function

Bride is free to marry and to divorce  $N$  times.

Denote groom's (new PC) goodness by  $\omega$ .

Assume "clairvoyant" bride

that defines goodness  $\omega = s$  exactly.

Goodness of actual husband (old PC) denote by  $q$ .

Expected "goodness" of the  $N$ -th proposal:

$$u_N(\omega, q_N) = \max_d (d\omega + (1 - d)q_N). \quad (128)$$

Here the optimal decision depends on both components: goodness of husband (keeping the old PC)  $q$  and goodness of proposal (new PC)  $\omega$ :

$$d_N^* = \begin{cases} 1, & \text{if } \omega_N > q_N - c_N, \\ 0, & \text{if } \omega_N \leq q_N - c_N \end{cases} \quad (129)$$

# Utility goodness

## **N-th proposal:**

The utility of the decision  $d = 0$ , to keep the old PC in the last year  $N$ , is  $q_N - c_N$ .

Here  $q_N$  is the utility of old PC,  $c_N = \tau_N - g_0(N)$  is the penalty of refusing to buy a new PC. This includes the waiting losses  $\tau_N$  minus the price  $g_0(N)$  of new PC in the year  $N$ .

It is assumed that we abandon the old PC as soon as we obtain the new one. Therefore, one "wins" the price  $g_0(N)$  of the new PC by using the old PC.

Bride is free to marry and to divorce  $N$  times.

# Utility vector- linear approximation

**N-th proposal:** Define PC parameters by a vector  $g = (g_0, g_1, g_2, g_3)$ . Here  $g_0$  is the market price of PC in \$,  $g_1$  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

$$\omega = a_1g_1 + a_2g_2 + a_3g_3. \quad (130)$$

Here  $a_i$  is user evaluations of utility of quality parameter  $g_i$  expressed in \$ per unit. The users evaluation  $\omega$  differs from the market price  $g_0$ , as usual.

The PC utility defined as a sum of utilities of different components is just first approximation. Therefore, we need to evaluate different PC configurations separately.

# Utility function

## **N-th proposal:**

The problem is how to define user evaluation based on the utility theory.

Step 1: define a set of events  $E$  as a set of PC described by different feasible vectors  $(g_1, g_2, g_3, g_4)$

Step 2: define a sequence of events  $E_i, i = 1, \dots, I$  ordered by the condition  $E_{i-1} \leq E_i \leq E_{i+1}$ . Condition  $E_I \leq E_{i+1}$  means that we prefer PC  $E_{i+1}$  to  $E_i$ .

Step 3: set the normalized utility functions

$$u_0(E_1) = 0, u_0(E_I) = 1$$

Step 4: define the remaining utility functions  $u_0(E_i) = p_i$  where  $p_i$  is the 'hesitation' probability determined by the lottery:

$$E_i \sim \{p_i E_I + (1 - p_i) E_1\}. \quad (131)$$

# Relation of quality and price

**N-th proposal:** Define by  $h$  the highest price a user is ready to pay for the best PC. Then the general utility  $u(E_i)$  of the PC of lesser configuration  $E_i$  is

$$u(E_i) = h * u_0(E_i) - g_{0i} \quad (132)$$

Or, in case of linear approximation:

$$u(E_i) = \omega_i - g_{0i} \quad (133)$$

where  $\omega_i$  is goodness of PC  $E_i$  calculated by (130) and  $g_{0i}$  is the market price.

# Maximization of expected goodness

Regarding  $(N - i)$ -th proposal

Optimal goodness depends on  $q$  and  $\omega$ .

$$u_{N-i}(\omega, q) = \max_d (d\omega + (1 - d)(u_{N-i+1}(q_{N-i+1}) - c_{N-i})).$$

Here  $u_{N-i+1}(q)$  is expected goodness of  $(N - i + 1)$ -th proposal given husband goodness  $q$ .

$$u_{N-i+1}(q) = \int_{-\infty}^{\infty} u_{N-i+1}(\omega, q) p_{N-i+1}(\omega) d\omega. \quad (134)$$

# Decision function

**Regarding  $(N - i)$ -th proposal**

$p_{N-i+1}(\omega)$  is probability of goodness  $\omega$  of  $(N - i + 1)$ -th proposal.

$$q_{N-i+1} = \begin{cases} \omega_{N-i}, & \text{if } q_{N-i+1} < q_{N-i}^*, \\ q_{N-i}, & \text{if } q_{N-i+1} \geq q_{N-i}^* \end{cases}$$

Here  $q_{N-i}^*$  is defined by

$$\omega_{N-i} = u_{N-i+1}(q_{N-i}^*) - c_{N-i}. \quad (135)$$

**Optimal decision regarding  $(N - i)$ -th proposal**

$$d_{N-i}^* = \begin{cases} 1, & \text{if } \omega_{N-i} > u_{N-i+1}(q_{N-i+1}) - c_{N-i}, \\ 0, & \text{if } \omega_{N-i} \leq u_{N-i+1}(q_{N-i+1}) - c_{N-i} \end{cases}$$

Here decision function  $d_{N-i}^*$  depends on both  $\omega$  and  $q$ .

# Diet problem

$$\min_x \sum_{i=1}^m ((c_i - s_i)x_i + g(\sum_{i=1}^m a_{i1}x_i - b_1)), \quad (136)$$

$$\sum_{i=1}^m a_{ij}x_i \geq b'_j, \quad \sum_{i=1}^m a_{ij}x_i \leq b''_j, \quad j = 1, \dots, n \quad (137)$$

$$x_i \geq 0, \quad i = 1, \dots, m. \quad (138)$$

For example,

$c_1$  is a price of bread,  $a_{11}$  are calories of bread,  $b'_1$  are necessary calories,  $b''_1$  are harmful calories,  $s_i$  is the taste (expressed in money units),  $g$  beauty factor (expressed in money units).

Second inequality (137) is not included into LP, it is a warning.

# Diverse diet problem

$$\min_x \sum_{i=1}^m ((c_i - s_i)x_i + g(\sum_{i=1}^m a_{i1}x_i - b_1) - d(\sum_{i=1}^m d_i x_i)), \quad (139)$$

$$\sum_{i=1}^m a_{ij}x_i \geq b'_j, \quad \sum_{i=1}^m a_{ij}x_i \leq b''_j, \quad j = 1, \dots, n \quad (140)$$

$$x_i \geq 0, \quad i = 1, \dots, m. \quad (141)$$

Here

$d$  is the taste of food diversity (expressed in money units),

$d_i$  is the dish indicator:  $d_i = 1$ , if  $x_i$  is a dish,  $d_i = 0$ ,

otherwise,

If  $x_i$  is a dish then  $x_i = int$  and  $0 \leq x_i \leq 1$ .

# Longer term diet problem

$$\min_x \sum_{i=1}^m ((c_i - s_i)x_i + g(\sum_{i=1}^m a_{i1}x_i - Tb_1) - d(\sum_{i=1}^m d_i x_i)) \quad (142)$$

$$\sum_{i=1}^m a_{ij}x_i \geq Tb'_j, \quad \sum_{i=1}^m a_{ij}x_i \leq Tb''_j, \quad j = 1, \dots, n \quad (143)$$

$$\sum_{i=1}^m d_i x_i \leq T, \quad \sum_{i=1}^m d_i \geq T. \quad (144)$$

Here  $d$  is the taste of food diversity,  $T$  is the time period,  $d_i$  is the dish indicator:  $d_i = 1$ , if  $x_i$  is a dish, otherwise  $d_i = 0$ . If  $x_i$  is a dish then  $x_i = int$  and  $0 \leq x_i \leq 1$ .

# Combinatorial diet problem

$$\min_x \left( \sum_{i=1}^m (c_i - s_i)x_i + g \left( \sum_{i=1}^m a_{i1}x_i - Tb_1 \right) - d(x) + b \sum_{j=1}^n (B'_{j+} + B''_{j+}) \right), \quad (145)$$

Here

$$B'_j = \left( - \sum_{i=1}^m a_{ij}x_i + Tb'_j \right), \quad B''_j = \left( \sum_{i=1}^m a_{ij}x_i - Tb''_j \right), \quad (146)$$

$B'_{j+}, B''_{j+}$  are positive parts of  $B'_j, B''_j$ ,  $d(x)$  is a function defining the taste of food diversity (for example,  $d(x)$  could be a number of different non-zero components  $x_i$  in the diet  $x$ ),  $b$  is a penalty factor for violation constraints. (143)

# Solving combinatorial diet problem

Step 1. Set an initial diet  $x = x^1$  and evaluate its quality by calculating the sum:

$$D(x^1) = \left( \sum_{i=1}^m (c_i - s_i)x_i^1 + g \left( \sum_{i=1}^m a_{i1}x_i^1 - Tb_1 \right) - d(x^1) + b \sum_{j=1}^n (B'_{j+} + B''_{j+}) \right) \quad (147)$$

Step 2. Generate next diet  $x^2$  by changing randomly some dishes and evaluate the quality  $D(x^2)$ .

Step 3. If  $h_2 = D(x^2) - D(x^1) \leq 0$  go to  $x^2$ .

If  $h_2 > 0$  go to  $x^2$  with probability  $r_2$ ;

keep  $x^1$  and return to step 1 with probability  $1 - r_2$ .

# Combinatorial diet problem-2

Probability  $r_2$  is generated by SA formula 227 defined in the slide "Simulated Annealing"

Parameter  $x$  of SA is optimized using GMJ system.

Bayesian Heuristic Approach (BHA) is recommended method for improving a user defined initial diet by optimizing parameters of Simulated Annealing (SA).

# Defining SA parameters

Step 1. Generate probability  $r_2$  by SA formula 173.

Step 2. Optimize parameter  $x$  of SA by GMJ.

The standart reference to GMJ is this:

```
public Domain domain ()
```

```
return domain;
```

```
public double f (Point pt)
```

```
.....
```

```
return f
```

Here 'domain' defines constraints of SA variables,

'pt' is vector of SA variables,

'f' is the 'goodness' function of the best diet

after 'IT' iterations using fixed SA variables.

