UNIVERSITY OF ZIELONA GÓRA
Faculty of Management
and Faculty of Mechanical Engineering

In association with the Design Society

FOURTH INTERNATIONAL SEMINAR AND WORKSHOP
7th - 9th October 2004
Zielona Góra - Rydzyna
POLAND

Engineering Design in Integrated Product Development
Management of Design Complexity

SOFT APPROACH TO COMPUTER AIDED OPTIMIZATION

W. TARNOWSKI

Control Systems Department, Koszalin Technical University
e–mail: tarnowsk@lew.tu.koszalin.pl

Keywords: *soft optimization, poly–optimization, optimality, constraints*

Abstract: *If decision is subjected to many constraints, it may be disputable how to formalise an optimality function, and even multi–attribute approach is not necessarily the solution, because decision maker may not intuitively grasp the play of criteria. What more, if the mathematical model contains non–linear partial differential equations, determining the Pareto solutions set may be a time – consuming process. Decision maker may be discouraged to proceed. Then a dialog procedure of finding satisfactory solution may be recommended. In the paper, a kind of such interactive method called the "soft optimization" is proposed and presented on examples.*

MOTIVATION

People always have striven to improve their decisions. The best way is a formal mathematical optimization, which yields definitely the accurate absolute best solution of a decision problem. But it has numerous disadvantages, among them are two:

1. The necessity to formalise one unique scalar optimality function; the trouble is that it must express a quality which always is a vector notion; hence very often there is no consensus how to build this function;
2. The other trouble of the classical optimization is the rather complicated and time consuming computation, unless the problem is very simple, with few constraints and few decision variables, what is not a case in the real practice.

Probably these are the reasons that the formal optimization is not common in life situations.

Methodological answer for the first drawback of the optimization is the poly–optimization [1], [3], but the obtained results of this method are still difficult for a human decision maker if there are more than two or three criteria, because a graphical presentation may be difficult to grasp by a human operator [2].

However, a more flexible approach exists, broadly known as a *trial–and–error* procedure, and the objective of this paper is to propose a full recognition and an 'official' approval of such method as a completely rational and effective technique, especially if extensively aided by computer.

THE GOAL

The goal of this paper is to formalise a quasi–optimal procedure of finding a satisfying decision, in a man–computer dialog procedure, which is in fact a lexicographic semi–optimization.

DEFINITIONS

A **decision x** is defined as a set (or a vector) of variables x_1, x_2,…,x_m which are called *decision variables*:

$$\mathbf{x} = [x_1, x_2, ..., x_m] \quad (1)$$

For each decision **x** one may calculate various *performance characteristics* $\mathbf{y} = [y_1, y_2, ...]$, and let us assume, that an adequate mathematical expressions are reachable:

$$y_j = f_j(x_1, x_2, ... x_m, p_1, p_2, ..., p_r) \quad j = 1, ... s \quad (2)$$

where $\mathbf{p} = [p_1, p_2, ..., p_r]$ is a vector of constant parameters.

In general case the decision **x** may be accepted as an allowable one provided it meets some set of various but well defined *requirements*. In the specific case all requirements may be divided into two complementary subsets: *objectives* (*criteria*) and *constraints*.

Objectives *k*

Objectives are preferences imposed on specific perfor-mance characteristics $\mathbf{k} \subset \mathbf{y}$:

$$k_j = f_j(x_1, x_2, ... x_m, p_1, p_2, ..., p_r) \rightarrow max! \quad (3)$$

or:

$$k_j = f_j(x_1, x_2, ... x_m, p_1, p_2, ..., p_r) \rightarrow min! \quad (4)$$

The symbol "$\rightarrow$ max!"means: "the bigger value the better quality", for example the effectiveness, reliability or safety, and the symbol "$\rightarrow$ min! " means: "the smaller the better", for example the energy consumption, cost or failure ratio.
Objectives used to be also named *criteria*, *quality indexes* or *quality criteria*.

