БЪЛГАРСКА АКАДЕМИЯ НА НАУКИТЕ . BULGARIAN ACADEMY OF SCIENCES

ПРОБЛЕМИ НА ТЕХНИЧЕСКАТА КИБЕРНЕТИКА И РОБОТИКАТА, 52
PROBLEMS OF ENGINEERING CYBERNETICS AND ROBOTICS, 52

София . 2001 . Sofia

Optimization System for Solving Problems of Nonlinear Multiobjective Programming

Milena Sarneva

Institute of Information Technologies, 1113 Sofia

1. Introduction

Every day almost everyone has to plan or organize some activity by using different resources – time, money, etc. It is even more important for managers that often are in a continuous process of decision making. Most managers rely mainly on their experience and intuition, rather than on strict mathematical models and optimization methods.

With advances in the computer industry, it is becoming far easier for these processes to be aided by software optimization systems. Development of different types of computer programs eases and helps the decision making process.

A good example of such systems is MONP-16, designed in the Institute of Information Technologies, Bulgarian Academy of Sciences. It solves the basic class of multiobjective nonlinear programming problems. It runs on personal computers IBM PC/XT/AT and their compatibles and MS-DOS operating system.

This system is designed basically for experts in mathematical programming as it requires a precise analysis and description of the problem as a multiobjective nonlinear programming one by the user. MOLP-16 uses the Satisfactory Trade-off method of Nakayama and single objective optimization Lagrange method (using Lagrange multipliers) as wells as SQP (sequential quadratic programming) method [2].

Computers are becoming faster and faster and operation systems – easier and more user friendly. The appearance and dramatic growth of the global Internet with its graphical user interface lead up to the development and usage of modern technologies and new types of optimization systems for decision making.

The Finland WWW-NIMBUS [3, 4], designed in the University of Jyvaskyla is an example of such a system. The NIMBUS algorithm is capable of solving nonlinear and real-world applications involving nondifferentiable and nonconvex functions. It was first implemented in Fortran for MS-DOS operating system and it was mainly

used in their laboratories as it was not user friendly enough. Later on it was transformed into C language for MS-Windows. The interface was much more intuitive and functional, but at the cost of reduced computational capacity, the common problem with such realizations. The widely spread Internet with its graphical user interface WWW (World Wide Web) has led to realization of the third version of the algorithm – WWW-NIMBUS, using the modern Java language [5]. This approach provides many advantages:

- any software in the Internet is available to large amounts of people;
- no special requirements are set on the user's computer, its operating system, etc.;
- all the software needed is a WWW browser;
- centralized computing part to one server;
- flexible, independent and convenient graphical user interface;
- no updates of the software have to be distributed, the user always has access to the latest available version.

So this system is a multiobjective optimization tool easily available for any Internet user.

This paper describes a different method of approach in optimization decision making systems, the so called *shell*-systems. The basic objective is the realization of a user friendly interface and a quick single objective optimization method. The nonlinear multiobjective algorithm can be easily changed as well as new methods added. Thus the optimization system is very flexible both for the user who can choose between different methods and for the developer of new algorithms who can easily add, test and update the new methods. Special attention is paid on the design and realization of an intuitive, convenient user friendly interface as it is well understood how important it is in the process of interaction between the Decision Maker (user) and the computer system

The continuation of this paper is organized as follows. Chapter 2 describes the basic concepts and notations. The characteristics of the *shell*-system and its realization are introduced in Chapter 3. Brief description of the technical realization can be found in Chapter 4. The paper is concluded in Chapter 6. And some References are listed in Chapter 6.

2. Concepts and notations

Most of the real world optimization problems are difficult to define properly with appropriate mathematical models. Basically this is due to the following characteristics of this kind of problems [7]:

- Many conflicting goals.
- Lack of specificity what the decision variables really are.
- Uncertainty of what the bound values and constraints on the decision variables and the corresponding functions are.
- Lack of clear cause-effect relationship.
- Many stochastic elements.
- The dynamic character of the problems that causes the goals, constraints and other relationship to vary over time.
- Lack of enough available information needful to specify the problem.
- Some of the data is described qualitative.
- Strong possibility for unexpected consequences resulting in alternation of the existing conditions.

All these elements should be considered when the mathematical optimization problem is being built.