# Simplex algorithm

Simple example:

$$\min_x z \quad (148)$$

$$z = x_1 - x_3 \quad (149)$$

$$x_1 + x_2 + x_3 = 1, x_i \geq 0 \quad (150)$$

Here  $x_2$  base variable,

$x_1, x_3$  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

$$\max_x \sum_{i=1}^m c_i x_i, \quad (151)$$

$$\sum_{i=1}^m g_i x_i \leq g, \quad (152)$$

$$x_i = \{0, 1\}. \quad (153)$$

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:

$$\max_x \sum_{i=1}^m c_i x_i \quad (154)$$

$$\sum_{i=1}^m g_i x_i \leq g, \quad x_i = \{0, 1\}. \quad (155)$$

# Calculating Bounds

Bounds are obtained by the auxiliary LP problems:

$$C_1 = c_1 + \max_x \sum_{i=2}^m c_i x_i \quad (156)$$

$$g_1 + \sum_{i=2}^m g_i x_i \leq g \quad (157)$$

$$C_0 = \max_x \sum_{i=2}^m c_i x_i \quad (158)$$

$$\sum_{i=2}^m g_i x_i \leq g \quad (159)$$

Branch 1- take the object, branch 0- leave the object.

If  $C_0 < c_1$ - cut branch 0, if  $C_0 \geq c_1$ - keep branching.

# Worst case

Worst case is when

$$c_i = c, g_i = g, i = 1, \dots, m.$$

Here no branch is cut and all  $N = 2^m$  knapsacks are regarded.

Difficult case is when

$$h_i = c_i/g_i = h, i = 1, \dots, m.$$

Here just a few branches are cut.

Favorable case is when  $h_i$  differ.

Here many branches are cut.

# Heuristic methods

$$\max_x \sum_{i=1}^m c_i x_i, \quad (160)$$

$$\sum_{i=1}^m g_i x_i \leq g, \quad x_i = \{0, 1\}. \quad (161)$$

Here heuristics

$$h_i = c_i / g_i. \quad (162)$$

Greedy heuristics is to take the best object

$$i^* = \arg \max_i h_i. \quad (163)$$

Here  $N \leq m^2$  if  $h_i$  differ. If  $h_i = \text{const}$ - 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

$$r_i = h_i / \sum_j h_j. \quad (164)$$

works well if  $h_i$  differ. If all  $h_i$  are equal this means uniform random search.

Example of mixed randomization:

$$r_i^0 = 1/m, \quad (165)$$

$$r_i^1 = h_i / \sum_j h_j, \quad (166)$$

$$r_i^\infty = 1, \text{ } j \in i \text{ } h_i = \max_j h_j. \quad (167)$$

# 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.

# Local optimization

## Descent methods

Objective is minimization of function

$f(x)$ ,  $x = (x_i, i = 1, \dots, m)$ ,

$$x^{n+1} = x^n - \alpha_n s_n. \quad (168)$$

Here  $\alpha$  is step size and  $s_n$  is step direction.

$$\alpha_n = \arg \min_{\alpha} f(x^n - \alpha_n s_n), \quad (169)$$

where  $n$  is iteration number.

Simple case is gradient algorithm when

$$s_n = \text{grad } f(x^n). \quad (170)$$

Here the gradient is

$$\text{grad } f(x^n) = \left( \frac{\partial f(x^n)}{\partial x_i}, i = 1, \dots, m \right). \quad (171)$$

# Newton's method

In Newton's method the step direction is

$$s_n = H_n^{-1} \text{grad } f(x^n), \quad (172)$$

where the Hessian

$$H_n = \left( \frac{\partial^2 f(x^n)}{\partial x_i \partial x_j}, i, j = 1, \dots, m \right). \quad (173)$$

# Quasi-Newton's method

In Quasi-Newton's (variable Metrics) method step direction is

$$H_n^{-1} \approx G_n, \quad (174)$$

where

$$G_{n+1} = G_n - \frac{(G_n \gamma_n)(G_n \gamma_n)^T}{\gamma_n^T G_n \gamma_n} + \frac{\delta_n \delta_n^T}{\delta_n^T \gamma_n}. \quad (175)$$

Here symbol  $T$  denotes transposition,

$$\gamma_n = \text{grad } f(x_n) - \text{grad } f(x_{n-1}), \quad (176)$$

$$\delta_n = x^n - x^{n-1}, \quad (177)$$

$$G_0 = I. \quad (178)$$

# Local convergence

Gradient method converges,  
if  $f(x)$  is a convex differentiable function.

Newton's and Quasi-Newton's methods converge, if  $f(x)$   
is convex twice-differentiable function.

Gradient method converges slowly:

$$\|x^n - x^*\| \rightarrow 0. \quad (179)$$

Newton's and Quasi-Newton's methods converge fast:

$$\frac{\|x^n - x^*\|}{\|x^{n-1} - x^*\|} \rightarrow 0, \quad (180)$$

where  $x^*$  is the optimum.

The main difficulty of Newton's method is calculation of  
inverse Hessian  $H_n^{-1}$ . Newton's method fails if  $\det H_n = 0$ .

No inverse Hessian is needed using Quasi-Newton's.

# Constrained optimization

$$\max_x f_0(x), \quad (181)$$

$$f_j(x) \leq c_j, \quad j = 1, \dots, n. \quad (182)$$

Penalty function:

$$\max_x \{f_0(x) - \sum_j b_j (f_j(x) - c_j)^2\}, \quad (183)$$

$$b_j = \begin{cases} b, & \text{jei } f_j(x) > c_j, \\ 0, & \text{jei } f_j(x) \leq c_j \end{cases} \quad (184)$$

It is difficult to define right penalty factor  $b$ : if  $b$  is small then constraints are violated, if  $b$  is great then we optimize the penalties instead of the original function  $f(x)$ .

# Lagrange multipliers

Constrained optimization

$$\max_x f_0(x), \quad (185)$$

$$f_j(x) \leq c_j, \quad j = 1, \dots, n. \quad (186)$$

Optimization of Lagrange function

$$\min_{\lambda > 0} \max_x \left\{ f_0(x) - \sum_j \lambda_j (f_j(x) - c_j) \right\}. \quad (187)$$

# Economics of Lagrange multipliers

$$\min_{\lambda > 0} \max_x \left\{ f_0(x) - \sum_j \lambda_j (f_j(x) - c_j) \right\}. \quad (188)$$

Economic interpretation of Lagrange function:

$f(x)$  is profit of factory owner,  $c_j$  is available resource  $j$ ,

$x$  is technology chosen by factory owner,

$\lambda_j$  is the price for additional resource  $j$  set by owner of the resource,

$\min_{\lambda > 0}$  maximizes profit of resource owner,

$\max_x$  maximizes profit of factory owner at fixed prices  $\lambda$  for additional resources.

If all  $f_j(x)$ ,  $j = 0, \dots, n$  are convex then minimization of Lagrangian provides minimum of original constrained optimization problem.

# Stochastic Gradient

Minimize

$$\min_x f(x). \quad (189)$$

(190)

When we observe the sum:

$$\phi(x) = f(x) + \xi \quad (191)$$

where  $\xi$  is Gaussian noise  $(0, \sigma)$ .

Then the stochastic gradient:

$$x^{n+1} = x^n - a_n/s_n (\psi(x^n + s_n) - \psi(x^n)). \quad (192)$$

# Stochastic Gradient, Convergence

The sequence  $\{x_n\}$  converges with probability 1 to the minimum of convex differentiable function  $f(x)$  if:

$$\lim_{n \rightarrow \infty} s_n = 0, \quad (193)$$

$$\lim_{n \rightarrow \infty} \sum_{i=1}^n a_i / s_i = \infty, \quad (194)$$

$$\lim_{n \rightarrow \infty} \sum_{i=1}^n (a_i / s_i)^2 = C < \infty. \quad (195)$$

where  $n$  is iteration number.

For example,

$$s_n = 1/n, \quad a_n = 1/n^2$$

# Stochastic Linear Programming 1

$$x + 2y = c \quad (196)$$

$$x + y = a \quad (197)$$

where  $c$  is cost,  $x$  is production,  $a \geq 0$  is uncertain demand,  $y$  is amount to buy.

Expected cost:

$$f(x) = xP(a \leq x) + 2(a - x)P(a \geq x) \quad (198)$$

where  $P(a \leq x)$  is probability that demand will not exceed production.

Suppose that probability density

$$p(a) = \begin{cases} 1, & \text{if } 0 \leq a \leq 1, \\ 0, & \text{otherwise} \end{cases} \quad (199)$$

# Stochastic Linear Programming 2

From here:

$$f(x) = \int_0^x x da + \int_x^1 2(a - x) da = \quad (200)$$

$$x^2 + 1 - x^2 - 2x(1 - x) = 1 - 2x + 2x^2 \quad (201)$$

minimum  $f(x) = 1/2$  is reached when  $x = 1/2$   
and is equal to expected demand:

$$Ea = \int_0^1 a da = 1/2$$

but not always, they say

# Global continuous optimization

## Uniform grid

$$r_n = \max_x \min_i \|x - x^i\|. \quad (202)$$

Grids  $x(n) = (x^i, i = 1, \dots, n)$  that minimize  $r_n^0 = \min_{x^i, i=1, \dots, n} r_n$ , provide best accuracy for Lipschitz functions

$$\frac{\|f(x^i) - f(x_j)\|}{\|x^i - x_j\|} \leq L < \infty, \quad (203)$$

where  $L$  is Lipschitz constant. Here best accuracy means minimization of maximal deviation  $\epsilon$  from the global optimum

$$\epsilon = LR_n, \quad (204)$$

$$R_n = \arg \min_{x(n)} r_n. \quad (205)$$

# Pareto-Lipschitzian optimality (PO)

Objective is to minimize Lipschitz-function  $f_L(x)$  with unknown Lipschitz constant  $L$ .

The decision  $x$  dominates the decision  $x^*$  if

$$f_L(x) \leq f_L(x^*), \text{ for all } L \quad (206)$$

$$f_L(x) > f_L(x^*), \text{ for at least one } L \quad (207)$$

The decision  $x^*$  is Pareto Optimal (PO) if there is no dominant  $x$ .

# Pareto-Lipschitzian optimization (PLO)

The variables  $x$  are represented by the intervals  $i : a_i \leq x \leq b_i$  and the function  $f_L(x)$  is approximated by the lower bounds  $f(x)$

$$f_L(x) \leq f(c_i) - L * l_i/2, \quad a_i \leq x \leq b_i. \quad (208)$$

Here  $c_i = (b_i + a_i)/2$ ,  $l_i = b_i - a_i$ . The interval  $i : a_i \leq x \leq b_i$  dominates the interval  $j$ , if

$$f(c_i) - L * l_i/2 \leq f(c_j) - L * l_j/2, \quad \text{for all } L \quad (209)$$

$$f(c_i) - L * l_i/2 < f(c_j) - L * l_j/2, \quad \text{for at least one } L \quad (210)$$

The interval  $j$  is Pareto Optimal (PO) if there is no dominant interval  $i$ .