Constraints

Constraints are one– or two–sides limitations imposed on performance characteristics (2):

$$y_j = f_j(x_1, x_2, ... x_m, p_1, p_2, ..., p_r) > a_j \quad j = 1, ... s \quad (5a)$$

or/and:

$$y_j = f_j(x_1, x_2, ... x_m, p_1, p_2, ..., p_r) < b_j \quad j = 1, ... s \quad (5b)$$

The constraints concern the threshold–type requirements, like stability, yield stress criterion etc. They are derived from the commision documents, standards or from other regulations.
Also typical are constraints imposed directly on values of the decision variables x_i, these are:
– lower limits $x_{i\ low}$ and upper limits $x_{i\ upper}$,

$$x_i \in \lfloor x_{i.low}, x_{i.upper} \rfloor \quad (6)$$

– or discrete values:

$$x_i \in \lfloor x_i^1, x_i^2, x_i^3, ... \rfloor \quad (7)$$

resulted from standarisation requirements or of other reasons.
Constraints (5a) may be re–written into the standard shape:

$$a_j - f_j(x, p) \leq 0 \quad (8a)$$

and respectively constraints (5b) into:

$$f_j(x, p) - b_j \leq 0 \quad (8b)$$

ASSUMPTIONS

It is reasonable to apply the proposed procedure if an overall quality function is not defined or not consent, or if an optimization computation time is not acceptable.
Conditions of success of the proposed procedure are:

- Decision–maker is experienced;
- Adequate optimization algorithm is implemented on computer;
- Each variant of decision may be presented (on computer) in a graphical shape, easy for the quality assessment by the decision–maker.

Figure 1 demonstrates, that an experienced operator may easy compare and assess various solutions, if given graphically, side by side.

It should be noted, that all sequential decisions are based on the human experience and intuition.

