*APPLIED COMPUTER SCIENCE*

**Vol. 7, No 1, 2011**

# IMPROVEMENTS METHODS IN MANUFACTURING DESIGN, SCHEDULING AND CONTROL

**Editors:**
**Zbigniew Banaszak, Antoni Świć**

**LUBLIN UNIVERSITY OF TECHNOLOGY**
**INSTITUTE OF TECHNOLOGICAL SYSTEMS OF INFORMATION**

**Sponsored by:** Lublin University of Technology, Poland



**Cover Design:** Dariusz PLINTA

**Typeset by:** Tomasz KUSZ



ISSN 1895-3735

ISBN 978-83-62596-32-4

WYDAWCA: Politechnika Lubelska
ul. Nadbystrzycka 38D, 20-618 Lublin
DRUK: Wydawnictwo-Drukarnia „Liber-Duo”
ul. Długa 5, 20-346 Lublin

# FOREWORD

The conditions of environment in which enterprises are working has drastically changed during last few decades. The basic factors which occurred as a new challenges for companies were: variability of customers' equirements and the same making shorter the products' life cycles, growing competiveness in global scale and the speed of technical progress do not meet up to now. Ipso facto the environment of functionality of enterprises has become extremely dynamic and directly the dynamics and variability of orders' realization varied into complexity of problems which instantly had to been solved by the companies. In many cases, these problems were non-trivial and its solving required analysis of huge number of solutions, frequently taking into account many contradictory and inverse criteria of assessment.

Inter alia, because of these reasons, the tools commonly used by designers and organizers of production became computer systems which assist decision processes and also planning and control of manufacturing processes. Simultaneously requirement for methodological solutions which allows fast analysis of many possible scenarios of solutions which are based on one or many criteria of evaluation was growing. Over a wide range, the scientific researches tending toward elaboration tools for finding importance of solutions when taking into account these criteria of evaluations were provided. In consequence of these fact the methods which have found wide applications are: mathematic methods of one- and multi-criteria optimization and also optimization methods of artificial intelligence (genetic algorithms, evolutionary algorithms, ant colony algorithms).

In this issue the possibility of application of methods for finding importance of solutions in problems of manufacturing systems' design, scheduling and control of manufacturing processes was presented. The contributed papers fall into four main problems area. The first paper presents reference model implemented in constraint programming techniques which can be successfully used for solving problems of rapid prototyping of alternative versions of project scheduling. The next paper presents possibility of using the evolutionary system of multicriteria analysis in problems of flexible manufacturing systems machine tools subsystem selection. The next group of three papers deals with problems of possibility of using the methods based on genetic algorithms and genetic programming in problems of both automated design of control systems and manufacturing processes scheduling.

The last paper emphasizes the importance of reliability of ERP systems and presents the method of standardized audit of ERP systems, safety and evaluation of these systems under the European and Polish norms.

We do hope that this issue will increase interest both among managers which deals with the problems of management and production engineering and also among scientific researcher for whom presented solutions will become the base for future works in finding new solutions which support processes of design and management of manufacturing processes.

Editors: Zbigniew Banaszak, Antoni Świć

Lublin, June, 2011

## CONTENTS

Marcin RELICH*

# CP-BASED DECISION SUPPORT FOR SCHEDULING

**Abstract**

*The paper presents the declarative approach to design of a reference model aimed at project prototyping. The reference model contains the finite set of decision variables, their domains and linking those constraints, i.e. can be seen as a kind of Constraint Satisfaction Problem. Consequently, the model considered can be treated as a knowledge base specifying both a class of enterprises and the projects that could be conducted on their base. So, the model provides a platform for rapid prototyping of alternative versions of project scheduling. The routine queries can be formulated in the straight or reverse way. In that context, the proposed reference model can be implemented in constraint programming (CP) techniques.*

## 1. INTRODUCTION

In the activity of present organizations more and more importance concerns unique activities, so-called projects. A project is a sequence of unique, complex, and connected activities having one goal or purpose and that must be completed by a specific time, within budget, and according to specification [9]. On account of this, the demand arises for new knowledge that enables the problems occurring in the realisation of unique projects to be solved. In this case, of particular significance is knowledge of project management that identifies factors which have an influence on the success or failure of the project, and that uses special methods and techniques.

Many cases of projects indicate that fewer than half of projects met cost and schedule targets [6, 11, 13, 16, 19]. The findings of various other authors indicate that projects which overrun are more common than projects which complete within original time scales, overruns likely to be between 40% and 200% [12]; for instance, only one third of World Bank projects met their aims, with typical delays of 50%. Another survey showing only 17% of projects meeting all three aspects of the project triangle (cost, time, and scope), with typical cost overruns as high as 189% [7]. In the case of software projects, the surveys on estimation performance report that 60-80 percent of all software projects encounter effort overruns [8, 10, 17].

Project success or failure depends on many critical factors, such as factors related to the project, availability of resources, project management, and the external environment [2, 13]. The reasons for project failure can be generally considered in availability of resources (e.g. human, financial, raw materials) and changeability of the external environment.

* Ph.D., Faculty of Economics and Management, University of Zielona Góra, m.relich@wez.uz.zgora.pl

Moreover, unstable requirements, lack of well-defined scope, quality of management, and skill of the employees can cause project failure. Another factor is that an enterprise which carries out a few projects can change the priority of the project.

The project requires planning that supports, among other things, the estimates of effort, resources, time, etc., which are fundamental to guide the project activities. To reduce project overruns, there are two ways to approach the problem. The first way is to increase the accuracy of the estimates through a better estimation process and the second, to increase the project control.

It is unrealistic to expect very accurate estimates of project effort because of the inherent uncertainty in development projects, and the complex and dynamic interaction of factors that influence its development. However, even small improvements will be valuable, especially if a project is connected with the large scale. More accurate forecasting supports the project managers in planning and monitoring the project, for instance in the project price set, resource allocation or schedule arrangement.

In the case of a significant difference between actual and planed project parameters, the manager should take a decision concerning the response to the change. The response can regard a support status quo, a correction of differences, a change of the norms, and also it may be connected with continuing the actual project. This approach is usually considered in the research works. The change of project scope can be another type of reaction regarding the performed variations. In this case, it seems important to build the approach that will generate a set of alternative projects and support the decision-maker. The alternative project is considered as a modification of the primary project, that can be made in different stages of the project life cycle, e.g. by the definition of the project or its implementation.

Rapidly changing expectations related to supporting strategic decisions, as well as aiming to reduce cost and investment risk, result in the need to make a task-oriented decision support system. Most of the publications have considered separately the fields of enterprise and project management. This results in a separate knowledge base respectively for an enterprise and project management. Consequently, it implies the difficulty of implementation of these fields within a single tool that is used for decision support. Hence, there is a need to build a single model that combines the fields of enterprise and project management, and that provides a base for making a task-oriented decision support system.

The paper is organized as follows. A reference model concerning an enterprise and project is presented in section 2. Scheduling in a form of the so-called constraint satisfaction problem is described in section 3. An example of the approach, which presents a possibility of decision problem specification in the straight and in the reverse way, is illustrated in section 4. Finally conclusions and future research are presented in section 5.

## 2. REFERENCE MODEL

The proposed approach combines the fields of an enterprise and project in single platform – the reference model. This type of approach seems to be natural in the case of an enterprise that executes projects and solves standard decision-making problems. In this way, a knowledge base is created that in addition to the inference strategies allows more efficient implementation of decision support system.

It is assumed that the reference model has the structure of constraints satisfaction problem (*CSP*), and it may be described as follows:

$$CSP = ((V, D), C) \quad (1)$$

where:

$V = \{v_1, v_2, ..., v_n\}$ – finite set of $n$ variables,

$D = \{D_1, D_2, ..., D_n\}$ – finite and discrete domains $D$ of variables, where $D_i = \{d_{i1}, d_{i2}, ..., d_{ir}\}$,

$C = \{c_1, c_2, ..., c_m\}$ – finite set of $m$ constraints binding decision variables.

Each constraint treated as a predicate can be seen as an $n$-ary relation defined by a Cartesian product $D_1 \times D_2 \times ... \times D_n$. The solution to the *CSP* is a vector $(d_{1i}, d_{2k}, ..., d_{nj})$ such that the entry assignments satisfy all the constraints $C$. So, the task is to find the values of variables satisfying all the constraints, i.e., a feasible valuation. Generally, the constraints can be expressed by arbitrary analytical and/or logical formulas as well as bind variables with different non-numerical events.

Thus, a constraint can be treated as a logical relation among several variables, each one taking a value in a given (usually discrete) domain. To solve such a problem stated by the set of requirements (constraints) that specify a problem at hand, the concept of constraint programming (*CP*) is employed. *CP* is an emergent software technology for declarative description *CSP* and can be considered as a pertinent framework for development of decision support system software aims. The main idea behind the *CP* concept is based on subsequent phases of constraint propagation and variable distribution [14].

Construction of the reference model requires certain assumptions concerning the structure of the modelled object and the tasks performed in it. It is assumed that the client orders may be taken and commenced at any time (possibly adding the new projects to a set of projects already in progress). The expenses regarding an order are paid from the enterprise's own means or from a bank loan. The budget of the project is set with cash flow budget in the investment period. The client order is chosen by the profitability analysis and technical realizability. The enterprise receives the order specification with the client requirements, regarding among others the scope, price and time completion of project.

The enterprise model can be described by its resources. The project model is created from the requirements of the client. In the model, some parameters are determined, among which a set of constraints and decision variables may be distinguished (Fig. 1). The constraints connect the variables that describe the capacity of the enterprise, as well as the variables that concern the conditions of project completion. For instance, the number of the enterprise's employees limits the duration of the project.

It means that fulfilment of specified constraints enables project completion according to client requirements. The enterprise and project model containing examples of decision variables and constraints is shown in Fig. 1.

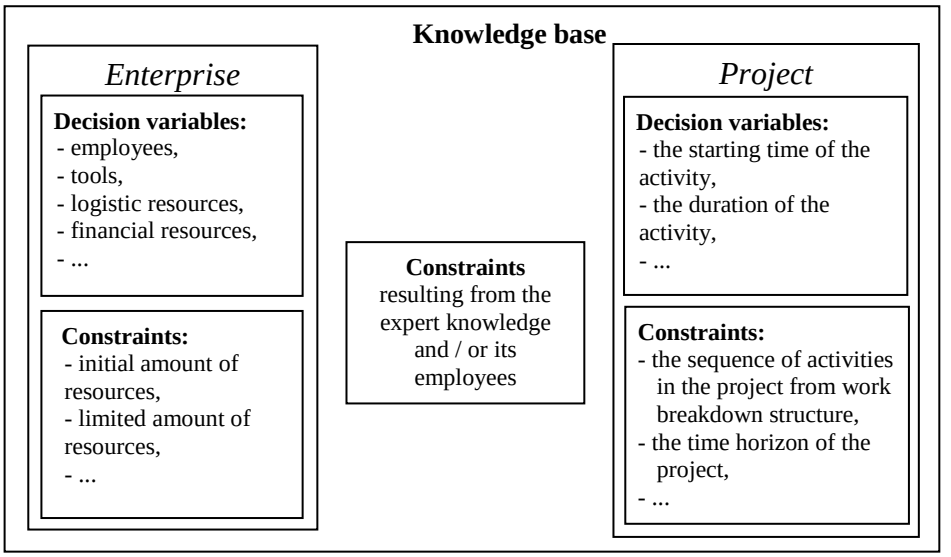


**Fig. 1. Enterprise-project model as a common knowledge base**

The assumed model enables descriptive approach to the problem statement, encompasses constraint satisfaction problem structure and then allows implementation of the problem considered in the constraint programming environment. The idea behind the proposed approach assumes the system considered can be represented in terms of a knowledge base (*KB*). *KB* comprises of facts and rules determining the system's properties and relations linking them respectively. Taking into account the concept of constraints propagation and variables distribution following from the constraint programming languages it is easy to note that any *KB* can be represented in a standard form of the *CSP* [18].

*KB* can be specified in terms of a system [5]. At the input of the system are the variables regarding the fundamental attributes of the object that are known and given by the user. In the considered *KB* for the enterprise-project model, there are, for example, variables concerning the amount of an enterprise's resources or the project structure. The output of the system is described by the attributes of the object that are unknown or are only partially known. In the considered case, there can be included variables regarding e.g. the cost or time of activity, use of resources or the level of investment performance indicators.

Classification of the decision variables in *KB* as input-output variables is arbitrarily made and allows the formulation of two classes of standard queries, in a straight and in a reverse way, as follows [1, 4]:

- a straight way (i.e. corresponding to the question: what results from premises?), e.g. Does a given resources allocation guarantee the schedule does not exceed the given deadline?