# Examples of (PLO)

## *Example 1*

The lengths:

$$l_1 = 3, l_2 = 2, l_3 = 1.$$

The function values:

$$f(c_1) = 3, f(c_2) = 2, f(c_3) = 2.$$

The intervals 1 *and* 3 are PO, the interval 2 is not PO.

## *Example 2*

The lengths:

$$l_1 = 3, l_2 = 2, l_3 = 1.$$

The function values:

$$f(c_1) = 3, f(c_2) = 2, f(c_3) = 1.$$

Here all the intervals 1 *and* 2 *and* 3 are PO.

# Defining user preferences

A convenient way to represent user preferences is by supplying an importance measure to each multi-criteria component  $L$ . Since the set of the Lipschitz functions are continuous, the proper measure would be the probability density  $p(L)$ . Then the PO interval

$$i(p) = \arg \min_i \int_L (f(c_i) - L * l_i/2)p(L)dL = \arg \min_i (f(c_i) - E\{L\} * l_i/2). \quad (211)$$

were  $E\{L\}$  is the expected value of the Lipschitz constant  $L$ .

If, for example  $p(L) = \exp(-L)$  then  $E\{L\} = 1$ . In standard applications the criteria set is discrete, so, the integral is replaced by sum.

# Building nearly-uniform grids

In multi-dimensional global optimization problems building exactly uniform grids is difficult.

Then "LPtau" or "Monte Carlo" approximations are used.

LPtau grids provide nearly-uniform coordinate projections.

Monte Carlo grids generate coordinates  $x_i, i = 1, \dots, n$  independently by uniform probability distributions.

# Bayesian methods

Uniform grids minimize maximal deviation by using

$T = C2^{mn}$  of computing time, where

$n$  is number of variables,

$m$  defines the accuracy by the condition  $\epsilon \leq 2^{-m}$ .

This is expensive if  $m$  or  $n$  are large.

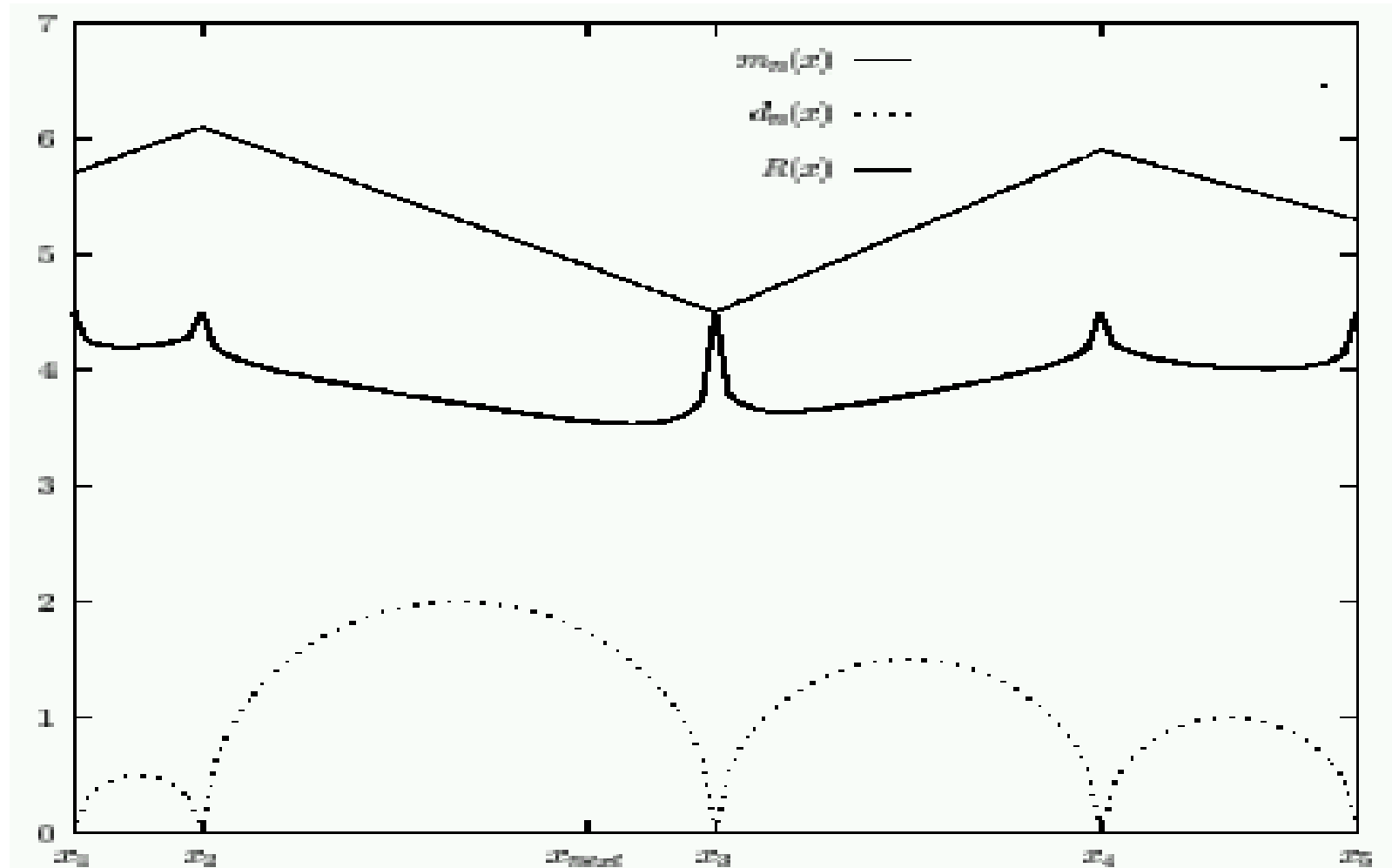
Then Bayesian methods are applied.

Bayesian methods minimize expected deviation for a given iteration number.

Defining expected deviation statistical models of objective functions are needed.

Simple one-dimensional example is the Wiener model.

# Fig. 2. Wiener model



# Wiener model optimization

In Fig. 2.  $m_n(x)$  is the conditional mean,  
 $d_n(x)$  is the conditional variance,  
 $R(x)$  is the risk function,  
Bayesian method minimizes the risk function

$$x^{n+1} = \underset{x}{\operatorname{arg\,min}} R(x). \quad (212)$$

# Risk function

In the Wiener model the risk function

$$R(x) = \frac{1}{\sqrt{(2\pi d_n(x))}} \int_{-\infty}^{+\infty} \min(c_n, y_x) e^{-\frac{(y_x - m_n(x))^2}{d_n(x)}} dy_x$$

Here

$$y_x = f(x),$$

$$c_n = \min_i f(x^i) - \epsilon, \quad \epsilon > 0.$$

Note that

$$R(x) = c_n, \text{ if } x = x^i$$

since

$$d_n(x^i) = 0.$$

# Coordinate optimization

Wiener model is applied for coordinate optimization when  $m > 1$ ,

$$x^1 = \mathop{\text{arg min}}_{x_1} f(x^0) \quad (213)$$

$$x^2 = \mathop{\text{arg min}}_{x_2} f(x^1) \quad (214)$$

.....

$$x^m = \mathop{\text{arg min}}_{x_m} f(x^{m-1}) \quad (215)$$

.....

Here results depend on initial points  $x^0$ .

However the coordinate optimization is a convenient tool of visualization by one-dimensional projections. Coordinate optimization by Wiener model converges if

$$f(x) = \sum_i f_i(x_i) \text{ or } f(x) = \prod_i f_i(x) \quad (216)$$

# Multi-dimensional Bayesian method

The point of next observation (calculation of  $f(x)$ ):

$$x^{n+1} = \operatorname{arg\,min}_x R(x) \quad (217)$$

$$R(x) = y_{0n} - \min_i \frac{\|x - x_i\|^2}{f(x_i) - c_n}, \quad (218)$$

$$c_n = \min_i f(x^i) - \epsilon, \quad \epsilon > 0. \quad (219)$$

When  $n$  is large

$$d^*/d_a = \left( \frac{f_a - f^* + \epsilon}{\epsilon} \right)^{1/2}$$

Here  $d^*$  density of observations in the vicinity of  $x^*$ ,

$f^*$  average value  $f(x)$  around  $x^*$ ,

$d_A$  average density of observations,  $f_A$  average values of  $f(x)$ .

# Vector optimization

Objective is to maximize vector-function

$$f(x) = (f_i(x), i = 1, \dots, m). \quad (220)$$

Pareto optimum is the set  $X^*$ .

$x^* \in X^*$ , if there are no such  $x$  that

$$f_i(x) \geq f_i(x^*), \quad \forall i \quad (221)$$

$$f_j(x) > f_j(x^*), \quad \exists j \quad (222)$$

# Scalarization

**By weights**

$$x(c) = \operatorname{arg\,max}_x \sum_i c_i f_i(x), \quad c_i > 0. \quad (223)$$

Here  $x(c) \in X^*$ .

**'bf By constraints**

$$x(b) = \operatorname{arg\,max}_x f_1(x) \quad (224)$$

$$f_i(x) \geq b_i, \quad i = 2, \dots, m. \quad (225)$$

Here  $x(b) \in X^*$ , if  $x(b)$  is unique,  
otherwise, non-Pareto points are possible.

# Simulated Annealing (SA)

Simulated Annealing (SA) is the most popular global optimization method Denote

$$h_j = f(x^j) - f(x^{j-1}), \quad (226)$$

if  $h_j \geq 0$ , then go to  $x^j$

if  $h_j < 0$ , then go to  $x^j$  with probability

$$r_j = \begin{cases} e^{\frac{h_j}{x/\ln(1+j)}}, & \text{if } h_j < 0, \\ 1, & \text{otherwise} \end{cases} \quad (227)$$

SA is very simple and convenient for theoretical analysis. Efficiency of SA is increased by optimization of parameters, for example by optimization of  $x$  for given family of objective functions  $f(x)$  at fixed number of iterations. Here the Bayesian approach is useful.

# Sawmil problem

Objective is to minimize waste while sawing  $i = 1, \dots, m$  planks from  $j = 1, \dots, n$  logs.

All dimensions are given.

Denote by  $h_{ij}$  priority rule used defining what plank  $i$  to saw and from which log  $j$ .

The results depends on goodness of heuristic  $h_{ij}$  and on the efficiency of algorithms providing geometric constraints. Here heuristics are not simple. Geometric constraints keeping planks inside logs are complicated.