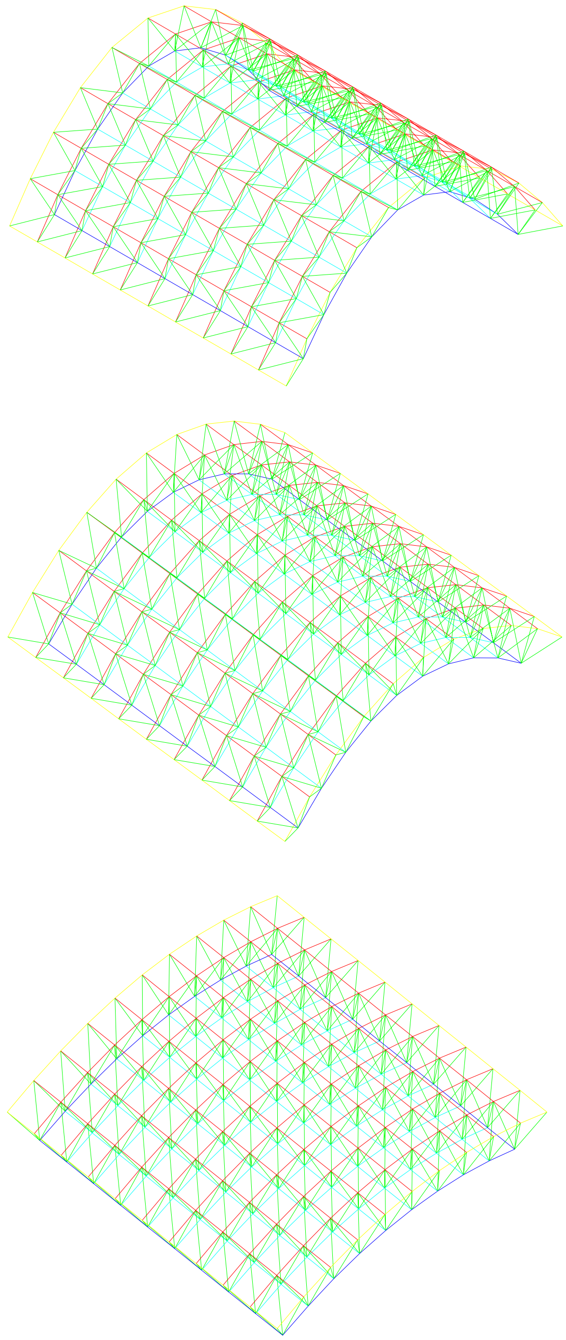

Fig. 1. *Variants of design of a truss roof construction*

GENERAL IDEA AND OUTLINE OF THE METHOD

In the standard optimalisation problem the triple is given: <**x**, Φ(**x**), Ω(**x**)>, where **x** is the decision variables vector, Φ(**x**) is an overall optimality criterion and **Ω**(**x**) is a set of constraints, defined in a fully mathematical form. To the contrary, in the proposed procedure only the pair is necessary: <**x**, Ω(**x**)>, however the experience and intuition of the human operator is necessary.

He/she indicates the most important objective k_1, and provisionally states its reasonable acceptable value, minimal c_1 or maximal d_1, as the lower and the upper limits, what adequate. Then applying any computer optimization programme, and setting the optimality function equals zero (or other arbitrary constant value), computer finds a solution **x**. This variant is allowable, as all constraints are in power.
The found solution is presented to a decision maker, by computer, in the most adequate form for assessment, typically graphically.

Then the decision maker may interactively change c_1 or d_1, what applicable, repeats the optimization computation and observe results, and so on, until a satisfactory result is obtained. Then the decision maker may skip to another active objective k_2, and proceeds the above actions, until the variant satisfying on all criteria is found.
For two decision variables, x_1 and x_2, and four constraints og_1 to og_4, an exemplary situation is depicted on Figure 2.