- a reverse way (i.e. corresponding to the question: what implies conclusion?), e.g. What activity duration times and resources amount guarantee the given schedule does not exceed the deadline?

The above-mentioned categories encompass the different reasoning perspectives, i.e. forward and backward reasoning. The corresponding queries can be stated in the same model that can be treated as composition of variables and constraints, i.e. assumed sets of variables

and constraints limiting their values. In that context, the problem statement of scheduling, which is specified in terms of CSP, is presented in next section.

## 3. CONSTRAINTS SATISFACTION PROBLEM FOR SCHEDULING

Given amount $z$ of discrete resources $r_k$ specified by (e.g. workforce, tools, money): $R = (r_1, r_2, ..., r_z)$. Given amounts $q_{k,h}$ of available resources at the moment of $H$: $H = \{0, 1, ..., h\}$. Given a project $P_i$ is specified by the set composed of $l$ activities: $P_i = \{A_{i,1}, A_{i,2}, ..., A_{i,l}\}$. The activity $A_{i,j}$ is specified as follows:

$$A_{i,j} = (s_{i,j}, t_{i,j}, Tp_{i,j}, Tz_{i,j}, Dp_{i,j}) \tag{2}$$

where:

$s_{i,j}$ – the starting time of the activity $A_{i,j}$, i.e., the time counted from the beginning of the time horizon $H$.

$t_{i,j}$ – the duration of the activity $A_{i,j}$.

$Tp_{i,j} = (tp_{i,j,1}, tp_{i,j,2}, ..., tp_{i,j,z})$ – the sequence of moments the activity $A_{i,j}$ requires new amounts of resources: $tp_{i,j,k}$ – the time counted since the moment $s_{i,j}$ of the $dp_{i,j,k}$ amount of the $k$-th resource allocation to the activity $A_{i,j}$. That means a resource is allotted to an activity during its execution period: $0 \le tp_{i,j,k} < t_{i,j}$; $k = 1, 2, ..., z$.

$Tz_{i,j} = (tz_{i,j,1}, tz_{i,j,2}, ..., tz_{i,j,z})$ – the sequence of moments the activity $A_{i,j}$ releases the subsequent resources: $tz_{i,j,k}$ – the time counted since the moment $s_{i,j}$ of the $dp_{i,j,k}$ amount of the $k$-th resource was released by the activity $A_{i,j}$. That is assumed a resource is released by activity during its execution period: $0 < tz_{i,j,k} \le t_{i,j}$; $tp_{i,j,k} < tz_{i,j,k}$; $k = 1, 2, ..., z$.

$Dp_{i,j} = (dp_{i,j,1}, dp_{i,j,2}, ..., dp_{i,j,z})$ – the sequence of the $k$-th resource amounts $dp_{i,j,k}$ are allocated to the activity $A_{i,j}$: $dp_{i,j,k}$ – the amount of the $k$-th resource allocation to the activity $A_{i,j}$. That assumes: $0 \le dp_{i,j,k} < q_k$; $k = 1, 2, ..., z$.

The constraints regarding the enterprise include the initial and available amounts of the resources. Moreover, the project portfolio should be completed within the given time horizon $H = \{0, 1, ... , h\}$. It is assumed the activities cannot be suspended during their execution, and also:

- each activity can request any kind and quantity (not exceeding the resource's limited amount) of any resource,
- each resource can be uniquely used by an activity,
- the quantity of resource used by an activity cannot be changed or allotted to other activity,
- an activity can start its execution only if required amounts of resources are available at the moments given by $Tp_{i,j}$.

The following activities order constraints are considered:

- the $k$-th activity follows the $i$-th one:

$$s_{i,j} + t_{i,j} \le s_{i,k} \tag{3}$$

- the $k$-th activity follows other activities:

$$\begin{gathered} s_{i,j} + t_{i,j} \le s_{i,k} \\ s_{i,j+1} + t_{i,j+1} \le s_{i,k} \\ ... \\ s_{i,j+n} + t_{i,j+n} \le s_{i,k} \end{gathered} \tag{4}$$

– the $k$-th activity is followed by other activities:

$$\begin{gathered} s_{i,k} + t_{i,k} \leq s_{i,j} \\ s_{i,k} + t_{i,k} \leq s_{i,j+1} \\ \ldots \\ s_{i,k} + t_{i,k} \leq s_{i,j+n} \end{gathered} \tag{5}$$

According to (1) the reference model for scheduling can be described as follows:
A set of decision variables $V$:
- the starting time of the activity $s_{i,j}$
- the duration of the activity $t_{i,j}$
- resources $z$, $Tp_{i,j}$, $Tz_{i,j}$, $Dp_{i,j}$

$$V = (s_{i,j}, t_{i,j}, z, Tp_{i,j}, Tz_{i,j}, Dp_{i,j}) \tag{6}$$

The values sets of variables $V$ is specified by the set of domains:

$$D = (D_{si}, D_{ti}, D_z, D_{Tpi}, D_{Tzi}, D_{Dpi}) \tag{7}$$

Note that for the known values of decision variables (e.g. for a variable concerning available amounts of $z$ resources), the domain is a set with single element.

A set of constraints $C$ includes the constraints regarding an enterprise and a project, for instance, the constraints concerning the sequence of activities, the cost or available amounts of the resources. Some of the constraints link the field of enterprise with project, e.g. the number of available employees.

$C = \{C_1, C_2\}$, where:

$C_1$: $H = \{1, ..., h\}$ – the constraint of the project horizon $H$,

$C_2$: $s_{i,j} + t_{i,j} \leq s_{i,k}$ - the constraint of the activities sequence in the project.

An answer to the following question is sought: does a given resources allocation guarantees the project completion by assumed constraints, and if so, what are its parameters?

This question can be expanded to the next, for instance, does a given resources allocation not exceed the given deadline $H$ and the given financial resources $q$ in time unit $h$? It allows a class of multicriteria problems to be taken into consideration.

The examples regarding the above-described problem are presented in next section.

## 4. ILLUSTRATIVE EXAMPLES

The example aims to illustrate a possibility of *CSP* specification for decision problem of project planning in the straight and in the reverse way. It assumes, the activities compete with the access to the discrete resources. In the example, single project with nine activities $P = \{A_1, ..., A_9\}$ is considered that network is presented in Fig. 2. Bold lines represent the critical path.

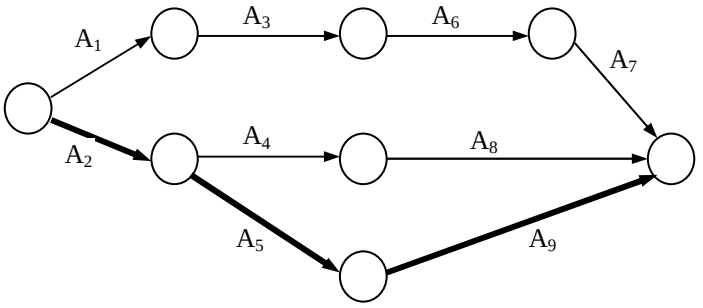


**Fig. 2. Activity network of project**

### 4.1. Routine queries formulated in the straight way

Example 1

Operation times for the project by the following sequence are determined: $T$ = (3, 4, 2, 2, 3, 3, 1, 4, 5). Moreover, given the time horizon $H$ = {0, 1, ..., 15}, and resource $r$ that is limited by 26 units. Number of resource is constant in whole time horizon $H$. It assumes, an amount of resource is allocated to an activity at the moment of its beginning and can be released only by this activity at the moment of its completion. The required number of resource from the database of past projects, which belong to the same class as considered project, is determined. The resource according to linear function is calculated as follows: $dp_j = 2 + 2 \cdot t_j$. Thus, the sequence of the resource amounts allocated to the activity $j$ is following: $Dp$ = (8, 10, 6, 6, 8, 8, 4, 10, 12).

The order constraints according to the activity network of the project and formulas (3), (4), and (5) are following:

$C_1$: $s_3 \geq s_1 + t_1$, $C_2$: $s_4 \geq s_2 + t_2$, $C_3$: $s_5 \geq s_2 + t_2$, $C_4$: $s_6 \geq s_3 + t_3$, $C_5$: $s_7 \geq s_6 + t_6$, $C_6$: $s_8 \geq s_4 + t_4$, $C_7$: $s_9 \geq s_5 + t_5$.

The considered problem belongs to the class of “straight” ones where for a given parameters describing the enterprise-project system the activities schedule is sought. It reduces to the following question: is there, and if so, what form does a schedule have that completion time does not exceed the deadline $H$, and that fulfils the resource constraints? Note the answer to above-mentioned question is connected with determination of the starting time of the activity $s_j$, where $0 \leq s_j < 15$; $j$ = 1, 2, ..., 9.

The obtained solution follows from model implementation in the *CSP*-based reference model and programmed in Oz Mozart. The first admissible solution has the following form: $S$ = (0, 0, 3, 4, 4, 5, 8, 6, 8). The project schedule fulfilled all constraints imposed by an enterprise capability and project requirements, is presented in Fig. 3.

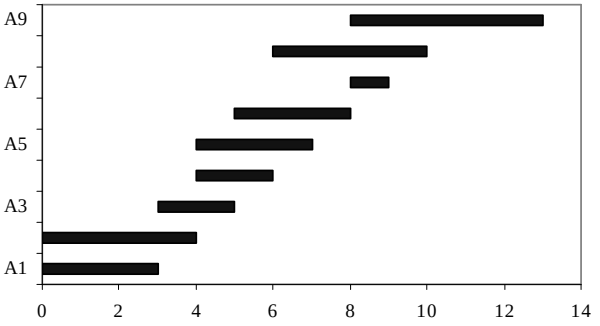


**Fig. 3. Gantt's chart of project**

The level of resource usage containing the assumed resource's limit in the time horizon is illustrated in Fig. 4.

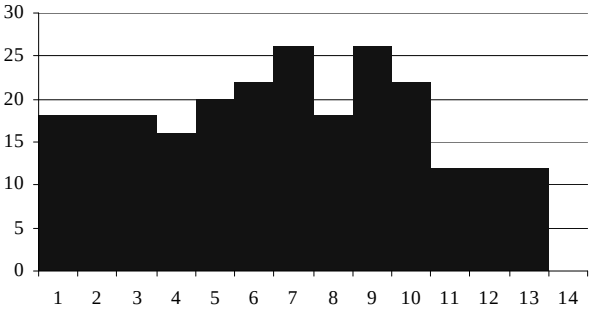


**Fig. 4. Gantt's-like chart of the resource usage**

Example 2

Given the project *P* specified by the same activity network, time horizon, durations of the activities, and amount of the resource allocated to the activity as in Example 1. However, the new limit of resource ($r \leq 20$) is considered.

The considered problem also belongs to the class of "straight" ones, and it can be reduced to the following question: is there, and if so, what form does a schedule have that completion time does not exceed the deadline *H*, and that fulfils the resource constraints?

Similarly to the previous case, the solution results in a determination of the beginning moments of the activities $s_j$, however regards smaller amount of the resource. By this constraint, the set of admissible solutions is empty. This means there is no schedule. Thus, there is a possibility to reformulate the considered problem by stating it in a reverse way, i.e. the way aims to search for decision variables (e.g. amount of resource for the activity) guaranteeing that the completion time of the project does not exceed the assumed deadline *H*. This way is considered in next subsection.

### 4.2 Routine queries formulated in the reverse way

Given the project $P$ specified by the same activity network, time horizon, durations of the activities and limit of the resource ($r \leq 20$) as in Example 2. Amounts of the resource allocated to the activities are not known, however the constraint determining the amounts is given. According to the database of past project, the relationship between an amount of the resource and a duration of the $j$-th activity has been determined as follows: $dp_j = a + b \cdot t_j$, where $a = \{1, 2\}$ and $b = \{2, 3\}$.

Taking into account above-mentioned assumptions, the problem reduces to the question: what amounts of the resource allocated to the activities $dp_j$ guarantee that completion time of a schedule does not exceed the deadline $H$, and resource limit $r$?

In order to response to this question the values of the following sentences are sought: $Dp = (dp_1, \ldots, dp_9)$ and $S = (s_1, \ldots, s_9)$. The reference model encompassing assumption of the considered example was implemented in Oz Mozart programming environment, and the obtained solution is following: $Dp = (7, 9, 5, 5, 7, 7, 3, 9, 11)$ and $S = (0, 0, 3, 4, 4, 5, 8, 7, 9)$. The project schedule fulfilled all constraints is presented in Fig. 5.