# Formalization of sawmil problem

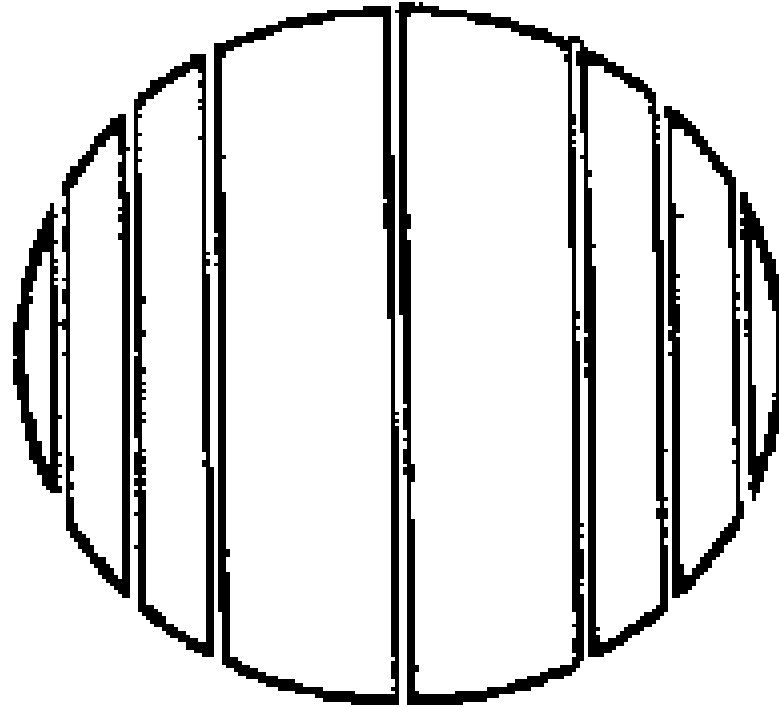
$$\begin{aligned} \min_y v(y), \quad y = (y_{ijk}), \quad (228) \\ i = 1, \dots, m, \quad j = 1, \dots, n, \quad k = 1, \dots, K, \\ a_i y_{ijk} \leq d_{ijk}. \end{aligned}$$

Here  $v(y)$  is the waste calculated as difference between the volumes of logs and planks,  $a_i$  is thickness of plank  $i$ ,  $d_{ijk}$  is thickness of log  $j$  at place  $k$  where plank  $i$  sawed.

$$y_{ijk} = \begin{cases} 1, & \text{jei } i \in (j, k), \\ 0, & \text{otherwise} \end{cases} \quad (229)$$

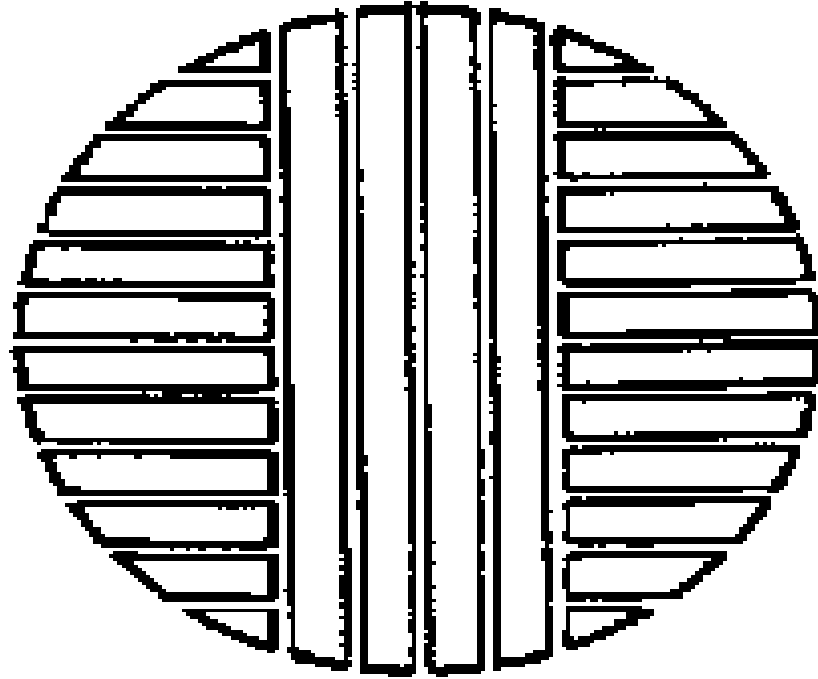
where  $i$  number of plank,  $j$  is number of log,  $k$  defines the place where plank  $i$  is sawed from log  $j$ ,  $i \in (j, k)$  means that plank  $i$  is in log  $j$  at the place  $k$ .

# Fig. 3. Complete cut



a- ištisinis

# Fig. 4. Segment cut



**c - ištinis  
segmentinis**

# Completing trains

The objective is to minimize the number of train  $n = n(y)$  for  $m$  cars

$$\min_y n(y), \quad (230)$$

$$\sum_{i=1}^m a_i y_{ij} \leq A_j, \quad (231)$$

$$\sum_{i=1}^m b_i y_{ij} \leq B_j. \quad (232)$$

Here

$$y_{ij} = \begin{cases} 1, & \text{jei } i \in j, \\ 0, & \text{otherwise.} \end{cases} \quad (233)$$

# Completing trains, notation

In the formula of completing trains:  $a_i$  is weight of car  $i$ ,  
 $A_j$  is maximal weight of train  $j$ ,  
 $b_i$  is length of car  $i$ ,  
 $B_j$  is maximal length of train  $j$ ,  
 $i \in j$  means that the car  $i$  is in the train  $j$ .

# Theory of games and applications

In the optimization part of this course the models of games and markets are regarded as examples of optimization problems.

In the game-theoretical part basic elements of theory of games and markets are explained and some additional problems are introduced.

# Simplest game, "Toss-up"

Here two players and two moves.  
Payoff matrix of the first player;

$$u(i, j) = \begin{vmatrix} 1 & -1 \\ -1 & 1 \end{vmatrix} \quad (234)$$

where  $i = 1, 2$  are moves of the first player,  
and  $j = 1, 2$  are moves of the second player.  
Payoff matrix of the second player

$$v(i, j) = \begin{vmatrix} -1 & 1 \\ 1 & -1 \end{vmatrix} \quad (235)$$

Here  $v(i, j) = -u(i, j)$ , that is a zero-sum game.

# Bayesian game, "Toss-up"

Here two players,  
first is a person, second is the "nature"  
and two moves.

Payoff matrix of the first player;

$$u(i, j) = \begin{vmatrix} 1 & -1 \\ -1 & 1 \end{vmatrix} \quad (236)$$

where  $i = 1, 2$  are moves of the first player,  
and  $j = 1, 2$  are moves of the 'nature'.

In Bayesian game nature uses mixed strategy,  
assume that  $p(1) = 1/2 + \epsilon$ ,  $p(2) = 1 - p(1)$   
then optimal strategy of the first player is 1-st row

# Pure and mixed strategies

Moves made directly are pure strategies. Moves made by random procedures are mixed strategies.

Expected payoffs are players objectives if mixed strategies are used.

$$\begin{aligned}U(x, y) &= x_1y_1 - x_1y_2 - x_2y_1 + x_2y_2 \\V(x, y) &= -x_1y_1 + x_1y_2 + x_2y_1 - x_2y_2\end{aligned}\tag{237}$$

where

$0 \leq x_i \leq 1$ ,  $i = 1, 2$  is mixed strategy of the first player (probability of making the move  $i$ ),

and  $y_i$  is mixed strategy of the second player

From here

$$\begin{aligned}U(x, y) &= 4x_1y_1 - 2x_1 - 2y_1 + 1 \\V(x, y) &= -4x_1y_1 + 2x_1 + 2y_1 - 1\end{aligned}\tag{238}$$

# Equilibrium strategies

Equilibrium are strategies with no incentives for change. In the "Toss-up" game there is no equilibrium by pure strategies. But there exists an equilibrium by mixed strategies

$$x_i = y_i = 0.5 \quad (239)$$

$$U(x_1 = 1/2, y_1 = 1/2) = 0,$$

$$V(x_1 = 1/2, y_1 = 1/2) = 0 \quad (240)$$

$$x_2 = 1 - x_1, y_2 = 1 - y_1.$$

Here mixed strategies  $x_i = y_i = 0.5$  are generated by tossing a coin. We see that

$$U(x_1 = 1/2 + \epsilon, y_1 = 1/2) < 0,$$

$$V(x_1 = 1/2, y_1 = 1/2 + \epsilon) < 0, \quad (241)$$

if  $\epsilon \neq 0$ .

# Bimatrix games

In bimatrix games  $u(i, j) \neq -v(i, j)$

Simple example is asymmetric version of the "Toss-up" game

$$u(i, j) = \begin{vmatrix} 1 & -1 \\ -1 & 1 \end{vmatrix} \quad (242)$$

$$v(i, j) = \begin{vmatrix} -1 & 1 \\ 1 & 0 \end{vmatrix} \quad (243)$$

# Strategies of bimatrix game

Expected payoff

$$\begin{aligned}U(x, y) &= x_1y_1 - x_1y_2 - x_2y_1 + x_2y_2, \\V(x, y) &= -x_1y_1 + x_1y_2 + x_2y_1,\end{aligned}\tag{244}$$

or

$$\begin{aligned}U(x, y) &= 4x_1y_1 - 2x_1 - 2y_1 + 1, \\V(x, y) &= -3x_1y_1 + x_1 + y_1.\end{aligned}\tag{245}$$

Here the equilibrium

$$y_1 = y_2 = 1/2,\tag{246}$$

$$x_1 = 1/3, x_2 = 2/3,\tag{247}$$

$$u^* = U(x_1 = 1/3, y_1 = 1/2) = 0,\tag{248}$$

$$v^* = V(x_1 = 1/3, y_1 = 1/2) = 1/3,\tag{249}$$

# Equalizing expected payoffs

Denote

$$U(1, y) = y_1 - y_2, U(2, y) = -y_1 + y_2, \quad (250)$$

$$V(x, 1) = -x_1 + x_2, V(x, 2) = x_1, \quad (251)$$

where  $U(i, y)$  expected payoff of the first player using the move  $i$  if the second uses mixed strategy  $y$ , expected payoff of the second player  $V(x, j)$  is defined similarly  
Strategies  $x, y$  can be defined using linear programming

$$U(1, y) = U, U(2, y) = U,$$

$$V(x, 1) = V, V(x, 2) = V.$$

(252)

where  $0 \leq x_i \leq 1, 0 \leq y_i \leq 1, x_1 + x_2 = 1, y_1 + y_2 = 1$ . The solution if exists satisfies equilibrium condition.