We study a multiobjective optimization problem of the following form [1, 7]:

$$\min\{f_1(x), \ldots, f_k(x)\}, \; x \in S,$$

subject to

$$g_i(x) \geq 0, \; i = \overline{1, m},$$

$$h_j(x) = 0, \; j = \overline{1, p}$$

where we have:

a) functions f_k, g_i, h_j, at least one of the which is nonlinear.

b) k, $k \geq 2$, conflicting objective functions of the form

$$f_i : R^k \to R.$$

Often we denote the criteria vector as follows:

$$f(x) = (f_1(x), \ldots, f_k(x))^T;$$

c) $x = (x_1, \ldots, x_n)^T$ is the notation for the *decision variable vector.*

d) S is a feasible nonempty set of variables and

$$S \subset R^n .$$

Let us denote the image of the feasible set S by $Z = f(S)$, $Z \subset R^k$. The elements $z \in Z$ are called *criterion vectors*.

The components z_i^0 of the *ideal vector* z^0 are derived by individual optimization of the objective function:

$$\min f_i(x),$$

subject to $x \in S$ for $i = \overline{1, k}$.

If there were no conflict between the objective functions, then the ideal criterion vector would be the optimal solution of the problem. But unfortunately the ideal criterion vector usually is unfeasible. In other words it is not possible to optimize all the objective functions simultaneously. That is why the Pareto and weak Pareto optimal sets of solutions are defined.

Definition. A decision variable vector $x^* \in S$ and the corresponding criterion vector $z^* \in Z$ are *Pareto optimal* if there does not exist another decision variable vector $x \in S$, such that

$$f_i(x) \leq f_i(x^*), \text{ for all } i = \overline{1, k},$$

and

$$f_j(x) < f_j(x^*)$$

for at least one objective function f_j.

Definition. A decision variable vector $x^* \in S$ and the corresponding criterion vector $z^* \in Z$ are *weak Pareto optimal* if there does not exist another decision variable vector $x \in S$, such that

$$f_i(x) < f_i(x^*), \text{ for all } i = \overline{1, k}$$

The solutions in the Pareto and weak Pareto optimal sets cannot be ordered mathematically as they all are equally good as optimal solutions of the multiobjective problem. That is why to choose only one solution we need the additional preference

information of a Decision Maker (DM). DM is usually the user of the optimization system. The program generates Pareto optimal solutions and DM analyses and evaluates them with a view to its aspirations and preferences.

So as a final solution of the multiobjective optimization problem we consider a feasible solution that is:

- Pareto optimal;
- the most preferable one according to DM.

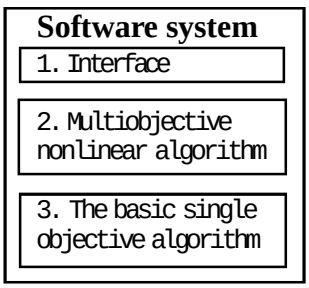
Often it is worthwhile to know the ranges of the objective functions in the Pareto optimal set. The ideal criterion vector represents the lower bounds of the Pareto optimal set. The upper bounds is represented by a criterion vector called *nadir criterion vector*, which is difficult to be calculated accurately. As a good approximation can be used the *payoff table*, which is formed in the process of the finding the ideal criterion vector [1].

The nadir criterion vector could be feasible as well as infeasible.

3. Optimization system description

3.1. Characteristics of the shell-system

The basic modules that the decision making system usually includes are the following:



The basic idea in designing this system is to assure maximal flexibility in choosing the multi-criteria non-linear optimization algorithm (module 2). That is why the main purpose in designing of the system is to realize a graphical "user friendly" interface (module 1) and to implement a basic and fast enough, single objective optimization algorithm. (module 3). These are the main characteristics of the *shell*-systems as it is a "shell" of a software decision making system with no fixed non-linear multi-objective optimization algorithms. It is structured in a way that the implementation does not depend on the module 2. This module is described separately and the system uses it. However, different module 2 algorithms can change some characteristics of the system such as speed.

All these characteristics make the described optimization system flexible both for users and developers. Experts in multiobjective decision making can easily add and test some new algorithms and users can choose between them all.

Let us describe the modules of this *shell*-system.

3.2. Interface

In order to guarantee the date security of the problems, the user has to specify an unique user name and password to be recognised by the system. Depending on the password the user can work only with the its own problems. Eventually in the future

versions there will be groups of users – expert, novice, etc. so the system can run differently depending on the individual. It can use some default algorithm for finding the optimal solution if the user is a *novice* and to let the Decision Maker control every step of the process.