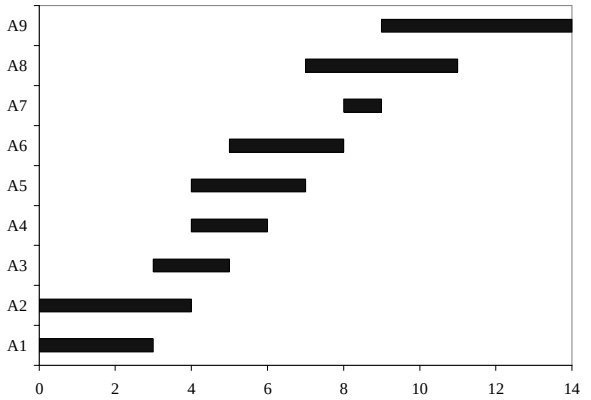


**Fig. 5. Gantt's chart of project**

The chart illustrating the changes of resource usages, by assumed resource's limit and the time horizon, is presented in Fig. 6.

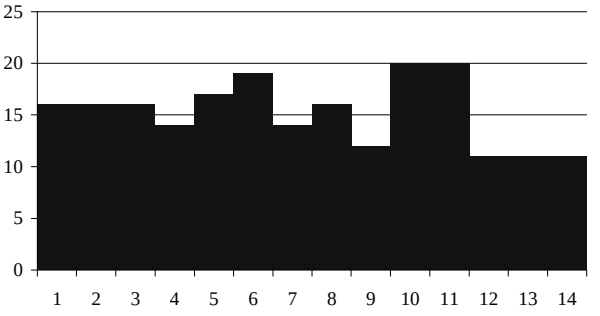


**Fig. 6. Gantt's-like chart of the resource usage**

The assumed ranges of decision variables and constraints determine the possible values of sought parameters. The result is a set of feasible solutions in time unit *h*. This set can be empty, or with one or many solutions. Note that the number of generated solutions depends not only on the knowledge base, but also on a user-declared granularity of solutions in constraint programming languages such as, for instance, ILOG or Oz Mozart [15].

## 5. CONCLUSIONS

In the present, changeable business environment, quickness of response to customer needs or pressure on innovation and effective cost management determine the success or failure in the struggle for market position. This forces more frequent and larger-scale changes in contemporary organizations. The answer to these new challenges is the application of the principles of project management. In the case of projects carried out on a client order, erroneous estimation of expenditures and project deadlines may result penalties being accrued, as agreed upon in the contract or covering the costs with the company's own money. A wrong decision may worsen the liquidity of an enterprise or even lead to its bankruptcy. In this situation, it seems extremely important to support the decision maker.

The proposed approach assumes a kind of reference model encompassing open structure, enabling one to take into account different sorts of variables and constraints as well as to formulate straight and reverse kinds of project planning problems.

Since a constraint can be treated as a logical relation among several variables, each one taking a value in a given (usually discrete) domain, the idea of *CP* is to solve problems by stating the requirements (constraints) that specify a problem at hand, and then finding a solution satisfying all the constraints. Because of its declarative nature, it is particularly useful for applications where it is enough to state *what* has to be solved instead of *how* to solve it [1].

The advantages of the proposed approach include the possibility of the description of enterprise and project management in terms of a knowledge base. Moreover, in the presented approach it is possible to obtain a set of feasible solutions in the different phases of the project life cycle. This is especially attractive in the absence of the possibility of continuing the project in its primary form and can support the decision maker in choosing the alternative project.

Further research focuses on the presentation of the model reference for the project management problem in a dynamic form, taking into account the subsequent project management functionality and assessing their impact on the set of feasible solutions. It should also define criteria for evaluating project alternatives, and carrying out verification of the knowledge base of described object.

## REFERENCES


1. BANASZAK Z., BOCEWICZ G., BACH I.: *CP-driven production process planning in multiproject environment*, Decision Making in Manufacturing and Services, pp. 5-32, Vol. 2, No.1-2, 2008.
2. BELASSI W., TUKEL O.I.: *A new framework for determining critical success/failure factors in projects*, International Journal of Project Management, pp. 141-151, Vol. 14, 1996.
3. BERTALANFFY L.: *General system theory - A Critical Review*, General Systems, pp. 1-20, Vol. 7, 1962.

4. BOCEWICZ G., BACH-DĄBROWSKA I., BANASZAK Z.: Declarative approach to computer aided project planning systems design. Akademicka Oficyna Wydawnicza EXIT, Warsaw 2009. (In Polish).
5. BUBNICKI Z.: *Logic-algebraic method for knowledge-based relation systems,* Systems Analysis Modelling and Simulation, pp. 21-35, Vol. 33, 1998.
6. COOKE-DAVIES T.J., ARZYMANOW A.: *The maturity of project management in different industries: an investigation into variations between project management models,* International Journal of Project Management, pp. 471-478, Vol. 21, 2003.
7. JORGENSEN M., SJOBERG D.I.: *The impact of customer expectation on software development effort estimates,* International Journal of Project Management, pp. 317-325, Vol. 22, 2004.
8. JORGENSEN M., GRUSCHKE T.: *The impact of lessons-learned sessions on effort estimation and uncertainty assessments,* IEEE Transactions on Software Engineering, pp. 368-383, Vol. 35, No. 3, 2009.
9. Meredith J.R., Mantel S.J.: *Project Management – a managerial approach*. Third Edition, John Wiley and Sons, New York 1995.
10. MOLOKKEN-OSTVOLD K., JORGENSEN M.: *A comparison of software project overruns,* IEEE Transactions on Software Engineering, pp. 754-766, Vol. 31, No. 9, 2005.
11. NITITHAMYONG P., SKIBNIEWSKI M.J.: *Success/failure factors and performance measures of web-based construction project management systems: professionals' viewpoint,* Journal of Construction Engineering and Management, pp. 80-87, Vol. 132, 2006.
12. REICHELT K., LYNEIS J.: *The dynamics of project performance: benchmarking the drivers of cost and schedule overrun,* European Management Journal, pp. 135-150, Vol. 17, 1999.
13. ROBERTSON S., WILLIAMS T.: *Understanding project failure: using cognitive mapping in an insurance project,* Project Management Journal, pp. 55-71, Vol. 37, 2006.
14. ROSSI F.: *Constraint (logic) programming: A survey on research and applications,* New Trends in Constraints (Apt K.R. et al.), pp. 40-74. Springer-Verlag, Berlin 2000.
15. SCHUTLE H., SMOLKA G., WURTZ J.: *Finite Domain Constraint Programming in Oz.* German Research Center for Artificial Intelligence, Saarbrucken 1998.
16. SHORE B.: *Systematic biases and culture in project failures,* Project Management Journal, pp. 5-16, Vol. 39, 2008.
17. SINGH R., KEIL M., KASI V.: *Identifying and overcoming the challenges of implementing a project management office,* European Journal of Information Systems, pp. 409-427, Vol. 18, 2009.
18. VAN ROY P., HARIDI S.: *Concepts, techniques and models of computer programming*. Helion, Gliwice 2005.
19. YEO K.T.: *Critical failure factors in information system projects,* International Journal of Project Management, pp. 241-246, Vol. 20, 2002.

*Keywords: machine tools selection, Evolutionary System of Multicriteria Analysis, ESAW*

Arkadiusz GOLA[*], Jerzy MONTUSIEWICZ[**], Antoni ŚWIĆ[***]

# COMPUTER AIDED FMS MACHINE TOOLS SUBSYSTEM SELECTION USING THE EVOLUTIONARY SYSTEM OF MULTICRITERIA ANALYSIS

**Abstract**

*One of the key problems in the area of flexible manufacturing systems (FMS) design is a problem of proper design of manufacturing subsystem and especially the machine tools selection. Although the problem seems to be simple, in fact it is difficult to solve because of large variety and number of parameters and also brief foredesign which are highly influential for the decision. This study shows possibility of implementation the Evolutionary System of Multicriteria Analysis <ESAW> for defining the importance of solutions in the process of casing-class FMS machine tools selection.*

## 1. INTRODUCTION

One of the key problems in the area of Flexible Manufacturing Systems (FMSs) design is a problem of manufacturing subsystem design and especially machine tools selection for designed FMS. It is the first and very important step which determines the system effectiveness to large extent. The proper selection of machine tools subsystem could both significantly minimize investments for construction, as well as lead to minimization of costs of system operation or make the most of machines. Moreover the purchased machinery stock directly determines the efficiency, automation and flexibility level of the whole FMS and the result of this step is a foundation for designing the residual subsystems of flexible manufacturing system [21].

Although the problem seems to be simple, selection of proper machine tools for designed system is not an easy one. The basic resource of the problem is a great variety and number of parameters and also complexity of design conditions which are need to be taken into account during the selection process. Therefore appears the necessity of using the formalized optimization methods which assist to find the best solution in the process of FMS machine-tools subsystem design.

[*] D.Sc., Eng., Lublin University of Technology, Faculty of Management, Department of Enterprise Organization, ul. Nadbystrzycka 38, 20-618 Lublin, e-mail: a.gola@pollub.pl

[**] D.Sc., Eng., Lublin University of Technology, Faculty of Fundamentals of Technology, Department of Fundamentals of Technology, ul. Nadbystrzycka 38, 20-618 Lublin, e-mail: j.montusiewicz@pollub.pl

[***] D.Sc., Eng., (Assoc. Prof.), Lublin University of Technology, Faculty of Mechanical Engineering, Institute of Technological Systems of Information, ul. Nadbystrzycka 36, 20-618 Lublin, e-mail: a.swic@pollub.pl

When taking into account that machine tools selection process is realized using more than one criterion of evaluation of solutions – the useful are methods of multicriteria analysis [9,17,24].
Various researchers have studied to determine the suitable equipment for the different manufacturing facilities using mathematical models, heuristic algorithms and MCDM methods. Some of them have been focused on machine tool selection directly. Several studies regarding the machine tool selection problem can be given as follows. Lin an Yang [12] presented a machine selection model from a range of machines for the manufacture of particular part types using the AHP method. Tabucanon et al. [20] developed a decision support framework for selecting the most appropriated machines in flexible manufacturing systems (FMS). Atamani and Lashkari [2] developed a model for machine tool selection and operation allocation in FMS. Wang et al. [22] presented fuzzy multiple attribute decision making model to select the appropriate machines for FMS. Fuzzy technique for order preference by similarity to ideal solution (TOPSIS) presented Onut at al. [16]. Arslan et al. [1] presented a muliti-criteria weighted average (MCVA) method for machine tool selection. Yourdakul [23] proposed a model linking machine alternatives to manufacturing strategy for machine tool selection. In that study, evaluation of machine tool alternatives was modelled considering strategic implications of the machine tool selection decisions by using the AHP method. Ayag and Ozdemir [3] used the fuzzy AHP technique to weight the machine tool alternatives under eight main and nineteen subcriteria and then carried out benefit/cost ratio analysis by using both the fuzzy AHP score and procurement cost of each alternative. By using the same criteria again, Ayag [4] proposed a hybrid approach, which integrates the AHP with simulation techniques, to determine the best machine tool satisfying the needs and expectations of a manufacturing organization among set of possible alternatives in the market. Mishra et al. [13] suggested a fuzzy goal-programming model having multiple conflicting objectives and constraints pertaining to the machine tool selection and operation allocation problem, and used a random search optimization methodology. Chan and Swarknar [6] presented a fuzzy goal programming approach to model the machine tool selection and operation allocation problem of FMS. An ant colony optimization based approach was also applied to optimized the model. Cimren at al. [7] proposed a decision support system for machine tool selection using the analytic hierarchy process. Dagdeviren [8] presented an integrated approach which employs analytic hierarchy process (AHP) and preference ranking organization method for enrichment evaluations (PROMETHEE) together for the equipment selection problem. Selection of a machine tool for FMS using ELECTRE III presented Balaij at al. [5]. Rao and Parnichkun [18] presented a methodology based on a combinatorial mathematics-basede decision method for evaluation alternative flexible manufacturing systems.
Although there were a number of publications evaluating the machine tools alternatives in the literature, many of them have been prepared using the MCDM methods considering human judgments, tangible, intangible and multiple criteria. In this paper the possibility of implementation the Evolutionary System of Multicriteria Analysis for the defining the importance of solutions in the process of casing-class FMS machine tools selection was shown. In particular, the issue of the process of machine tools selection, the essence of Evolutionary System of Multicriteria Analysis and solutions of the process of defining the importance of solutions for selected decision problem were presented.