# Equilibrium in pure strategies

Equilibrium in pure strategies of of bimatrix games can be found by comparing all pairs  $(i, j)$  of pure strategies. For example:

$$u(i, j) = \begin{vmatrix} 1 & -1 \\ -1 & 1 \end{vmatrix} \quad (253)$$

$$v(i, j) = \begin{vmatrix} -1 & 1 \\ 1 & 2 \end{vmatrix} \quad (254)$$

Equilibrium strategies are  $(i = 2, j = 2)$  and payoffs are  $u(2, 2) = 1, v(2, 2) = 2$ .

Note that here is no solution of the linear programming problem equalizing expected payoffs.

# Prisoners' dilemma

1 yes, 2 no rows controlled 1-st prisoner  
columns controlled 2-cond prisoner

$$u(i, j) = \begin{vmatrix} 3 & 1 \\ 10 & 2 \end{vmatrix} \quad (255)$$

$$v(i, j) = \begin{vmatrix} 3 & 10 \\ 1 & 2 \end{vmatrix} \quad (256)$$

Here

Equilibrium (1,1)

Pareto (2,2)(2,1)(1,2)

# Family problem

1 opera, 2 soccer  
rows controlled by husband  
columns controlled by wife

$$u(i, j) = \begin{vmatrix} 10 & 0 \\ 0 & 20 \end{vmatrix} \quad (257)$$

$$v(i, j) = \begin{vmatrix} 20 & 0 \\ 0 & 10 \end{vmatrix} \quad (258)$$

Here  
Equilibrium (1,1)(2,2)  
Pareto (1,1)(2,2)

# The deal

If  $v(i, j) \neq -u(i, j)$ ,  
then deal is possible.

For example inspector can make deal with poacher to  
divide the pray.

The solution is Nash deal.

Nash deal depends on a set  $D$  of feasible payoff partitions  
and on the "no-deal" payoffs

$$\begin{aligned}u^* &= \max_x \min_y U(x, y), \\v^* &= \max_x \min_y V(x, y).\end{aligned}\tag{259}$$

Here

$u^*$  is maximal guaranteed payoff of the first player  
and  $v^*$  is maximal guarantee payoff of the second player.

The deal is  $(\bar{u}, \bar{v})$ , where  $\bar{u}$  is payoff of the first player and  $\bar{v}$   
is payoff of second player.

# Nash axioms

1.  $(\bar{u}, \bar{v}) \geq (u^*, v^*)$ ,
2.  $(\bar{u}, \bar{v}) \in D$ ,
3. if  $(u, v) \in D$  ir  $(u, v) \geq (\bar{u}, \bar{v})$ ,  
then  $(u, v) = (\bar{u}, \bar{v})$ ,
4. if  $(\bar{u}, \bar{v}) \in T \subset D$  ir  $(\bar{u}, \bar{v}) = \phi(D, u^*, v^*)$ , then  
 $(\bar{u}, \bar{v}) = \phi(T, u^*, v^*)$ ,
5. if a set  $D'$  is obtained from the set  $D$  by these equalities  
 $u' = a_1u + b_1$ ,  $v' = a_2v + b_2$ , then, from  $\phi(D, u^*, v^*) = (\bar{u}, \bar{v})$   
follows that  $\phi(D', a_1u + b_1, a_2v + b_2) = (a_1\bar{u} + b_1, a_2\bar{v} + b_2)$ ,
6. if  $(u, v) \in D \Leftrightarrow (v, u) \in D$ ,  $u^* = v^*$  ir  $\phi(D, u^*, v^*) = (\bar{u}, \bar{v})$ ,  
then  $\bar{u} = \bar{v}$ .

# Nash deal

If Nash axioms are true then exists an unique deal function

$$\phi(D, u^*, v^*) = (\bar{u}, \bar{v})$$

If there is such pair  $(u, v) \in D$  that  $u > u^*$ ,  $v > v^*$ , then the deal is

$$(\bar{u}, \bar{v}) = \arg \max_{u,v} (u - u^*)(v - v^*), \quad (260)$$

where

$$(u, v) \in D, \quad u \geq u^*, \quad v \geq v^* \quad (261)$$

If the feasible payoff is limited by  $c$  then

$$D = \{(u, v) : u + v \leq c\}. \quad (262)$$

Here the deal

$$(\bar{u}, \bar{v}) = \left( \frac{(c + u^* - v^*)}{2}, \frac{(c + v^* - u^*)}{2} \right) \quad (263)$$

# Nash deal, simple example

In simple example of bimatrix game

$$(u^*, v^*) = (0, 1/3).$$

If feasible payoff  $u + v$  is not limited then the deal

$$(\bar{u}, \bar{v}) = (1, 1).$$

If feasible payoff is limited:  $u + v \leq c = 1$ ,

then the deal

$$(\bar{u}, \bar{v}) = (1/3, 2/3).$$

If the payoff limit is  $c = 1/3$

then a part of the first player in the deal is zero

$$(\bar{u}, \bar{v}) = (0, 1/3)$$

Therefore the first player makes no deal because no-deal payoff is high enough

$$u^* = \bar{u}.$$

# Nash deal, inspector's example

In the inspector-poacher deal  
feasible payoff is limited by  $c = 1$ .

Guaranteed payoffs are

$$u^* = 2/9, v^* = 5/9.$$

Then the Nash deal

$$(\bar{u}, \bar{v}) = (1/3, 2/3).$$

This deal is stable since

$$(1/3, 2/3) > (2/9, 5/9)$$

The deal can be prevented if inspector's  
expected deal penalty

$$B > \bar{u} - u^* = 1/9.$$

Here  $B = p_b b$  where  $p_b$  is probability of deal penalty  $b$ .

If the penalty  $b = 1$  is equal to price of pray

then the penalty probability should be  $p_b > 1/9$ .

# Nash deal, strike example

Here  $x \in [0, a]$  is employer's pure strategy that means to pay salary  $x$ , where  $a$  is the employer's income.

Employee's strategies are

$$y = \begin{cases} 0, & \text{strike} \\ 1, & \text{work} \end{cases} \quad (264)$$

Employer's payoff

$$u(x, y) = \begin{cases} a - x, & \text{if } y = 1 \\ -x, & \text{otherwise.} \end{cases} \quad (265)$$

Employee's payoff

$$v(x, y) = \begin{cases} x, & \text{if } x > 0 \\ -b, & \text{if } x = 0. \end{cases} \quad (266)$$

# Strike example, payoffs

Here guaranteed payoffs

$$u^* = 0, v^* = -b,$$

where  $b$  define the employee's "zero-income" loss.

The feasible payoff is limited by  $c = a$ .

If  $b < a$ , then the deal

$$(\bar{u}, \bar{v}) = ((a + b)/2, (a - b)/2). \quad (267)$$

# Equilibrium, formal definition

Notation:

$m$  is number of players,

$y_j, j = 1, \dots, m$  are strategies of players  $j$ ,

$u_i = u_i(y_j, j = 1, \dots, m), i = 1, \dots, m$  is expected payoff of the player  $i$  that depends on the strategies of all players.

$y_j^0, j = 1, \dots, m$  is a "contract" strategy of player  $j$ , and  $U_j^0$  is the expected "contract" profit. Suppose that a player brakes the contract only if expects larger profit

$$U_j^1 > U_j^0, \quad (268)$$

$$U_j^1 = \max_{y_j} u_j(y_1^0, \dots, y_j, \dots, y_m^0), j = 1, \dots, m. \quad (269)$$

Thus a contract  $y^0 = (y_j^0, j = 1, \dots, m)$  is Nash equilibrium if

$$\Delta U = \sum_j (U_j^1 - U_j^0) = 0. \quad (270)$$

# Equilibrium, sufficient conditions

Equilibrium exists if

1. Expected payoffs  $U_i(x_j, j = 1, \dots, m)$  are convex functions of strategies  $x_j$

2. Sets  $X_j$  of feasible strategies  $x_j$  are convex.

In the "Toss-up" game the set of two pure strategies  $x_1 = 0$  and  $x_2 = 1$  is not convex and no equilibrium exists.

A set of mixed strategies  $x \in [0, 1]$  is convex, so the equilibrium exists.

# Contracting operator

Stability of equilibrium depends on operator  $T: y^{n+1} = T(y^n)$  transforming the "contract" vector ,  $y^0 = (y_j^0, j = 1, \dots, m)$  into "no-contract" vector  $y^1$  by maximization of the expected profit