When the system is loaded the user has to enter his/her user name and password. If he/she is a *new user*, i.e. uses the system for the first time, a profile should be created first. It includes a user name and password and some optional data for the user.

User can perform the following:

- create a new problem;
- open an existing problem;
- remove a problem;
- change a password.

Let us describe briefly these main activities.

Create a new problem

Every problem has *name* and *description*, so the user that creates it can recognize it later on to continue the work on it or to use the obtained results.

To create a new problem, the user has to enter some information about the problem – the number of the necessary variables (n), linear (m) and nonlinear (p) constraints. Depending on these data the respective number of text fields for entering the information are generated. The following names are generated as well:

- variables $x_1, \ldots, x_n$;
- linear constraints $N_1, \ldots, N_k$;
- linear constraints $g_1, \ldots, g_m$.

These names are useful to identify the data later on.

The objective function and the constraints are entered in the text fields using determinate form that the system recognizes. The user has to describe the date in the following way (*italic* is used to notify the fields that have to be filled by the user):

- enter the values and borders for variables:

$$x_i : \text{lower bound} \le \text{start value} \le \text{upper bound};$$

- linear constraints:

$$Ax \; relationship \; b$$

i.e. they are described as

$$N_1: \text{coef}_1 x_1 + \text{coef}_2 x_2 + \ldots + \text{coef}_m x_m \;\; relationship \; b,$$

so the user has to enter values for $\text{coef}_1 x_1$, $\text{coef}_2, \ldots, \text{coef}_m$, *relationship* and b. Possible values for the field *relationship* are listed below.

- nonlinear constraints

$$g_i : constrained \; relationship \; 0,$$

where *constraint* is entered in the text field with determinate format.

In the text description of the objective function and constraints the following standard operations, algebraic and trigonometric functions can be used:

- Operations:

+ Addition (as unary operation as well);
– Subtraction (as unary operation as well);

/ Division;
* Multiplication;
^ Raising to power.

• Algebraic functions:

ABS(X) computes the absolute value of X;
EXP(X) computes the exp of X;
SQRT(X) computes the square root of X;
LOG(X) computes natural logarithm of X;
LOG10(X) computes common logarithm of X;
MIN(X_1, X_2, ..., X_n) restores the argument of the smallest value;
MIN(X_1, X_2, ..., X_n) restores the argument of the greatest value.

• Trigonometric functions:

COS(X) computes cosine function of X;
SIN(X) computes sine function of X;
TAN(X) computes tangens function of X;
COSH(X) computes hyperbolic cosine function of X;
SINH(X) computes hyperbolic sine function of X;
TANH(X) computes hyperbolic tangens function of X;
ACOS(X) computes arccos function of X;
ASIN(X) computes arcsin function of X;
ATAN(X) computes arctangens function of X.

The following conditional operator can be used as well:

IF (arg_1 OP arg_2) THEN arg_3 ELSE arg_4,

where OP is one of the following relationships:

< Less;

> Greater;

≤ Less or equal;

≥ Greater or equal;

= Equal.

These relationships are used for choosing the desired relationship when the constrains are described.

Restore an existing problem

The user of the optimization system (the Decision Maker) can suspend the process of finding the optimal solution in any time due to many different reasons – tiredness, necessity of getting addition information, etc. That is why there is an option to save the current problem at some intermediate state of the optimization process and continue working on it later on.

To restore an existing problem, the user must provide a valid user name and password to legitimate and then to choose the name of the problem. It is worthwhile to use its description as well. This is a convenient way to make some additional notes that are extremely helpful in searching the needed problem to restore.

Remove a problem

The user can remove some of the problems in any moment.

Change of password

The user can change the password.

There is a detailed help as well as a context help that can be activated by pressing *F*1 key.

It is well known that people perceive the information way differently. For some it is more convenient and clear to see the date digitally presented, while other prefer the graphical presentation and find it much more intuitive. That is why the results obtained from the system both the final and intermediate ones are visualized digitally as well as graphically. The user can choose between different types of charts and scales depending on the problem and the preferences. This makes it much easily to perceive the information, to observe the differences, to analyze and to make a decision.

3.3. Nonlinear optimization module

This module is not one of the main purposes in designing of the described optimization system. It can be changed as well as new methods can be added dynamically. Implementation of the main algorithm is independent of the structure of this module.

The idea is to describe the algorithm, usually an iterative process of communication between the computer program and the Decision Maker, that provides an optimal set of solutions among which DM have to choose one.

One such module could be briefly described as follows.