## 2. THE ALGORITHM OF THE PROCESS OF CASING-CLASS FMS MACHINE TOOLS SELECTION

The process of selection of machine tools subsystem for designed casing-class FMS is implemented using the assumptions of the methodology presented in works [9,19]. The selection is realized according the four-stages algorithm presented in fig. 1.

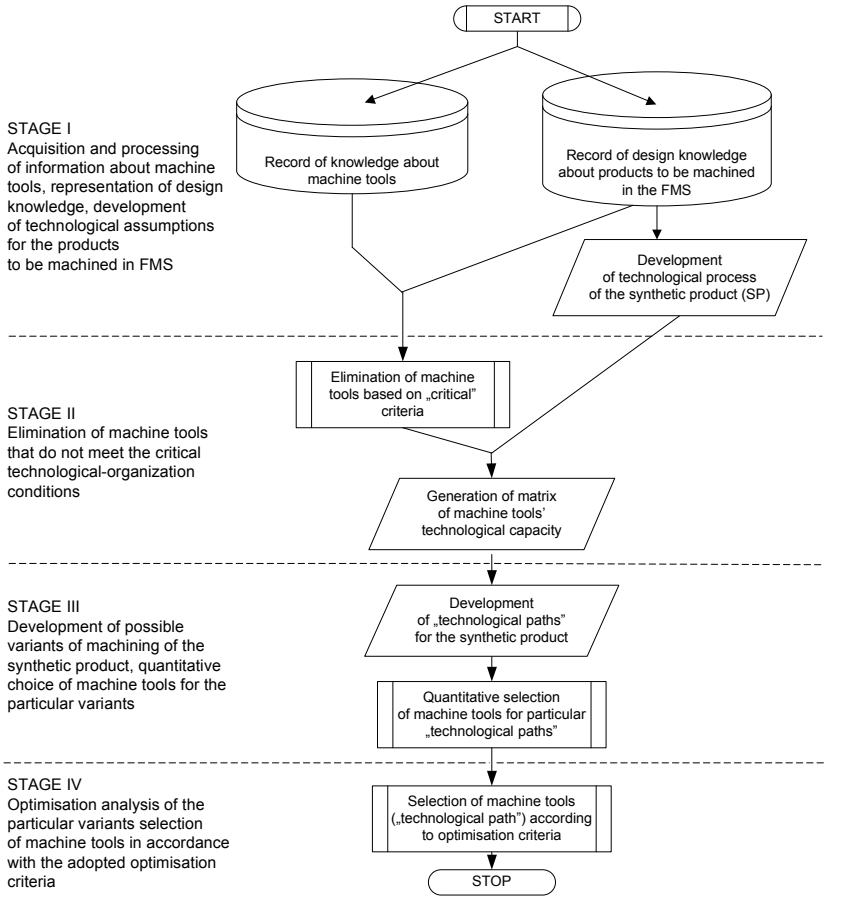


**Fig.1. Main algorithm of the methodology of machine tools selection in casing-class FMS [9,19]**

The first step in the process of selection is the preparation of a record of knowledge about all machines tools from among which the choice is to be made $O = \{o_1, o_2, \ldots o_n\} = \{o_i\}$, products to be machined in the FMS being designed $W = \{w_1, w_2, \ldots, w_t\} = \{w_\alpha\}$ and development and saving of technological process of the synthetic product (SP).

In the second stage elimination from the O database of those machine tools that are incapable of producing the parts that are to be machined in the system, based on certain limit

criteria ("critical" criteria) is realized. In accordance with the adopted assumptions, we should eliminate from the database those machine tools that:

1. Do not meet the limit conditions resulting from the technical parameters of products to be machined in FMS.
2. Do not meet the limitations imposed by the user and/or designer of the flexible manufacturing system.
3. Do not have the design-technological capabilities to perform the machining operations provided for realization within the process of manufacturing.

Those machine tools that „remain" in the database after the stage of elimination constitute of set of machine tools that are taken into consideration at further stages of selection ($X = \{x_1, x_2, \ldots x_m\} = \{x_k\}$).

Machine tools which meet the critical conditions are saved in the set of technological machines $X = \{x_1, x_2, \ldots x_m\} = \{x_k\}$. On the base of X set and the developed technological process of synthetic product the $A_{kj}$ [0-1] matrix of machine tools capabilities is generated. The matrix defines which of the machine tools has the ability to realize specified cut from the technological process of WS.

In the stage three the generation of technological paths and the quantitative selection of machine tools for particular technological paths is realized. Technological paths determines possible ways of going the synthetic product through the system, i.e. following machine tools which realizes following cuts in the technological process of WS. Technological paths and the results of quantitative selection of machine tools which is realized using the method of balancing the burden level of particular machine tools with the manufacturing tasks forms solutions to be analyzed in fourth stage of methodology.

The last step in the process of selection is a choice the best solution using the accepted criteria of evaluation. The optimization criteria (target functions) in presented model are as follows:

1) Minimisation of total costs of machine tools acquisition and operation (per annum) calculated using formula (1):

$$F_1(M_\mu) = \sum_{k=1}^{m} \{L_k[(C_k * a_{ok}) + k_{sk}]\} \rightarrow \min \tag{1}$$

where: $L_k$ – number of *k* machine tools, $C_k$ – total purchasing price of *k* machine tool, $a_{ok}$ – annual depreciation rate of *k* machine tool, $k_{sk}$ – average annual cost of service for *k* machine tool.

2) Minimization of time of machining (throughput time) of synthetic product (exclusive of inter-cut transport and storage operations time) – calculated using formula (2):

$$F_1(M_\mu) = \{[\max(t_{wnk}; t_{wpk}) + t_{1k}] + \sum_{j=2}^{z} \{\lambda * \max(t_{wnk}; t_{wpk}) + [(1-\lambda) * t_{wnk}] + t_{jk}\}\} \rightarrow \min \tag{2}$$

where:

value λ assumes the following values:

$$\lambda = \begin{cases} 0 & \text{, when cut } \delta_j \text{ is realized on the same machine tool as cut } \delta_{j-1} \\ 1 & \text{, when cut } \delta_j \text{ is realized on another machine tool than cut } \delta_{j-1} \end{cases}$$

$t_{wnk}$ – tool change time „from chip to chip" on *k* machine tool, $t_{wpk}$ – technological palette change time on *k* machine tool, $t_{1k}$ – unit time of realization of first operations in technological process of synthetic product on *k* machine tool, $t_{jk}$ – unit time of realization of *j* cut on *k* machine tool.

# 3. STRUCTURE AND CHARACTERISTICS OF THE EVOLUNTARY SYSTEM OF MULTICRITERIA ANALYSIS

To solve the task of optimization defined in section 2 (stage 4) the Evolutionary System for Multicriteria Analysis <ESAW> was used. The system takes advantage of many different cooperating with each other methods and enables to generate one solution or small set of solutions, optimal in Pareto sense which are not much sensitive for changing the preferences for criteria given by experts [14].
The Evolutionary System of Multicriteria Analysis was built taking into account the internal features included both into analyzed values of solutions and parameter given in percentage. Values of evaluation of solutions decide of position of ideal vector, which is a basic reference point in the Compromise Solution Determination Method. The indistinctive interval given in percentage enables filtration of solutions using the Undifferentation Interval Method. The final effect of filtration depends both on the defined value of indistinctive interval and mutual position of analyzed valuation of solutions in the criteria space [15].
The Evolutionary System of Multicriteria Analysis includes following methods: the Boundary Value Method (BVM), the Ideal Point Definition Method (IPDM), the Undifferentation Interval Method (UIM) and the Compromise Solution Determination Method (CSDM) (fig. 2).

- ***Boundary Value Method (BVM)***
  BVM eliminates undominated solutions, which values of rate are located on the extreme border of set of undominated solutions along orthogonal directions of components of criteria vector – i.e. values of solutions which determine the corner points and these one which are located in its neighborhood [14]. The values of solutions which determine the corner points usually defines the ideal value (ideal vector), so its elimination causes necessity of determining new ideal vector. BVM is over a wide range similar to formulated in an area of one-criterion and multicriteria optimization task of satisfaction [15]. In a task of multicriteria optimization occurs the vector target function $F(x) = [F_1(x), F_2(x), \ldots, F_j(x)]^T$, it is needed to specify $j$ satisfactory values $f_{sj}$. (where $j \in J = \{1, 2, \ldots, J\}$ is a number of target function). The task of satisfaction assumes the shape as follows:

$$F(x_s) = \underset{x \in X}{sat}\ F(x) \qquad , \tag{3}$$

$$\underset{x \in X}{sat}\ F(x) = \begin{cases} F_j \le f_{sj} \quad in \quad task \quad \min \quad F_j(x),\ j \in J = \overline{1, J} \\ F_j \ge f_{sj} \quad in \quad task \quad \min \quad F_j(x),\ j \in J = \overline{1, J} \end{cases}$$

  where: $F_j$ – $j$ component of the target function, x – vector of decision variables, $f_{sj}$ – $j$ satisfactory value of crierion, xs – vector of decision variables for which the target function F(x) take the favourable value in comparison with previously selected satisfactory value.

- ***Ideal Point Definition Method (IPDM)***
  In the IPDM method the situation is reversed. It was proposed to treat the referential point which is the positive standard as a new ideal point. Accepted ideal point chooses from the set of valuations of undominated solutions the subset of valuations of solutions which satisfy the conditions that any of component values will not be adequately lesser (or larger) than the value of component of ideal point (depending if the task is the minimization or maximization one).

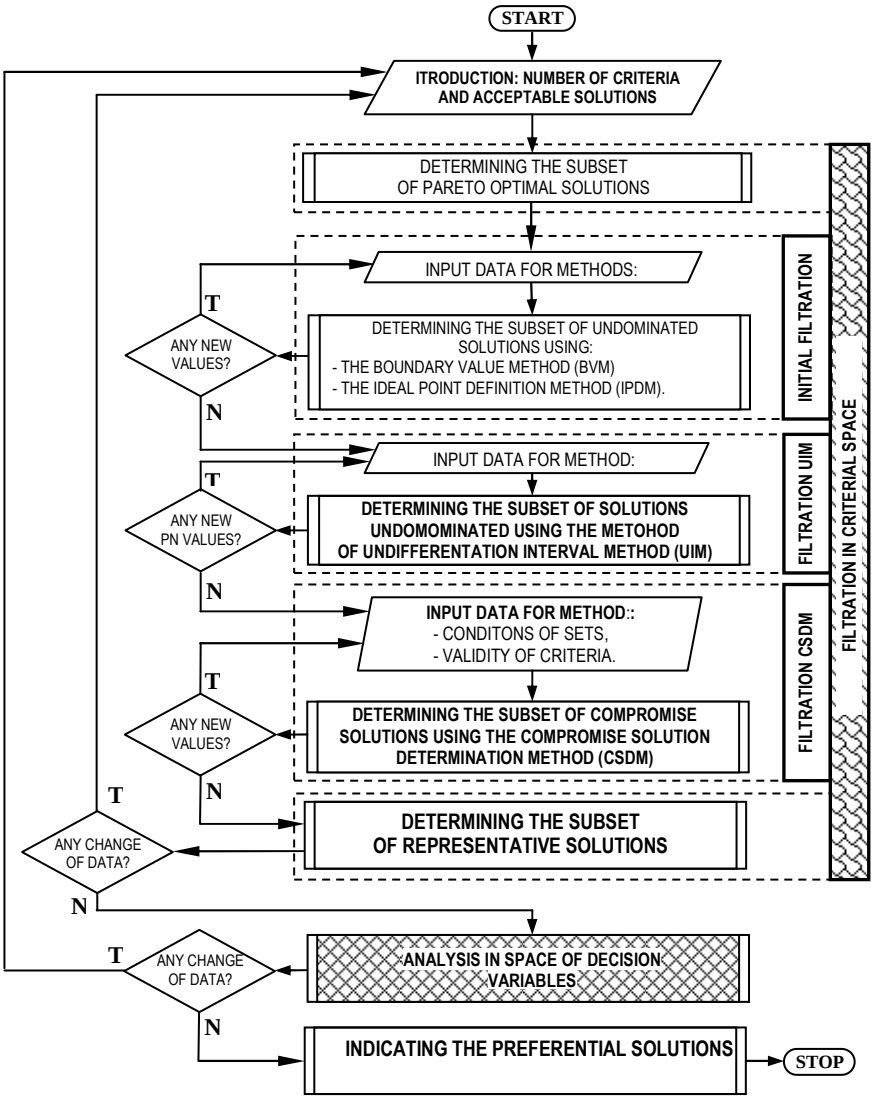


**Fig. 2. Block diagram of the Evolutionary System of Multicriteria Analysis [14]**

There is, of course, possibility of simultaneous using this two mentioned above methods of selection: BVM and IPDM. The selection of set of undominated solutions with accepted positive standard as a new ideal point $F^{o,}$ and satisfactory values $f_s$ was presented in fig 3. Using the inverse criteria in the multicriteria analysis causes that the elimination of solutions, which have very small values one component, leads simultaneously to rejecting this solutions with have big or very big values of different components.