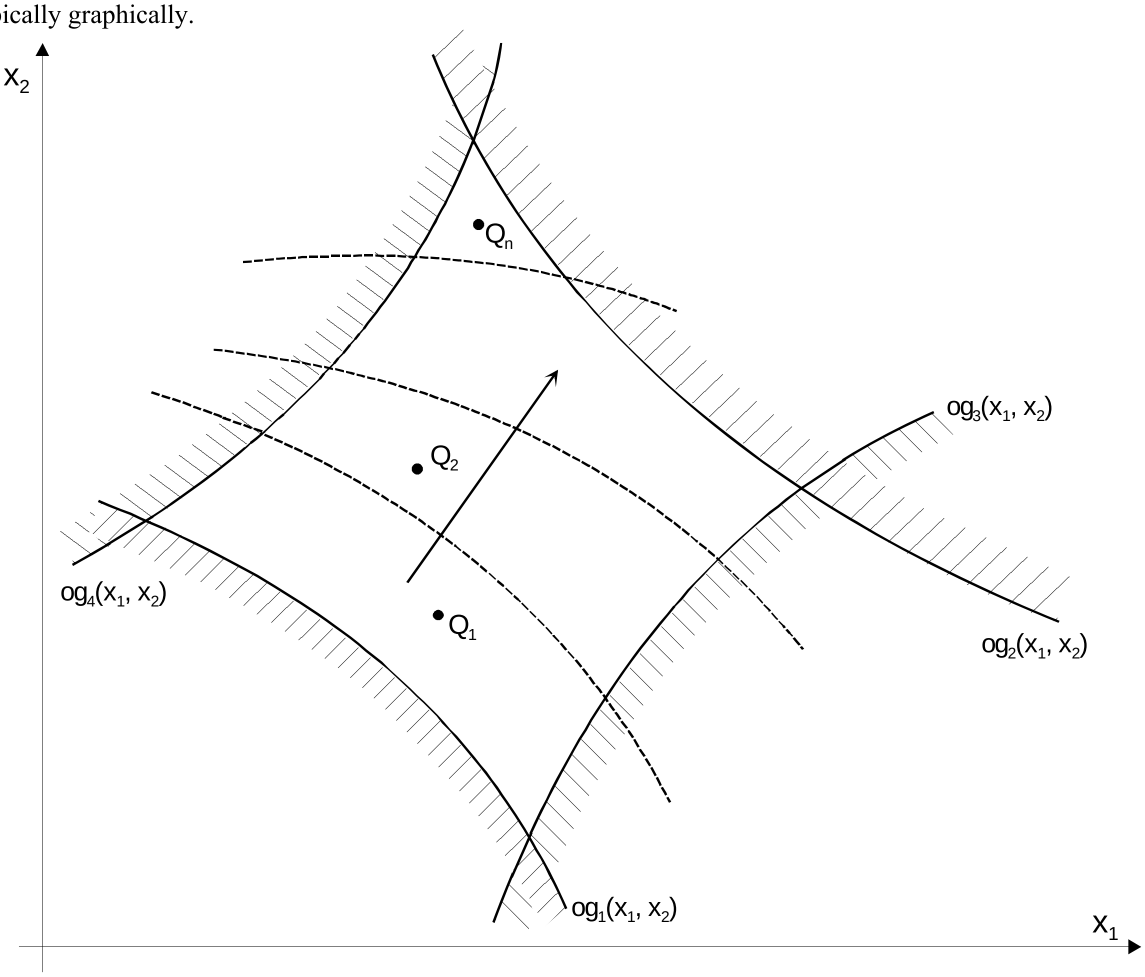

Fig. 2. *Exemplary set of allowable decisions defined by four constraints og_1 ... og_4; the arrow → shows the direction of quality improving when the constraint og_1 is set to be more restrictive; Q_1, Q_2, ... – sequential quasi–optimal solutions*

PROPOSED PROCEDURE

The following stages may be proposed.

1) State the scope of the decision problem; define decision variables **x** and constant parameters **p**. Identify performance parameters **y**.
2) Define mathematically all requirements, preferences and limitations in term of **x** and **p** (Eqns 8a and 8b)
3) From the set **y** derive objectives (criteria) **k**; the remaining are constraints **og**; (**k + og = y**)
4) Rank objectives **k** accordingly to their impact on the overall quality of variants: k_1, k_2, …. Here k_1 is the most important.
5) Choose computer procedure for optimization (for example *constr* function in MATLAB). Implement constraints and objectives, as constraints. Set the optimization function equals zero.
6) Define the way of variants presentation and design a pro-gramme on computer (typically it is a visualisation).

Below there is a flow diagram of the proposed algorithm.