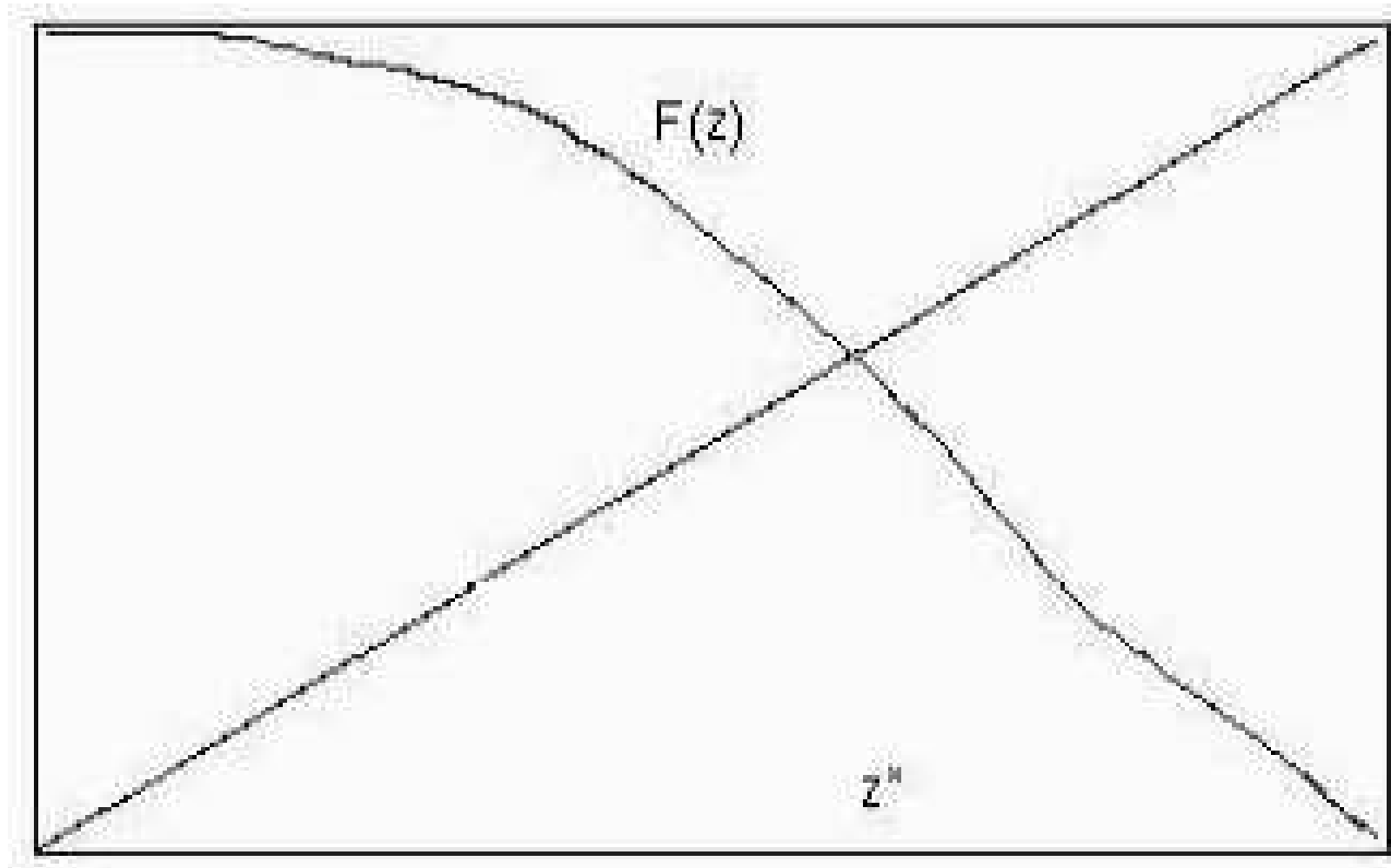
$$y_j^1 = \arg \max_{y_j} u_j(y_1^0, \dots, y_j, \dots, y_m^0), j = 1 \dots, m. \quad (271)$$

under the assumption that competitors honour the contract. Operator  $T$  is contracting if

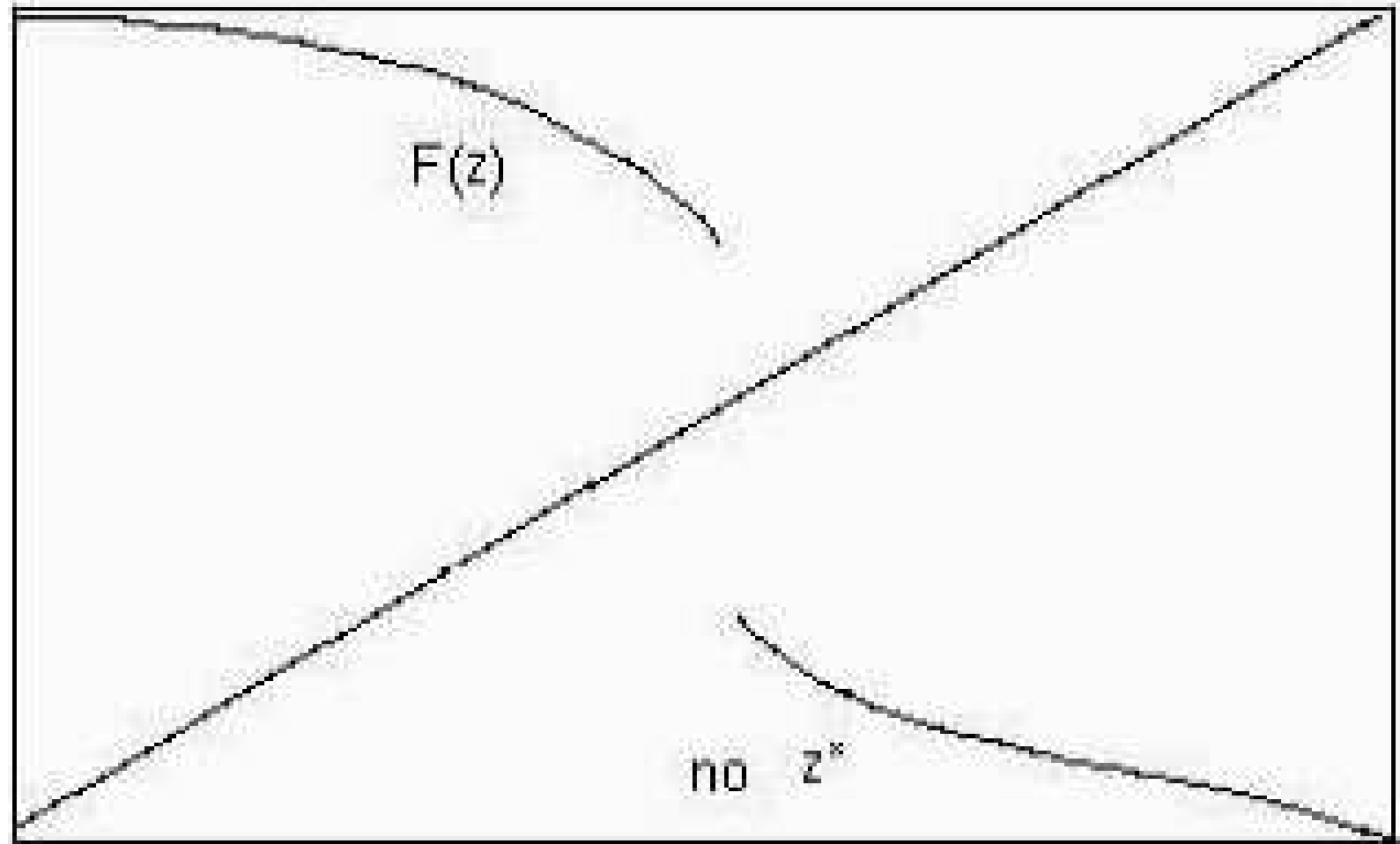
$$\frac{\|T(y^{n+1}) - T(y^n)\|}{\|y^{n+1} - y^n\|} \leq \rho < 1 \quad (272)$$

Equilibrium is stable if  $T$  is contracting.