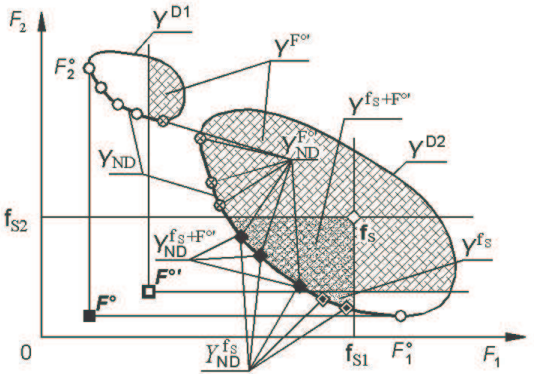


**Fig. 3. Selection of the set of undominated solutions (○) using simultaneously BVM and IPDM methods, ■ – ideal point (PI), ◘ - new PI, ⊗- valuation of the solutions which meet the new ideal point, ✱- satisfying valuation (OS), ⟐- valuation of solutions which meet the OS, ✖- valuation of solution which meet the OS and new ideal point [15].**

- ***Undifferentation Interval Method (UIM)***
  The selection using the UIM method was realized according to valuations of undominated solutions. Elimination of elements of subset uses on the idea of optimality in the sense of undifferentation interval which is based on the idea of modified mutation. The multicriteria analysis of undominated solutions is realized in the criteria space and pursue to find if the value of mutated solution ("made worse") by the accepted interval of undifferentation UI still remains as an undominated solution and will be added to actually created set of undominated solutions. In case of minimization of criteria, the element $x^{\wedge} \in \Omega$ will be undominated in the sense of undifferentation interval if and only if in the $\Omega$ set there is not an element $x^{+}$, that for each $\lambda \in N$,

$$\begin{aligned} &\textit{when} \quad F_1(x^{\wedge}) \geq 0: \quad F_l(x^{\wedge}) < F_1(x^{+}) \quad \textit{proceed} \quad (1 + \frac{PN_l}{100}) F_l(x^{\wedge}) > F_l(x^{+}) \\ &\textit{when} \quad F_1(x^{\wedge}) < 0: \quad F_l(x^{\wedge}) < F_1(x^{+}) \quad \textit{proceed} \quad (1 - \frac{PN_l}{100}) F_l(x^{\wedge}) > F_l(x^{+}) \end{aligned} \tag{4}$$

where: $\Omega$ – non-empty set of solutions optimal in Pareto sense.

The situation where the element $x^{\wedge}$ is eliminated, because after the mutation of valuation of this element about the value of selected interval of undifferentation $PN_1$, so it gets into the domination cone with the top in $F(x^{+})$ point was presented in fig . 4a. The case when both of solutions $x^{\wedge}$ and $x^{+}$ are undominated elements in the sense of undifferentation interval method are presented in fig. 4b.

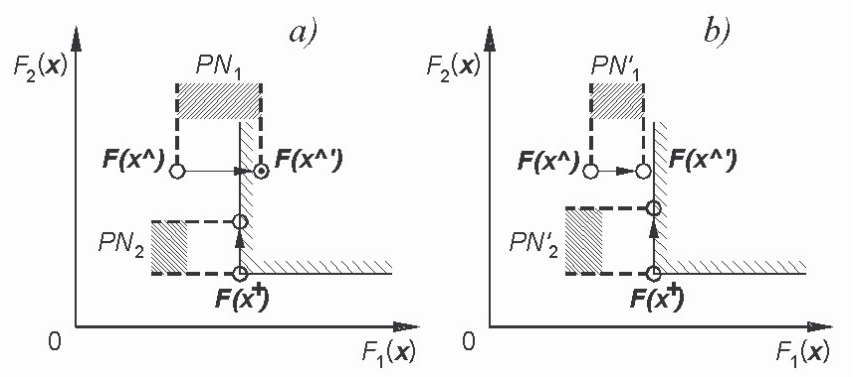


**Fig. 4. Graphic visualization of (4) condition in case of two-criteria minimization [14]**

- ***Compromise Solution Determination Method (CSDM)***
  This method tends to finding "the best solution" or subset of "the best solutions" using the analysis of domination relations in the set of vector values of indexes. In tasks of selection the decider has at his disposal complete set of acceptable solutions and theirs valuations and is not able to make new solutions. Therefore the operation of intersection applies to components of valuations of generated compromise solutions and components of the ideal point. Received in this way new ideal points, called following-up ideal points, fulfill the function of reference points during the next multicriteria analysis. The operation of intersection allows to get many reference points which are the base for generating successive compromise solutions. To visualize the way of operating the CSDM method, the situation, where the analyzed set of undominated solutions is an unseparately one and is composed of two subsets $Y^{D1}$ and $Y^{D2}$ was presented in fig 5. The subset of valuations of compromise solutions reflects the shape of analyzed set of valuations even in case if it consists of two subsets.

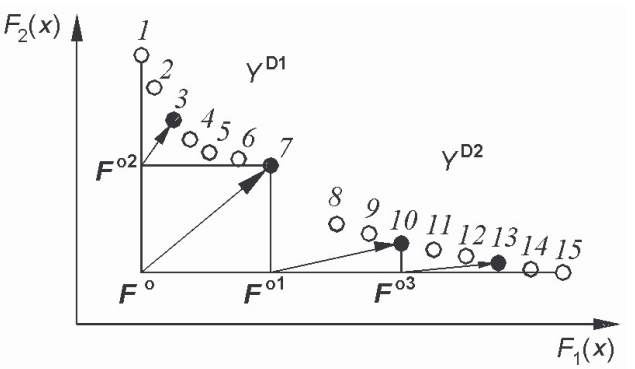


**Fig. 5. An example lay-out of valuations of compromise solutions [15].**

# 4. PROCESS OF DEFINING THE IMPORTANCE OF SOLUTION IN THE PROBLEM OF FMS MACHINE TOOLS SELECTION

Using the methodology presented in section 2, the process of machine tools selection for the task formulated in paper [10] was realized. As result of execution stages I-III the solution in form of 36 different technological paths $M=\{M_1, M_2, \ldots, M_{36}\}$ with corresponding values of target functions $F_1(M_\mu)$, $F_2(M_\mu)$ were received. The values of target functions connected with the solutions are presented in tab. 1.

**Tab. 1. Values of target functions in realized experiment of selection**

| Symbol (number) of solution | Value of target function | |
|---|---|---|
| | $F_1(M_\mu)$ [sek.] | $F_2(M_\mu)$ [zł] |
| $M_1$ | 33 482 | 3 553 054,74 |
| $M_2$ | 33 675 | 3 765 964,99 |
| $M_3$ | 33 597 | 3 548 251,65 |
| $M_4$ | 33 445 | 3 905 830,10 |
| $M_5$ | 33 712 | 3 413 189,64 |
| $M_6$ | 33 560 | 3 901 027,01 |
| $M_7$ | 33 565 | 3 535 561,80 |
| $M_8$ | 33 758 | 3 395 696,70 |
| $M_9$ | 33 680 | 3 530 758,72 |
| $M_{10}$ | 33 528 | 3 535 561,80 |
| $M_{11}$ | 33 795 | 3 395 696,70 |
| $M_{12}$ | 33 643 | 3 530 758,72 |
| $M_{13}$ | 33 638 | 3 468 319,36 |
| $M_{14}$ | 33 831 | 3 681 229,62 |
| $M_{15}$ | 33 753 | 3 463 516,28 |
| $M_{16}$ | 33 601 | 3 821 094,72 |
| $M_{17}$ | 33 868 | 3 328 454,26 |
| $M_{18}$ | 33 716 | 3 816 291,64 |

| Symbol (number) of solution | Value of target function | |
|---|---|---|
| | $F_1(M_\mu)$ [sek.] | $F_2(M_\mu)$ [zł] |
| $M_{19}$ | 33 029 | 4 306 080,63 |
| $M_{20}$ | 33 222 | 3 901 027,01 |
| $M_{21}$ | 33 144 | 3 548 251,65 |
| $M_{22}$ | 32 992 | 4 658 855,99 |
| $M_{23}$ | 33 259 | 3 548 251,65 |
| $M_{24}$ | 33 107 | 3 901 027,01 |
| $M_{25}$ | 33 112 | 4 288 587,69 |
| $M_{26}$ | 33 305 | 3 530 758,72 |
| $M_{27}$ | 33 227 | 3 530 758,72 |
| $M_{28}$ | 33 075 | 4 288 587,69 |
| $M_{29}$ | 33 342 | 3 530 758,72 |
| $M_{30}$ | 33 190 | 3 530 758,72 |
| $M_{31}$ | 33 185 | 4 221 345,26 |
| $M_{32}$ | 33 378 | 3 816 291,64 |
| $M_{33}$ | 33 300 | 3 463 516,28 |
| $M_{34}$ | 33 148 | 4 574 120,62 |
| $M_{35}$ | 33 415 | 3 463 516,28 |
| $M_{36}$ | 33 263 | 3 816 291,64 |

The lay-out of received solutions according to calculated target functions was presented in fig. 6.

A multicriteria analysis was realized using the Evolutionary System of Multicriteria Analysis according to algorithm presented in section 3 (fig. 2). In the first step the optimal in Pareto sense solutions were determined. This set contains 10 elements as follows: $M_5$, $M_8$, $M_{17}$, $M_{19}$, $M_{21}$, $M_{22}$, $M_{24}$, $M_{28}$, $M_{30}$, $M_{33}$.

In second step the selection using the Undifferentation Interval Method (UIM) was realized. There were accepted values of interval of undifferentation as follows: PN = 0% according to the criterion $F_1(M_\mu)$ and PN = 1,0% according to the criterion $F_2(M_\mu)$. Non-zero value of interval of undifferentation according to the criterion $F_2(M_\mu)$ was accepted as a result of possible inaccuracy of calculated target functions what follows from rounding and differences in rates when calculating the prices of purchasing the machine tools. As a result of realized analysis using the UIM method the received subset was limited to 7 elements. This are: $M_5$, $M_{17}$, $M_{19}$, $M_{21}$, $M_{22}$, $M_{24}$, $M_{33}$.

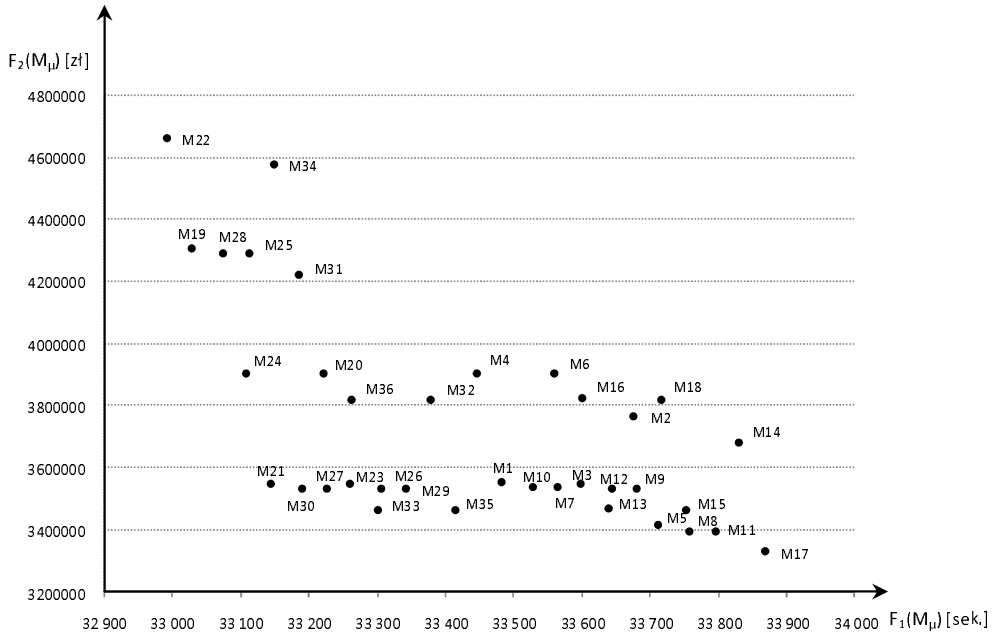


**Fig. 5. Lay-out of solutions according to calculated target function**

In third step, the filtration using the Compromise Solution Determination Method was realized. The metrics both min-max and min-max with weight with different preferences of analyzed criteria were used. The results of analyses were presented in Tab. 2. It is worth to pay attention that to find the degree of sensitiveness each of solution, the weights from 0,2 to 0,8 for each of criteria have been taken.