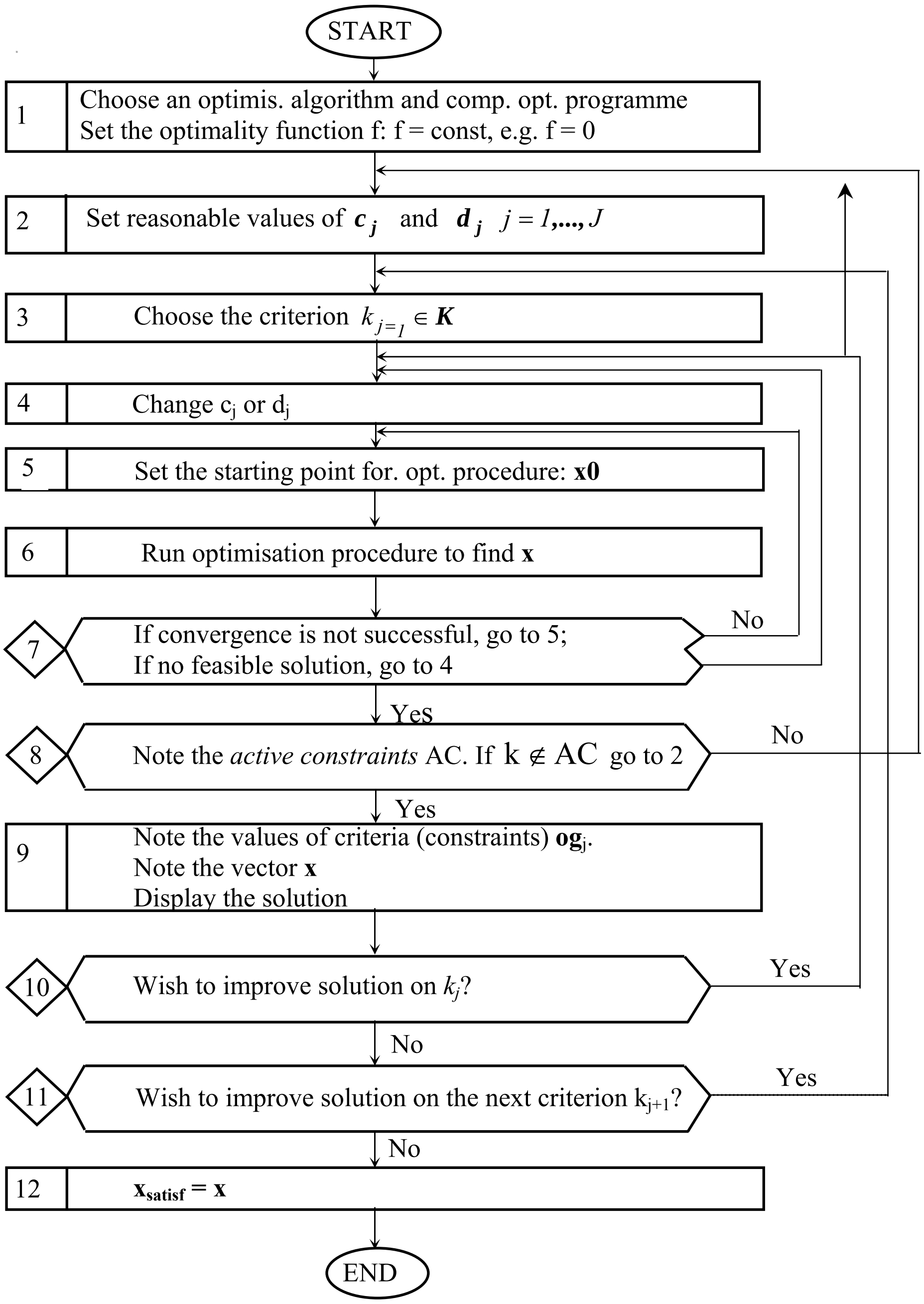

Fig. 3. *Flow diagram of the 'soft' optimization algorithm for defining* $\mathbf{x}_{satisf}$

EXAMPLES

Example 1 [5]: The object is a servomechanism with two feedback loops, an outer one from the position and the other from the velocity signal, with two controllers. The design optimization problem is to determine 7 decision variables (3 parameters of each PID–type controller and the gain coefficient of a tachometric transducer). Constraints refer to characteristics of the step response: an overshot, the response time and the gain of a closed loop. There are few possible definitions of an optimality criterion: each of performance parameters could be adopted; these are the response time, a few integrals of error, the power consumption and others.

Following the proposed soft optimization algorithm, as the first constraint the response time was selected, with the exemplary limit d_1 = 60 s. In Fig. 4 exemplary step answers are depicted: for the optimal solution (a) and for a satisfactory solution (b) (soft optimization). The second characteristic may seem a bit worse (longer transient process and a small overshoot), but it requires smaller energy (see the smaller velocity pick) and was obtained in much shorter computation time.