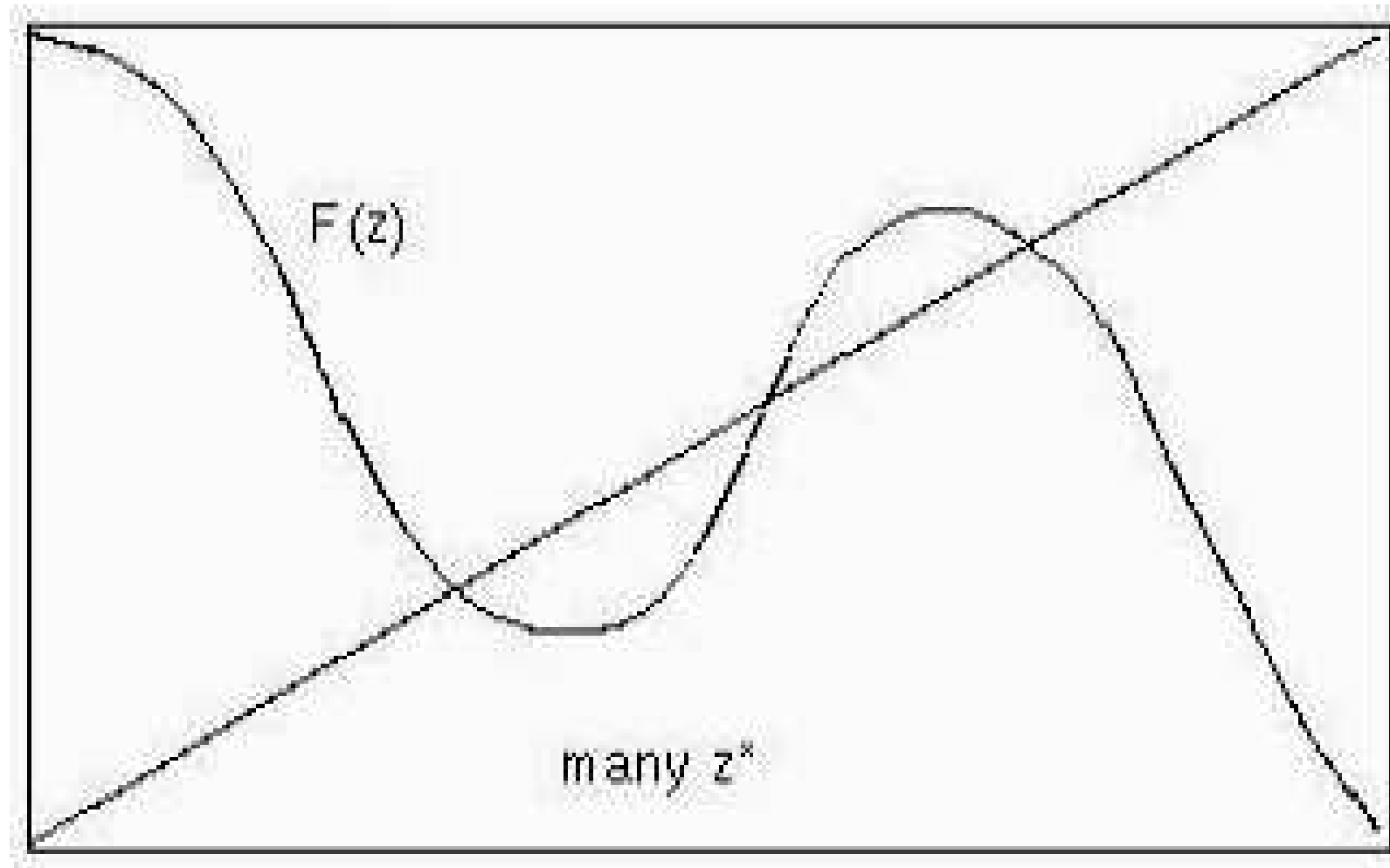
# Fig.5. Fixed point exists



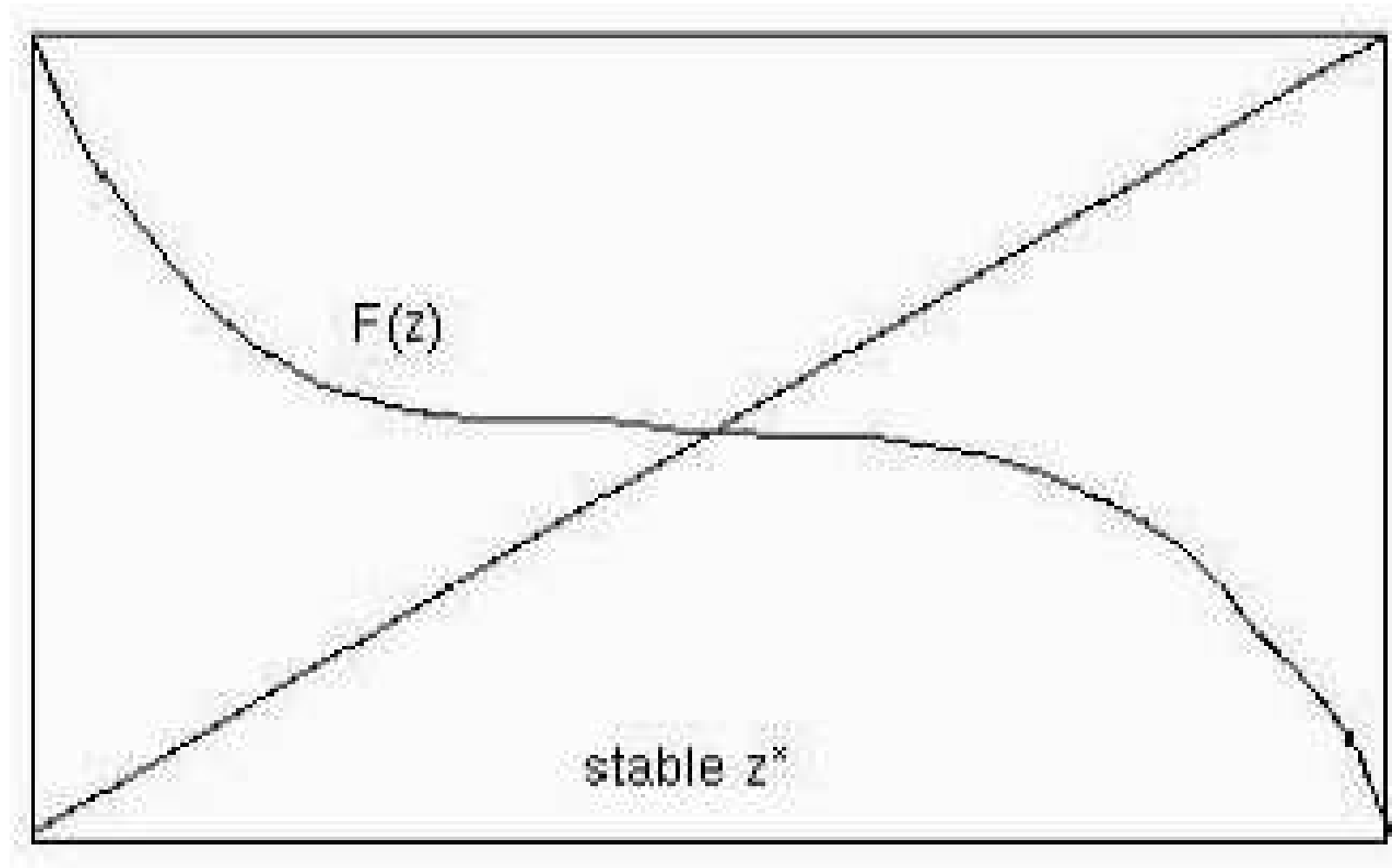
# Fig. 6. No fixed point



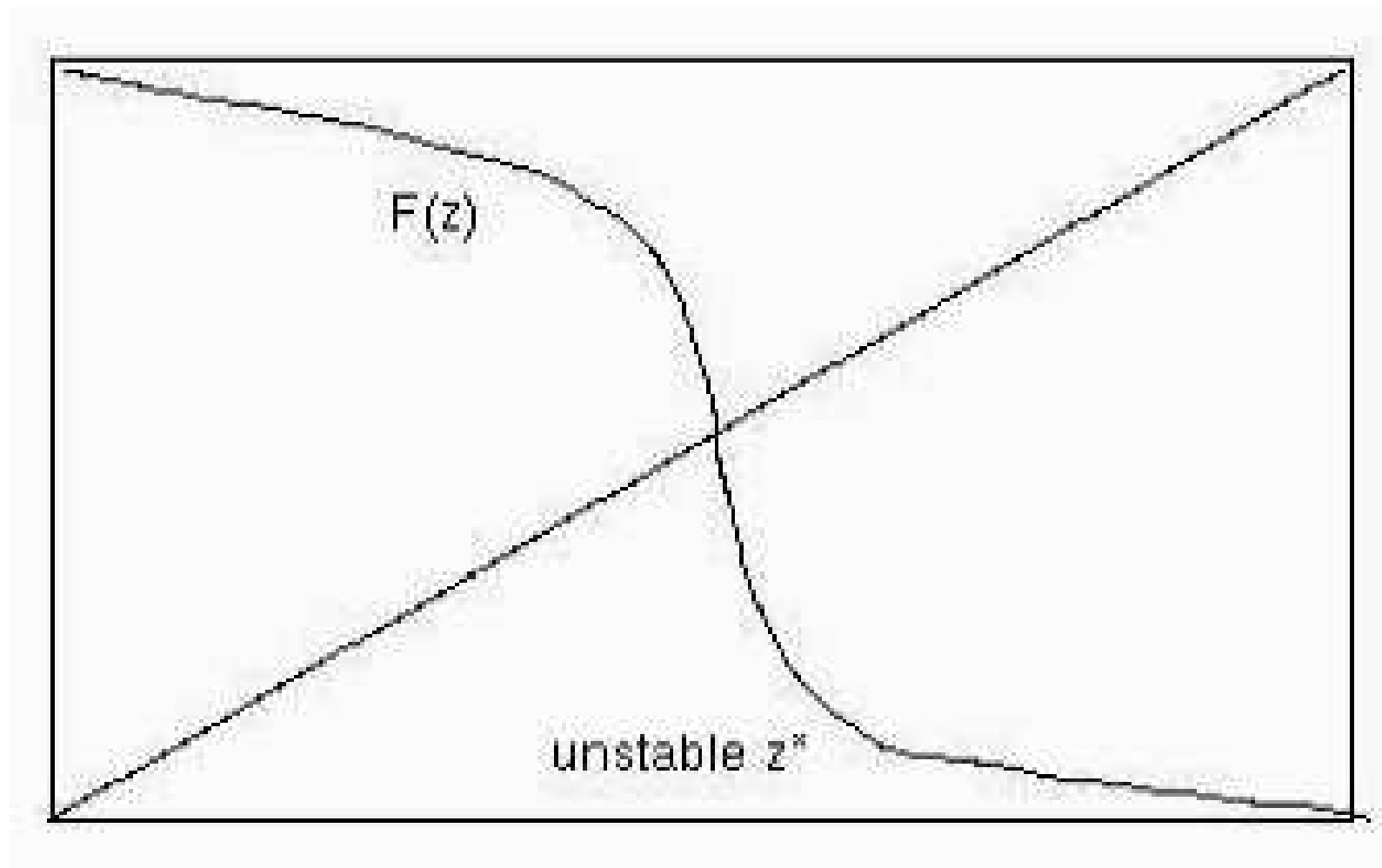
# Fig.7. Several fixed points



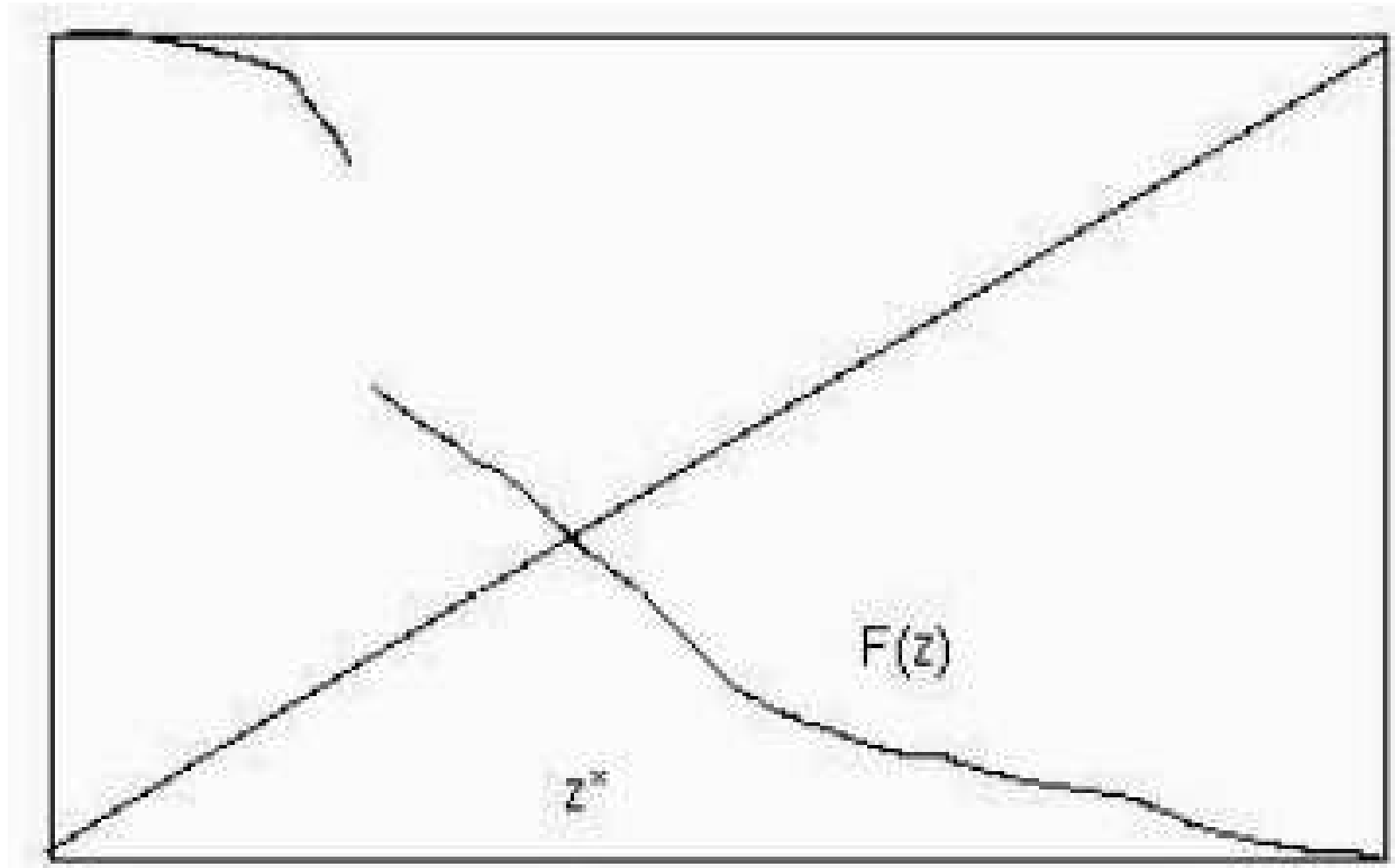
# Fig. 8. Stable fixed point



# Fig. 9. Unstable fixed point



# Fig. 10. Discontinuous example



# Cooperative games

Stability of coalitions is important if the number of players  $m > 2$ .