**Tab. 2. Results of filtration using the CSDM method**

| No. | Preferention weights $\sum\omega_l = 1$ | First compromise solution | Subset of compromise solutions |
|---|---|---|---|
| 1. | $\omega_{1 =} \omega_2 = 0{,}5$ | $M_5$ | $\mathbf{M_5}^*$, $\mathbf{M_{33}}$, $M_{21}$, |
| 2. | $\omega_1 = 0{,}6;\ \omega_2 = 0{,}4$ | $M_5$ | $\mathbf{M_5}$, $\mathbf{M_{33}}$, $M_{21}$, |
| 3. | $\omega_1 = 0{,}7;\ \omega_2 = 0{,}3$ | $M_{33}$ | $\mathbf{M_{33}}$, $\mathbf{M_5}$, $M_{21}$ |
| 4. | $\omega_1 = 0{,}8;\ \omega_2 = 0{,}2$ | $M_{33}$ | $\mathbf{M_{33}}$, $\mathbf{M_5}$, $M_{24}$ |
| 5. | $\omega_1 = 0{,}4;\ \omega_2 = 0{,}6$ | $M_{17}$ | $M_{17}$, $\mathbf{M_5}$, $\mathbf{M_{33}}$ |
| 6. | $\omega_1 = 0{,}3;\ \omega_2 = 0{,}7$ | $M_{17}$ | $M_{17}$, $\mathbf{M_5}$, $\mathbf{M_{33}}$ |
| 7. | $\omega_1 = 0{,}2;\ \omega_2 = 0{,}8$ | $M_{17}$ | $M_{17}$, $\mathbf{M_5}$, $\mathbf{M_{33}}$ |

* - preffered solution – present in each of compromise solutions' subset

In fourth step the subset of representative solutions was searched. Analysis of the results presented in tab. 2 showed that solutions $M_5$ and $M_{33}$ exists in each of determined subset of solutions, solutions $M_{17}$ and $M_{21}$ appeared three times and the $M_{24}$ solution appeared one time. Ipso facto the realized analysis in the space of decision variables showed

that received solutions $M_5$ and $M_{33}$ are characterized by the minimal sensitivity of changing the weights of particular criteria and taking into account major assumptions of Evolutionary System of Multicriteria Analysis – they are preferred solutions (with the same degree of importance). The final decision of about solution should be done by the designer taking into account particular analysis and criteria of individual preferences according to received values of target functions.

## 5. CONCLUSIONS

Decision support systems should help the designer to find the optimal solution among many possibilities for the defined decision task. It is especially highly important, when the quality of analyzed variants of solutions is described with many criteria and the decision problem is burdened with the high risk of non-objective criteria when taking the decision.

One of the more important problem in the area of modern manufacturing systems design is a question of proper machine tools (technological machines) selection. When take into account that in the process of machine tools selection the relation between objective and subjective criteria is 20 to 80 [11] and the choice should be done considering some or several frequently inverse criteria, the need of searching methods which maximize the objectivity of taken decision.

In this paper the possibility of implementation the Evolutionary System of Multicriteria Analysis <ESAW> for the defining the importance of solutions in the process of casing-class FMS machine tools selection was shown. Results of realized analysis shows that the <ESAW> system allows to find among the number of analyzed solutions few (or sometimes only one) proffered solutions from the selected criteria of evaluation point of view. Thanks to fact that the selection process is based onto internal features of solutions' set – the preferred solutions are characterized with the "immunity" for subjective criteria of decider's evaluation.

## REFERENCES


1. ARSLAN M.C., CATAY B., BUDAK E.: *A decision support system for machine tool selection,* Journal of Manufacturing Technology Management (2004), Vol. 5, pp. 101-109.
2. ATAMANI A., LASHKARI R.S: *A model of machine tool selection and operation allocation in flexible manufacturing systems,* International Journal of Production Research (1998), Vol. 36, pp. 1339-1349.
3. AYAG Z, OZDEMIR R.G.: *A fuzzy AHP approach to evaluating machine tool alternatives,* Journal of Intelligent Manufacturing (2006), Vol. 17, 179-190.
4. AYAG Z.: *A hybrid approach to machine tool selection through AHP and simulation.* International Journal of Production Research (2007), Vol. 45, No. 9, pp. 2029-2050.
5. BALAJI C.M., GURUMURTHY A., KODALI R.: *Selection of a machine tool for FMS using ELECTRE III – a case study.* Automation Science and Engineering (2009), pp. 171-176.
6. CHAN F.T.S., SWARNKAR R.: *Ant colony optimization approach to a fuzzy goal programming model for a machine tool selection and operation problem in an FMS,* Robotics and Computer-Integrated Manufacturing (2006), Vol. 22, pp. 353-362.
7. CIMREN E., CATAY B., BUDAK E.: *Development of a machine tool selection system using AHP,* Intelligent Journal of Advances Technology (2007), Vol. 35, pp. 363-376.
8. DAGDEVIREN M.: *Decision making in equipment selection: an integrated approach with AHP and PROMETHEE,* Journal of Intelligent Manufacturing (2008), Vol. 19, pp. 397-406.

9. GOLA A., ŚWIĆ A., *Brief Preliminary Design for a Method of FMS Machine Tools Subsystem Selection*, PAMM (2010) Vol. 9, Issue 1, pp. 663-664.
10. GOLA A., ŚWIĆ A., *Computer Aided FMS Machine Tools Subsystem Selection – Conception of Methodology*, [w:] Z.Banaszak, J.Matuszek, Applied Computer Science. Supporting Enterprise Management Processes (2009), Vol. 5, No 1, Wyd. ATH, Bielsko-Biała, s. 27-39.
11. HONCZARENKO J., Słaby I.: Metodyka doboru obrabiarek skrawajacych, Mechanik, Nr 3/2009, s. 166-173.
12. LIN Z.C., YANG C.B.: Evaluation of machine selection by the AHP method, Journal of Materials Processing Technology (1994), Vol. 57, pp. 253-258.
13. MISHRA S., PRAKASH, TIVARI M.K, LASHKARI R.S.: *A fuzzy goal-programming model for machine tool selection and operation allocation problem in FMS: A quick converging simulated annealing-based approach*, International Journal of Production Research (2006), Vol. 44. No. 1, pp. 43-76.
14. MONTUSIEWICZ J., *Ewolucyjna analiza wielokryterialna w zagadnieniach technicznych*, Prace IPPT Polskiej Akademii Nauk, Warszawa 2004.
15. MONTUSIEWICZ J.: *Komputerowe wspomaganie decyzji przy użyciu Ewolucyjnego Systemu Analizy Wielokryterialnej*, Przegląd Mechaniczny nr 5'07, Suplement, s. 107-110.
16. ONUT S., KARA S.S., EFENDIGIL T.: *A hybrid fuzzy MCDM approach to machine tool selection*, Journal of Intelligent Manufacturing (2008), Vol. 19, pp. 443-453.
17. PRIMOSE P.L., LEONARD R., *Selecting Technology for Investment in Flexible Manufacturing*, International Journal of Flexible Manufacturing Systems (1991), Vol. 4, pp. 51-77.
18. RAO R.V., PARNICHKUN M.: *Flexible manufacturing system selection using a combinatorial mathematics-based decision-making method*, International Journal of Production Research, Vol. 47, Issue 24, pp. 6981-6998.
19. ŚWIĆ A. GOLA A., *Elements of Design of Production Systems – Methodology of Machine Tool Selection in Casing-Class FMS*, Management and Production Engineering Review (2010), Vol. 1, No. 2, pp. 73-81.
20. TABUCANON M.T., BATANOV D.N., VERMA D.K.: *Intelligent Decision Support System (DSS) for the selection process of alternative machines for Flexible Manufacturing Systems (FMS)*, Computers in Industry (1994), Vol. 25, pp. 131-143.
21. TOLIO T. (red.): Design of Flexible Production Systems. Methodologies and Tools, Springer-Verlag, Berlin Heidelberg 2009.
22. WANG T.Y., SHAW C.F., CHEN Y.L.: *Machine selection in flexible manufacturing cell: A fuzzy multiple attribute decision making approach*, International Journal of Production Research (2000), Vol. 30, pp. 2079-2097.
23. YOURDAKUL M.: *AHP as a strategic decision making tool to justify machine tool selection*, Journal of Materials Processing Technology (2004), Vol. 146, pp. 365-376.
24. ZAWADZKA L., *Modele optymalizacji wielokryterialnej. Przykłady aplikacji*, [w:] L. Zawadzka, *Inżynieria systemów zarządzania*, Wyd. Politechniki Gdańskiej, Gdańsk 2002.

Juraj SPALEK[*] Michal GREGOR[**]

# ADAPTIVE SWITCHING OF MUTATION RATE FOR GENETIC ALGORITHMS AND GENETIC PROGRAMMING

**Abstract**

*The paper concerns the application of Genetic Algorithms and Genetic Programming to complex tasks such as automated design of control systems, where the space of solutions is non-trivial and may contain discontinuities. An adaptive value-switching mechanism for mutation rate control is proposed. It is shown that the proposed mechanism is useful in preventing the search from getting trapped in local extremes of the fitness landscape.*

## INTRODUCTION

Genetic Algorithms represent a well-known optimization method recognized in particular for its flexibility in representation of solutions and for its ability to produce reasonably fit results in a reasonable amount of time. Genetic Programming applies the theory of Genetic Algorithms to evolving computer programs, usually represented by syntactic trees.

There is a multitude of research papers that aim to improve convergence and robustness of both methods. Some of these concentrate on parameter control, that is to say on setting and modifying various parameters of the algorithm.

This paper presents an adaptive value-switching mechanism for control of the mutation rate, which aims to decrease the probability that the search will become trapped in local maxima by increasing mutation probability to a high value once such scenario is detected.

## GENETIC ALGORITHMS AND GENETIC PROGRAMMING

Although the methods in question are relatively well known, let us first present some fundamental information about both – Genetic Algorithms (GA) and Genetic Programming (GP).

[*] Prof. Ing. Juraj Spalek, PhD. – Department of Control and Information Systems, Faculty of Electrical Engineering, University of Žilina, Univerzitná 1, 010 26 Žilina, Slovak Republic, juraj.spalek@fel.uniza.sk

[**] Bc. Michal Gregor – Department of Control and Information Systems, Faculty of Electrical Engineering, University of Žilina, Univerzitná 1, 010 26 Žilina, Slovak Republic, o.m.gregor@gmail.com

Genetic algorithms represent one of the several computational techniques based on simulation of evolution, a process based on the principle of *natural selection*, that is, on the *survival of the fittest*. The genetic algorithm operates on a population of individuals.

The individuals represent various solutions of a specific problem. The main principle of the algorithm is as shown in figure 1.

The first step is to generate the initial population – this typically involves generating a group of random individuals. The next step is to perform evaluation of those individuals, which enables the algorithm to compare the individuals to each other and, furthermore, to introduce the survival of the fittest: the individuals with the best scores (also known as *fitness* in the genetic algorithm terminology) are the most likely* to participate in *reproduction*, that is, in forming the next generation. This is analogous to the natural selection process, in which the fitter individuals have greater chance to survive and to reproduce.

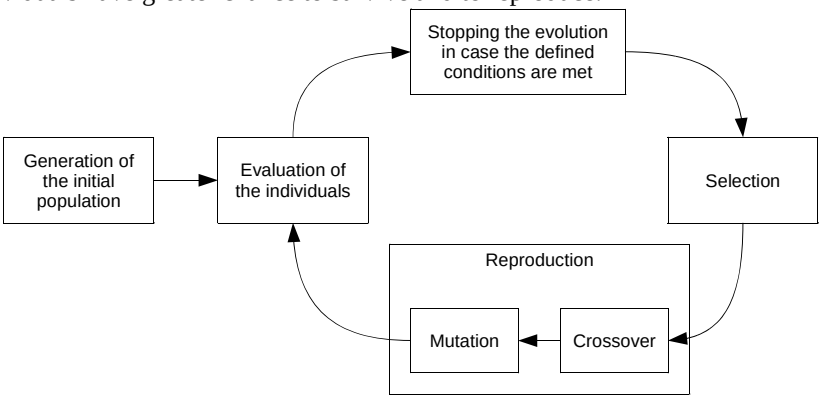


**Fig. 1. The general principle of genetic algorithms**

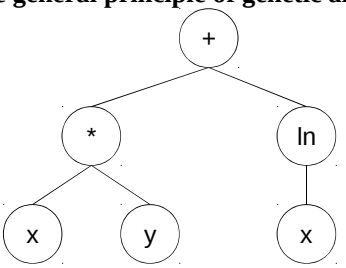


**Fig. 1. A simple example of a syntactic**

Genetic programming (GP) is a technique developed by John Koza (see *Genetic Programming: On the Programming of Computers by Means of Natural Selection* [1]). It applies the theory of Genetic Programming to the task of evolving computer programs. The main idea of Genetic Programming is the way in which the individuals are represented – by syntactic trees (also known as parse trees). Fig. 2 shows a simple example of a syntactic tree that codes the expression $x.y + \ln x$.