Generally this algorithm is a cyclic iterative process of communication between the Decision Maker and the computer program and includes the following basic steps:

- analysis of the problem;
- making a decision (according to the aspirations and preferences of the Decision Maker);
- optimization (finding of new optimal solution according to the Decision Maker and the additional information provided).

The first two steps are implemented by the Decision Maker and the third one – by the computer program.

On every step of the iterative process, the Decision Maker defines the preferences more precisely and as a result enters additional information. So, the computer program generates a better (according to the Decision Maker) solution.

The iterative process stops due to one of the following reasons:

- the computer program generates a solution that is optimal according to the Decision Maker;
- the Decision Maker realizes that it is worthless to continue as it is impossible to find a better solution;
- the Decision Maker suspend the process explicitly due to tiredness, lack of enough time, need of providing some additional information, etc. Current state can be saved, so the user can continue the optimization process later on.

As it was mentioned before, the algorithm described above is only an example of the basic form. Specific algorithms will be designed and added later on, the optimization system is open.

3.4. Single objective optimization method

It is very important to choose the basic single objective algorithm properly as it is essential for the overall efficiency of the optimization system.

The described software system uses Lagrange optimization method. This is a powerful method which finds an optimal solution quickly and is easy for describing the nonlinear constraints. The detailed description of the algorithm and its characteristics could be found in [8].

The module implements Lagrange multipliers method. There is an option to view intermediate results of the calculations, which is very useful for the Decision Maker to follow the implementation and make needful corrections.

4. Technical realization

The optimization system is implemented on *Borland C Builder* using MS-Windows 95/98/NT/2000 operation system. The information is described in tables structured as a DataBase that uses *Borland Interbase Server*. So it is flexible and easy to update. This eases the future Internet version that will provide some valuable advantages (some of them are described in the Introduction).

All the data are being saved together with the data for the corresponding user. Thus the data security in the system is guaranteed.

The parser module is implemented in C language as well. It takes care of the correctness of the data entered by the user in the text fields. Then the parser implements the symbol differentiation of the objective functions and constraints. And transfer the data to the solver module that calculates them.

The visualization module provides the option to choose between different types of charts to illustrate the obtained results graphically.

5. Conclusion

Described shell-system has user friendly interface, which helps the Decision Maker. The user doesn't need specific computer knowledge, but general skills in working with MS-Windows operating system. DM doesn't have to describe the information in a sophisticated way, only to choose between the generated solutions according to its preferences and aspirations or to indicate the needful changes in the values. Thus the interface is intuitive and very user friendly.

The opportunity for independent description of new multiobjective optimization algorithms offers a convenient way for testing and adjustments in the designing process. When the new module is developed it can be detailed examined and compared with other algorithms in terms of speed and efficiency. It can be easily changed updated. The final version of the algorithm can be added and it becomes a part of the system.

The system is flexible and with addition of new methods is continuously enriched. This gives additional option for the user to choose between all the available described algorithms.

The described optimization system is a convenient tool not only for laboratory research, but for the wide usage from making decision people as well.

References

1. Vassilev, V. Lection Notes on Multicriteria Decision Making.
2. Djambov, V., L. Kirilov, A. Atanassov. MONP-16 User's Guide.
3. Miettinen, K., Makela, M. M. Optimization System WWW-NIMBUS. University of Jyvaskyla, Report 9/1998.
4. Nimbus Home Page URL http://nimbus.math.jyu.fi/
5. Java Home Page URL http://java.sun.com
6. Bazar, M., K. Sheti. Nonlinear Programming. Theory and Algorithms, Moskva, Mir, 1982.
7. Garth, P., McCormick. Nonlinear Programming. Theory, Algorithms and Applications.
8. Bertsekas, D. Constrained Optimization and Lagrange Multiplier Methods, Massachusetts Institute of Technology. Gambridge, Academic Press, 1987 (Russian edition).

Программная система для решения многокритериальных оптимизационных задач

Милена Сарнева

Институт информационных технологий, 1113 София

(Резюме)

Обсуждается программная система для решения многокритериальных оптимизационных задач. Применяется новый подход с так называемой shell системой, которая помагает лицо принимающим решения. Она представляет оболочкой программной системы для принятия решений, в которой нет фиксированных оптимизационных алгоритмов. Основная идея является созданием удобного потребительского интерфейса и быстрый одно-критериальный алгоритм, при том нет необходимости дополнительного описания нелинейного алгоритма и его можно легко изменять.