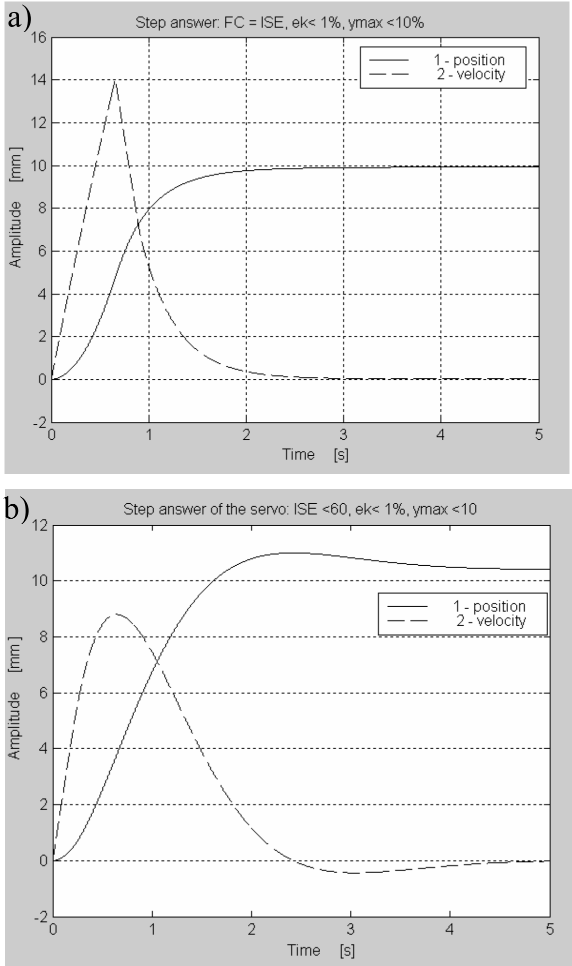

Fig. 4. *Step answers of the servomechanism: a) the optimal on the ISE optimality function (computation time 4625s), b) 'soft' optimal (computation time 797s)*

Example 2: [4] Now there is a continuous object, which is a flat plate, heated by an external source on one edge, cooled on the others;

There may be various goals of synthesis, for example:

- to find a control function (this is a dynamic optimization problem);
- to find parameters of a control function (a static problem);
- to design the object.

Hence, adequate decision variables or input functions are to be found, for example power time characteristic of the heat source, or its position and shape.

What more, various definitions of an optimization function are possible, for example the maximal difference of temperatures in the space of the plate (in a given span of time), or the time when a defined temperature is reached, or a time of the transient process – and others.

Also, many constraints may be imposed.

In [4] results of experiments are presented. Exemplary result is shown on Fig. 5.

As a conclusion, for 2D object the computation time was even 14 times shorter for soft optimization then for the traditional, and a quality of a solution was only a little worse.

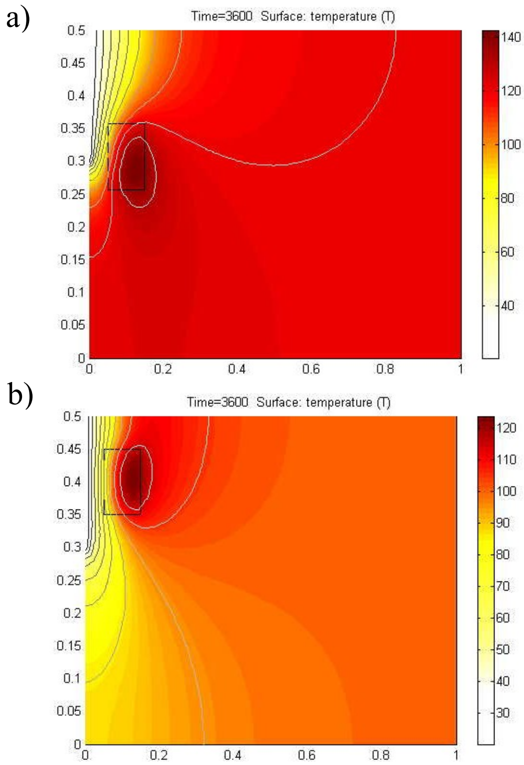

Fig. 5. *Exemplary results of he standard optimisation (a), optimality function is 121,5 deg and computations time is 489 s, and the soft optimization (b): 105 deg and 42 s, respectively*

The above given examples illustrate that there may be formulated various optimization problems, and various procedures may be applied to solve them.