Stability of a coalition depends on the guaranteed payoff of the coalition and on the partition of this payoff.

A coalition is stable if no changes will provide greater part of guaranteed payoff.

The guaranteed payoff of a coalition is defined by the characteristic function.

# Characteristic function

Notation:

$S$  is a set of all players.

A subset  $s \subset S$  is coalition.

Maximal guaranteed payoff of coalition  $s$

$$v(s) = \max_x \min_y u_s(x, y). \quad (273)$$

Here  $x = x(s)$  a strategy of coalition  $s$ ,

$y = y(S \setminus s)$ , a strategy of coalition of remaining players,

$v(s)$  is characteristic function.

A game  $v$  is fixed sum game if

$$v(s) + v(S \setminus s) = v(S). \quad (274)$$

A game is essential if

$$\sum_{i \in S} v(i) < v(S). \quad (275)$$

# Cooperative game, "Three Boys"

Consider three players  $i = 1, 2, 3$  and three coalitions  $S_j = 1, 2, 3$ , where  $s_1 = \{1, 2\}$ ,  $s_2 = \{1, 3\}$ ,  $s_3 = \{2, 3\}$ . Table shows parts of player payoffs in different coalitions

$$u(i, j) = \begin{vmatrix} 1 & 1 & -2 \\ 1 & -2 & 1 \\ -2 & 1 & 1 \end{vmatrix} \quad (276)$$

Here the characteristic function

$$v(s) = \begin{cases} 2, & \text{if } |s| = 2 \\ -2, & \text{if } |s| = 1. \end{cases} \quad (277)$$

where  $|s|$  is a number of players in coalition  $s$ .

# Game properties, "Three Boys"

This is fixed sum game since

$$v(s) + v(S \setminus s) = v(S) = 0. \quad (278)$$

The game is essential because

$$\sum_{i \in S} v(i) = -6 < v(S) = 0. \quad (279)$$

# Cooperative game, "Joint-Stock"

Consider stock holders  $i = 1, 2, 3, 4$  and their coalitions  $s_j$ .  
Numbers of stocks  $g_1 = 10, g_2 = 20, g_3 = 30, g_4 = 40$ .  
Coalition  $s_j$  is winning if  $G_j > 50$ , where  $G_j = \sum_{i \in s_j} g_i$ . Here  
the characteristic function

$$v(s_j) = \begin{cases} 1, & \text{if } G_j > 50 \\ 0, & \text{if } G_j \leq 50. \end{cases} \quad (280)$$

$v$  is a fixed sum game since

$$v(s) + v(S \setminus s) = v(S) = 0. \quad (281)$$

$v$  is essential game because

$$\sum_{i \in S} v(i) = 0 < v(S) = 1. \quad (282)$$

# Payoff partition

Notation:

$z_i$  is a part of player  $i$  in the payoff partition  $z = (z_1, \dots, z_m)$ , A partition  $z$  is feasible if

$$\sum_{i \in S} z_i = v(S) \quad (283)$$

$$z_i \geq v(i) \quad (284)$$

A partition  $z$  dominate partition  $w$  by coalition  $s$

$z \succ_s w$ ,

if

$$\sum_{i \in s} z_i \leq v(s) \quad (285)$$

$$z_i > w_i, \quad i \in s \quad (286)$$

If there is such a coalition  $s$  then we say  $z \succ w$

# Core of game

The core  $C(v)$  of the game  $v$  is a set of all stable partitions. A partition is stable if no coalition can offer better partition. Core provides stability of coalitions since there are no incentives for changes. In practice it means political and economic stability.

However there is no core in essential fixed sum games:

$C(v) = \emptyset$  if

$$\sum_{i \in S} v(i) < v(S),$$
$$v(s) + v(S \setminus s) = v(S).$$

Such are examples 1 and 2.

In competition models "Nash" and "Walras" the core  $C(v)$  may exist because they both are not fixed sum games

# Shapley vector

If no core exists convenient tool of partition is Shapley vector. Shapley partition can be stable too if all the payers understand and agree with the Shapley conditions:

1.  $\sum_{i \in s} \phi_i[s] = v(s)$ ,  
where  $\phi_i[s]$  is Shapley partition,
2.  $\phi_{\pi(i)}[\pi v] = \phi_i[v]$ , where  $\pi$  is permutation of players,
3.  $\phi_i[u + v] = \phi_i[u] + \phi_i[v]$ , where  $u$  and  $v$  are two games.

If these conditions are true then there exists the unique Shapley partition:

$$\phi_i[v] = \sum_{s \subset S, i \in s} (|s| - 1)! (|S| - |s|)! / |S|! (v(s) - v(s \setminus \{i\})), \quad (287)$$

# Shapley vector, relevant coalitions

For a player  $i$  relevant are only those coalitions  $S_i$  that wins with the player  $i$  and loses without this player. Thus a player  $i$  can pretend for a part of payoff of coalition  $S_i$

# Shapley partition, "Three Boys"

In the "Three boys" example:

$$S_1 = \{\{s_1, s_2\}, \{s_1, s_3\}, S_2 = \{s_2, s_3\}, \{s_1, s_2\}, S_3 = \{s_2, s_3\}, \{s_1, s_3\}\}.$$

Here Shapley partition

$$\begin{aligned} \phi_i[v] &= (2 - 1)!(3 - 2)!/3! + \\ &(2 - 1)!(3 - 2)!/3! = 1/3. \end{aligned} \tag{288}$$

# Shapley partition, "Joint-Stock"

In the "Joint Stock" example:

$$S_1 = \{1, 2, 3\}, S_2 = \{\{1, 2, 3\}, \{2, 4\}, \{1, 2, 4\}\},$$

$$S_3 = \{\{1, 2, 3\}, \{3, 4\}, \{1, 3, 4\}\},$$

$$S_4 = \{\{2, 4\}, \{1, 2, 4\}, \{1, 3, 4\}, \{2, 3, 4\}, \{3, 4\}\}.$$

$$\phi_1[v] = (2 - 1)!(4 - 1)!/4! = 1/12,$$

$$\begin{aligned} \phi_2[v] &= 2(3 - 1)!(4 - 3)!/4! + \\ &\quad (2 - 1)!(4 - 2)!/4! = 3/12, \end{aligned}$$

$$\begin{aligned} \phi_3[v] &= 2(3 - 1)!(4 - 3)!/4! + \\ &\quad (2 - 1)!(4 - 2)!/4! = 3/12, \end{aligned}$$

$$\begin{aligned} \phi_4[v] &= 3(3 - 1)!(4 - 3)!/4! + \\ &\quad 2(2 - 1)!(4 - 2)!/4! = 5/12. \end{aligned}$$

# Stabilization of partitions

Payoff partitions belonging to the core of game  $C(v)$  are stable without conditions.

Shapley partitions are stable if all the players understand and agree with Shapley conditions.

That means that players predict correctly the reaction of their partners and corresponding consequences. That is not always a practical assumption.

Practically partitions can be stabilized by penalties for "bad" behavior and bonuses for "good" behavior.

That transforms fixed sum game into non-fixed sum game with stable core  $C(v)$ .

# Stabilization of game, "Three Boys"

Introduce in the "Tree Boys" example a "Unity Bonus"-  
additional payoff +1 for each "boy" if all three unites to form  
a coalition  $s_4 = \{1, 2, 3\}$

$$u(i, j) = \begin{vmatrix} 1 & 1 & -2 & 1 \\ 1 & -2 & 1 & 1 \\ -2 & 1 & 1 & 1 \end{vmatrix}. \quad (289)$$

Then the characteristic function

$$v(s) = \begin{cases} 2, & \text{if } |s|=2 \\ -2, & \text{if } |s|=1 \\ 3, & \text{if } |s|=3 \end{cases} \quad (290)$$

Core  $C(v) = (1, 1, 1)$  implements the coalition  $s_4 = \{1, 2, 3\}$ .

$$v(s) + v(S \setminus s) \neq v(S) = 3, \quad |s| < 3. \quad (291)$$

# Stabilization of game, "Joint-Stock"

If we introduce in the "Joint Stock" game a penalty for coalitions for breaking the rules of proportional partitions then the characteristic function

$$v(s_j) = \begin{cases} 1, & \text{if } G_j > 50 \text{ and } z_i \in Z, i \in s_j \\ 0, & \text{if } G_j > 50 \text{ and } z_i \notin Z, i \in s_j \\ -1, & \text{if } G_j \leq 50 \text{ and } z_i \notin Z, i \in s_j. \end{cases}$$

Here  $Z$  is a partition of payoff in proportion to stock number.  
Here exists core of game  $C(v) = (10, 20, 30, 40)$   
implementing the coalition  $s = \{1, 2, 3, 4\}$ .

# AR-ABS models

This is a version of AR (Auto-Regression) model

$$w_t = \sum_{i=1}^p a_i w_{t-i} + \epsilon_t, \quad (292)$$

where

$w_t$  prediction for tomorrow

$w_{t-1}$  observed value today,

$\epsilon_t$  random unpredictable variable tomorrow,

$a_i$  coefficients of "importance"

defined by minimization of residual

$$f(x) = \sum_{t=1}^T |\epsilon_t|. \quad (293)$$

# Solving AR-ABS model

We minimize the residual by linear programming

$$\min_{a,u} \sum_{t=1}^T u_t \quad (294)$$

$$u_t \geq \epsilon_t, \quad t = 1, \dots, T, \quad (295)$$

$$u_t \geq -\epsilon_t \quad t = 1, \dots, T, \quad (296)$$

$$a_i = a_i^1 - a_i^2, \quad a_i^k \geq 0, \quad k = 1, 2, \quad i = 1, \dots, p. \quad (297)$$

# Last slide

This is the last slide. Do you want to go to the **second one**?