* However, we usually refrain from directly choosing the best *n* individuals as that tends to reduce diversity, which leads to getting trapped in a local extreme.

For syntactic trees crossover is usually done by swapping 2 randomly selected sub-trees of the 2 parent individuals, while mutation may be implemented by replacing a randomly selected sub-tree by a newly generated one. For a more detailed introduction to the problem refer to [1] or [2].

## THE ARTIFICIAL ANT PROBLEM

The artificial ant problem described by John Koza in [1] is essentially a trail-following task. The actor – an artificial ant – is supposed to navigate in an environment following an irregular path consisting of pieces of food which it collects. The ant has very limited sensing capabilities – it only sees a single tile right in front of it. John Koza successfully solves the problem by applying Genetic Programming†.

In our work we have set some additional requirements concerning the form of the solution – the evolved controller should, when executed, return the action that the ant is to execute next instead of calling functors that directly execute the action and wait for its completion. The set of terminals contains persistent variables and the controller has access to a pre-set number of its previous inputs and outputs.

Controllers based on such mode of execution seem to be much more difficult to evolve than those originally proposed by Koza. The search usually gets trapped in a local maximum from which it is often unable escape.

## EXISTING APPROACHES TO PARAMETER CONTROL

In some applications based on the theory of genetic algorithms, the optimization task may be so difficult – with a complex space including a great number of local optima in which the search process can get trapped – that additional techniques may be required to find the global optimum. Genetic programming does in a multitude of tasks serve as an especially good example of the problem, as it evolves computer programs and it is obvious that two very similar computer programs may produce drastically different results and thus the space of solutions is highly complex.

Among the approaches that aim to prevent getting trapped in a local optimum are adaptive schemes that observe various parameters of the algorithm or the search process itself and using the observed values adapt some of the parameters. The approaches to parameter setting can basically be divided into the following categories [3], [4]:

- static parameter control,
- dynamic parameter control,
- adaptive parameter control,
- self-adaptive parameter control.

### Static Parameter Control

The common feature of approaches falling into this category is that the setting they provide remains constant for the entire duration of the evolutionary process. There are many works analysing the problem of finding optimum settings for parameters like mutation probability and

† See [1] for detailed information about the solution.

crossover probability. Some of these are listed in [3], e.g. the work of Mühlenbein, which proposes the following formula for the mutation probability:

$$p_m = 1/L\,, \tag{1}$$

where $L$ is the length of the bit string by which the individual is represented.

**Dynamic Parameter Control**

As stated in [4] dynamic parameter approaches typically prescribe a deterministically decreasing schedule over a number of generations and provides a formula for mutation probability derived by Fogarty:

$$p_m(t) = \frac{1}{240} + \frac{0.11375}{2^t}\,, \tag{2}$$

where $t$ is the generation counter.

Articles [3], [4] both refer to a more general expression derived by Hesser and Männer:

$$p_m(t) = \sqrt{\frac{\alpha}{\beta}} \times \frac{\exp\left(\frac{-\lambda t}{2}\right)}{\lambda\sqrt{L}}\,, \tag{3}$$

where $\alpha$, $\beta$, $\gamma$ are constants, $\lambda$ is the population size and $t$ is the generation counter and $L$ is again the length of the bit string.

**Adaptive Parameter Control**

Adaptive parameter control techniques monitor the search process itself and provide feedback. Some examples can be found in [5]. The authors propose the following formulas for crossover and mutation probability respectively:

$$p_c = \begin{cases} k_1 \dfrac{f_{max} - f'}{f_{max} - \bar{f}} & f' > \bar{f} \\ \\ k_3 & f' \le \bar{f} \end{cases} \tag{4}$$

$$p_m = \begin{cases} k_2 \dfrac{f_{max} - f}{f_{max} - \bar{f}} & f > \bar{f} \\ \\ k_4 & f \leq \bar{f} \end{cases} \tag{5}$$

where $f$ is the fitness value of the individual to be mutated, $f'$ is the larger of the fitness values of the individuals to be crossed and $k_3$ and $k_4$ are constants. It is required that $k_1$ and $k_2$ be less than 1.0 in order to constrain $p_c$ and $p_m$ to the range of $\langle 0,1 \rangle$. The $p_c = k_3 \quad f' \leq \bar{f}$ and $p_m = k_4 \quad f \leq \bar{f}$ expressions are to prevent crossover and mutation probabilities from exceeding 1.0 for suboptimal solutions.

Authors of [5] also observe that $p_c$ and $p_m$ are zero for the solution with maximum fitness and that $p_c = k_1$ for $f' = \bar{f}$, while $p_m = k_2$ for $f = \bar{f}$. For further details and for information concerning setting the values of the constants refer to [5].

### Self-adaptive Parameter Control

When using the self-adaptive parameter control approach, parameters such as mutation rate and crossover probability of each individual are part of its genome and are evolved with it. As stated in [4], the idea behind this is that a good parameter value will provide an evolutionary advantage to the individual. For further reference see [3] or [4].

## ADAPTIVE VALUE-SWITCHING OF MUTATION RATE

### Motivation

Most of the existing parameter setting mechanisms, as presented in the previous chapter, either focus on setting GA-specific parameters such as length of the bit string (e.g. rule (1)), or are not adaptive (e.g. (2) and (3)). The adaptive mechanism described in [5] (formulas (4) and (5)) seems more fit to the task because it implements certain form of convergence detection based on comparison of the maximum and average fitness values. However this approach does little to solve the problem of getting trapped in a local optimum as the method does not discern between local and global optima.

Furthermore – as mentioned hereinbefore – equations (2) and (3) assign the best individual zero crossover and mutation probabilities, while assigning high probabilities to less fit individuals. The reasoning behind this is that the less fit individuals can safely be disrupted by high mutation rates and recombined by crossover (thus employing the solutions with subaverage fitness to search the space [5]), while the highly fit individuals should be preserved.

However, such approach has a very obvious downside which the authors do not seem to address – the highly fit individuals obviously contain the most excellent genetic material available and by disallowing mutation and crossover for these individuals the genetic code they carry becomes isolated and is not used to generate new solutions.

**Description of the Proposed Adaptive Mechanism**

The idea that the most fit solutions should survive crossover and mutation unmodified is valid, yet that feature can be enforced by using elitism[‡]. Keeping that in mind we propose a different adaptation scheme in order to address the other issues. The main idea is that the mutation probability should be increased to a high value when the search has become trapped in an extreme so as to provide the search process with new genetic material some of which may previously have been unavailable. To determine whether the search has become trapped the adaptive mechanism observes the change of average fitness in time.

To describe the solution in more detail – the algorithm works with 2 values of mutation probability – the normal value and the high value. The algorithm switches from the normal value to the high value once the trigger criterion activates.

The trigger criterion itself is based on a measure that we will herein term a *delta sum*:

$$\Delta S_i = \alpha.\Delta S_{i-1} + \frac{\bar{f}_i - \bar{f}_{i-1}}{\bar{f}_i}, \tag{6}$$

where $\Delta S_i$ is the delta sum in generation $i$ and $\bar{f}_i$ is the average fitness in generation $i$ and $\alpha$ is the feedback coefficient (the experiments have been carried out for $\alpha = 0.4$).

If the delta sum is lower than a pre-set value for a predefined number of generations, that is to say the increase of average fitness in the last few generations is low, indicating that the search has become trapped[§] – the mutation probability is set to its high value so as to provide the search with new genetic material. As mentioned before, when used in conjunction with elitism it is guaranteed that the best solution is not destroyed by the high mutation probability.

The mutation probability is reset back to its normal value when at least one of the following conditions is true:

- the average fitness increases enough to produce a sufficiently large delta sum;
- the maximum fitness increases;
- mutation has been set to its high value for at least $n$ generations.

The *n*-generation limit is to ensure that the activation does not go on indefinitely (with the high mutation probability it is not very likely that the average fitness will increase enough to satisfy the first condition and maximum fitness may not increase as well).

It has been observed that average fitness typically decreases when the criterion activates because the search process is to a large extent disrupted by the high mutation probability. However after the *n*-generation limit forces the mutation rate back to its normal value, average fitness tends to increase rapidly, thus usually moving away from the local extreme.

**Experimental Results**

Several experiments have been carried out – Fig. 3 shows performance of the search algorithm with the AGA adaptive mechanism proposed in [5] with constants set according to

‡ The best individual is copied to the next generation unmodified.

§ This may also indicate convergence to the global maximum, it is, however, hardly possible to tell global and local maxima apart unless the algorithm is provided with additional problem-specific data.

recommendations. It also shows performance of the search algorithm without any adaptive mechanism and with the adaptive mechanism proposed in this paper. The maximum fitness value achieved is shown for each of the 5 runs displayed.

As shown, search achieves suboptimal results when running with no adaptive mechanism. This can be ascribed to its inability to escape from local extremes. With no adaptive mechanism the search has not found the global optimum (fitness = 89) in any of the 5 runs.

As expected, the AGA mechanism has caused further deterioration and its results are even worse than those produced in the previous case.

The Value-switching adaptive mechanism proposed in this work improves the process of search – in 2 of the runs the global optimum is found, yet in certain cases not even the high mutation rate is guaranteed to help the search escape from the local maximum (runs 2, 3, 4).

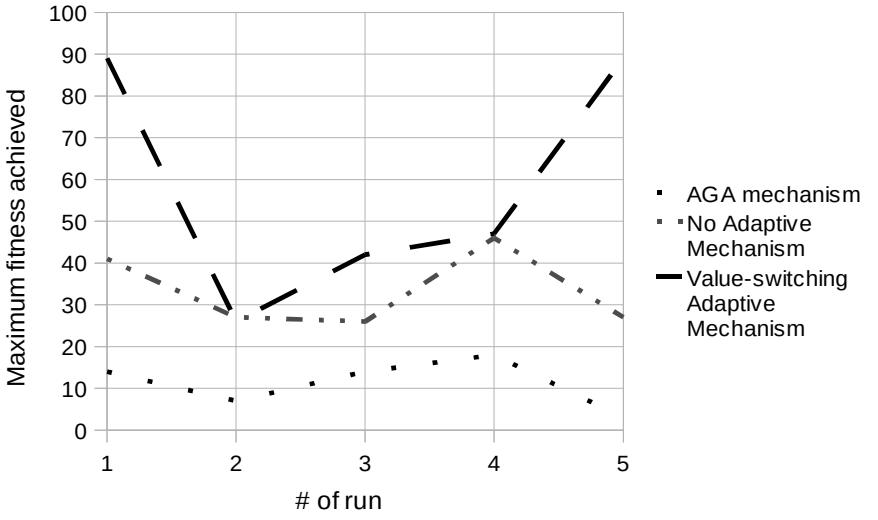


**Fig. 2. Comparison of the AGA Adaptive Mechanism and the Value-switching Adaptive Mechanism**

**Suggestions for Further Work**

It has been shown that the adaptive mechanism described in this work is able to effect considerable improvements and that it is able to some extent prevent getting trapped in local maxima. Further experiments should now be carried out in order to ascertain that the principle is valid for a wider range of tasks.

It has also become apparent that even with the high mutation rates it is not always guaranteed that the search will indeed escape from the local maximum. Value-switching, or piecewise continuous relationships for other parameters could perhaps help to alleviate the problem – this issue requires further investigation.

## CONCLUSION

It is well known that search processes based on genetic algorithms and genetic programming are prone to getting trapped in local maxima when exploring highly complex spaces.

As shown in the paper, search process based on the standard genetic programming approach fails to find the global optimum when applied to the modified version of the artificial ant problem.

This paper investigates the problem and proposes an adaptive mechanism for mutation rate control, which should help the search to escape from local extremes. As shown, the results are considerably better than those of the standard genetic programming approach.