Possible approaches

There are several possible approaches to the 'soft' optimization. Typical ones are:

1) Searching for any acceptable solutions according only to the given constraints. Such approach may be justified by the fact, that typically the optimal solution is located on the edge of the acceptable solutions space ("on a constraint(s)"), although a question remains which one(s). Intuition and experience may suggest the limit: for example, in mechanical objects it is the stress/strain restriction. In MATLAB the function *constr* or *fmincons* is a comfortable technique, which finds the solution on constraint, near to the starting point, if a constant value of optimality function is declared. The human operator provides only the starting point.
2) Trial–and–error procedure: the session is carried on by an operator and may be stopped at any point. The operator submits each probable solution (values of all decision variables) for verification.
3) Standard analysis: the operator proposes a specific solution (values of all decision variables), and his/her computer calculates all programmed characteristics and/or a schematic view of the

object. Example: a bridge or any other construction object – an architect is the operator, and the computer completes computations of distribution of the stress/strain analysis; an automatic control system – the designer defines the structure and its parameters and the computer simulates its operation and displays characteristics.

In all cases an adequate mathematical model of the designed object is necessary. The resulted information about the decision (the selected variant i.e. the solution) should be displayed graphically on the monitor, to make an assessment easier for the human operator.

CONCLUSIONS

A quasi–optimization (in the absence of the optimality function) may be completed in a dialog procedure. The presented idea is based on the dialog mode of decision–making, which is a constitutive and inherent feature of CAD systems. The proposed approach intends to encourage the interactive mode of the human–computer co–operation and make it more efficient. It may be especially efficient for objects with complex models: typically those containing non–linear partial differential equations, with many decision variables and numerous constraints.

The "soft" optimization creates a possibility for resolving non–unique inverse problems [1].

The interactive (dialog) procedure for finding a satisfying solution may be recommended when:

- There are many challenging requirements and a decision maker cannot or denies to define one global optimality function which may represent a compromise;
- The mathematical model of the designed object is rather complex and difficult for computations, for example the model comprises non–linear differential (partial) equations; or there are many non-linear constraints and many decision variables. In such case the decision–maker may rather postpone formalising and resolving a poly–optimization problem.

The proposed soft–optimization procedure reduces the computation time and does not require defining the optimality function. However, it needs a heuristic co–operation with a human operator; but this is not a particular drawback of this method, as it is also the feature of any other type of optimization.

References

[1] Tarnowski W.: *Poly–optimization as a paradigm of an inverse problem*, (in:) Polioptymalizacja i CAD – Mielno 2002. WNT, Warszawa 2002.

[2] Rudzki K., Tarnowski W.: *Multi–attribute optimization by MATLAB software on the tooth gear example,* (in Polish). (in:) Proceed. of Techn. Univ. Koszalin, Mech. Engng. Dept., No 21, Koszalin 1997.

[3] Tarnowski W.: *Poly–Optimization as a Bridge Between Heuristics and a Formal Optimization In Design,* Transactions of the 10th Intern. Conf. on Design Tools and Methods in Ind. Engng., Florence, Italy, Sept. 1997.

[4] Zaporski P., Tarnowski W.: *Soft optimization of objects with the heat flow in 2D and 3D continuous space,* (in:) Polioptymalizacja i CAD. WNT, Warszawa 2004.

[5] Ociepa Z.: *Soft optimization of servomechanisms,* (in Polish). (in:) Polioptymalizacja i CAD. WNT, Warszawa 2004.