Although the results are significantly better, even the adaptive value-switching of mutation rate as here proposed cannot always guarantee that the process will escape from a local maximum. It is possible that value-switching, or piecewise continuous relationships for other parameters could help to alleviate the problem. Such approaches could provide area for further research.

*This paper is a part of a more comprehensive research supported by: ASFEU 26220220049.*

## REFERENCES

1. KOZA J. R.: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. The MIT Press. Cambridge, Massachusetts, 1998. ISBN 0-262-11170-5
2. HYNEK J.: *Genetické algoritmy a genetické programování*. Grada Publishing, a. s. Praha, 2008. ISBN 978-80-7300-218-3
3. EIBEN Á. E., ROBERT, H., MICHALEWICZ, Z.: *Parameter Control in Evolutionary Algorithms*. IEEE Transactions of Evolutionary Computation: 3, 1999. http://www.gpa.etsmtl.ca/cours/sys843/pdf/Eiben1999.pdf
4. THIERENS, D.: *Adaptive mutation rate control schemes in genetic algorithms*. Proceedings of the 2002 IEEE World Congress on Computational Intelligence: Congress on Evolutionary Computation, 2002. http://dynamics.org/~altenber/UH_ICS/EC_REFS/GP_REFS/CEC/2002/GP_WCCI_2002/7315.PDF
5. SRINIVAS, M., PATNAIK, L. M.: *Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms*. IEEE Transactions on Systems, Man and Cybernetics: 24, 1994. http://eprints.iisc.ernet.in/archive/00006971/02/adaptive.pdf

Juraj SPALEK[*]
Michal GREGOR[**]

# Adaptive Approaches to Parameter Control in Genetic Algorithms and Genetic Programming

**Abstract**

*The paper concerns the application of Genetic Algorithms and Genetic Programming to complex tasks such as automated design of control systems, where the space of solutions is non-trivial and may contain discontinuities. Several adaptive mechanisms for control of the search algorithm's parameters are proposed, investigated and compared to each other. It is shown that the proposed mechanisms are useful in preventing the search from getting trapped in local extremes of the fitness landscape.*

## Introduction

Genetic Algorithms represent a well-known optimization method recognized in particular for its flexibility in representation of solutions. Genetic Programming applies the theory of Genetic Algorithms to evolving computer programs, usually represented by syntactic trees.
There is a multitude of research papers that aim to improve convergence and robustness of both algorithms. Some of these concentrate on parameter control, that is to say on setting and modifying various parameters of the search algorithm.
This paper proposes several adaptive mechanisms, which aim to decrease the probability that the search will become trapped in local maxima by various techniques. They are all based on detecting that the search has become trapped by observing how average fitness of the population changes in time.

## Genetic Algorithms

Genetic algorithms represent one of the several computational techniques based on simulation of evolution, a process based on the principle of *natural selection*, that is, on the *survival of the fittest*. The genetic algorithm operates on a population of individuals. The individuals represent various solutions of a specific problem. The main principle of the algorithm is as shown in figure 1.

* Prof. Ing. Juraj Spalek, PhD. – Department of Control and Information Systems, Faculty of Electrical Engineering, University of Žilina, Univerzitná 1, 010 26 Žilina, Slovak Republic, juraj.spalek@fel.uniza.sk
** Bc. Michal Gregor – Department of Control and Information Systems, Faculty of Electrical Engineering, University of Žilina, Univerzitná 1, 010 26 Žilina, Slovak Republic, o.m.gregor@gmail.com

The first step is to generate the initial population – this typically involves generating a group of random individuals. The next step is to perform evaluation of those individuals, which enables the algorithm to compare the individuals to each other and, furthermore, to introduce the survival of the fittest: the individuals with the best scores (also known as *fitness* in the GA terminology) are the most likely** to participate in *reproduction*, that is, in forming the next generation. This is analogous to the natural selection process, in which the fitter individuals have greater chance to survive and reproduce.

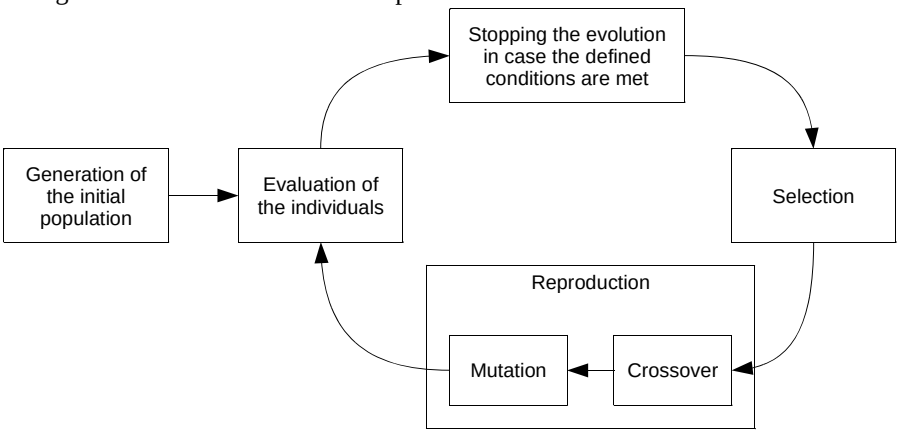


**Fig. 3. The general principle of genetic algorithms**

Figure 1 also shows that the process of forming the next generation typically involves two main genetic operators – crossover and mutation. Mutation represents a random modification of the genetic code of a single individual.
In crossover, however, several (usually two) individuals exchange parts of their genome. Therefore, if we choose mostly the highly fit individuals for reproduction, crossover provides a mechanism which may produce an offspring that combines their good properties (and thus achieves greater fitness that any of the parents).
The process of evolution runs iteratively until certain conditions are met (like achieving a predefined level of (maximum or average) fitness, or reaching the maximum number of generations††).
The individual phases will not be covered in detail here, see [1], [2], or [3]. However, the next section will present some information concerning fitness scaling as this concept will be utilized in the following sections.

**Fitness Scaling**

There is a well known problem associated with the fitness-proportionate selection methods. As [3] says, when the evolution starts, the fitness variance in population is usually high and a small number of individuals are much fitter than the others. Those individuals are consequently

** However, we usually refrain from directly choosing the best *n* individuals as that tends to reduce diversity, which leads to premature convergence and to getting trapped in a local extreme.
†† The latter is usually monitored in every implementation so as to prevent an infinite loop in case the algorithm does not converge.

much more likely to be selected than any of the others and so their offspring quickly multiplies, which leads to premature convergence and non-optimal results.
On the other hand, later in the search, when all individuals are very similar and the fitness variance is therefore low, the evolution becomes extremely slow as there are virtually no fitness differences to explore.
To address these problems a fitness scaling function can be applied – that is, the original fitness function $f$ will be wrapped into a scaling function $f_s$:

$$f_s : F \to F \,. \tag{7}$$

The scaling function wraps the original fitness function and the selection algorithm uses the scaled values:

$$Scaled\ \ fitness = f_s(f(x)), \tag{8}$$

where $x \in I$ represents an individual.
There are several widely used types of fitness scaling functions – [4] lists 3 basic categories:

1. linear,
2. sigma truncation,
3. power law.

**Linear Scaling**

A fitness function with linear scaling then has the following definition [4]:

$$f_{linear}(x) = a + b.f(x), \tag{9}$$

where $f(x)$ is the raw fitness and $a$, $b$ are user-defined constants – article [4] experiments with $a = max\{f(x)\}$ and $b = -min\{f(x)\}/N$, where $N$ is the number of individuals. In [5] author presents a way to derive relationships for $a$, $b$, which provide linear scaling that preserves the average fitness.

**Sigma Truncation Scaling**

For a fitness function with sigma truncation scaling, source [6] provides the following definition:

$$f_{sigma}(x) = 1 + \frac{f(x) - \mu_f}{\sigma_f}, \tag{10}$$

where $\mu_f$ and $\sigma_f$ are the mean fitness and the standard deviation – respectively – of fitness for the current generation.

**Power Law Scaling**

Source [5] provides the following definition of fitness function scaled using the power law scaling:

$$f_{power}(x) = f(x)^k , \tag{11}$$

where $k$ is a problem-dependent exponent that may require to be changed during the run. [5] also states that a value of $k = 1.005$ has been successfully used in machine-vision applications.

**Boltzmann Scaling**

There are also several special scaling methods, such as the Boltzmann scaling [6], definition of which is as follows:

$$f_{Boltzmann}(x) = \frac{\exp(f(x)/T)}{mean[\exp(f(x)/T)]}, \tag{12}$$

where $T$ represents a *temperature* parameter, which gradually reduces over time (with an increasing number of generations).

**Scaling the Fitness Function to Satisfy the Requirements**

Certain selection methods also impose requirements on the range of the fitness function, the most obvious example being the fitness roulette selection, where fitness values must be greater than or equal to zero (see (11)). The most apparent way to achieve this is to use the following scaling, which could be considered a special case of linear scaling:

$$f_s(x) = \begin{cases} f(x) - min\{f(x)\} & min\{f(x)\} < 0 \\ f(x) & min\{f(x)\} \geq 0 \end{cases} \tag{13}$$

The minimum can be evaluated over the current generation, or over the current and $n$ previous generations in which case the subtraction of the minimum is referred to as *fitness windowing* [7].

**Adaptive Genetic Algorithms**

In some applications based on the theory of genetic algorithms, the optimization task may be so difficult – with a complex space including a great number of local optima in which the search process can be get trapped – that additional techniques may be required to find the global optimum. Genetic programming presented in the next section does in a multitude of tasks serve as an especially good example of the problem, as it evolves computer programs and it is obvious that two very similar computer programs may produce drastically different results and thus the space of solutions is highly complex..

Among the approaches that aim to prevent getting trapped in a local optimum are the adaptive schemes that observe various parameters of the algorithm or the search process itself and using the observed values adapt some of the parameters. The approaches to parameter setting can basically be divided into the following categories [8], [9]:

4. static parameter control,
5. dynamic parameter control,
6. adaptive parameter control,
7. self-adaptive parameter control.

**Static Parameter Control**

The common feature of approaches falling into this category is that the setting they provide remains constant for the entire duration of the evolutionary process. There are many works analysing the problem of finding optimum settings for parameters like mutation probability and crossover probability. Some of these are listed in [8], e.g. the work of Mühlenbeinm which proposes the following formula for the mutation probability:

$$p_m = 1/L\,, \tag{14}$$

where $L$ is the length of the bit string.

**Dynamic Parameter Control**

As stated in [9] dynamic parameter approaches typically prescribe a deterministically decreasing schedule over a number of generations and provides a formula for mutation probability derived by Fogarty:

$$p_m(t) = \frac{1}{240} + \frac{0.11375}{2^t}\,, \tag{15}$$

where $t$ is the generation counter.
Papers [8] and [9] both refer to a more general expression derived by Hesser and Männer:

$$p_m(t) = \sqrt{\frac{\alpha}{\beta}} \times \frac{\exp\left(\frac{-\lambda t}{2}\right)}{\lambda\sqrt{L}}\,, \tag{16}$$

where $\alpha$, $\beta$, $\gamma$ are constants, $\lambda$ is the population size and $t$ is the generation counter and $L$ is again the length of the bit string.

**Adaptive Parameter Control**

Adaptive parameter control techniques monitor the search process itself and provide feedback. Some examples can be found in [10], which starts with a simple expression for the mutation and crossover probabilities. Crossover probability is expressed as follows:

$$p_c = \frac{k_1}{f_{max} - \bar{f}}, \tag{17}$$

where $k_1$ is a constant and $f_{max}$, $\bar{f}$ are the current generation maximum and average fitness values respectively.
A similar formula is proposed for mutation probability:

$$p_m = \frac{k_2}{f_{max} - \bar{f}}, \tag{18}$$

where $k_2$ is a constant.
It is further concluded in [10] that these expression do not depend on the fitness value of any particular solution, which means that the crossover and mutation probabilities will be the same for both – individuals with low and high fitness values. Another version of these formulas is derived that reflects these concerns [10]:

$$p_c = \begin{cases} k_1 \dfrac{f_{max} - f'}{f_{max} - \bar{f}} & f' > \bar{f} \\ k_3 & f' \le \bar{f} \end{cases} \tag{19}$$

$$p_m = \begin{cases} k_2 \dfrac{f_{max} - f}{f_{max} - \bar{f}} & f > \bar{f} \\ k_4 & f \le \bar{f} \end{cases} \tag{20}$$

where $f$ is the fitness value of the individual to be mutated, $f'$ is the larger of the fitness values of the individuals to be crossed and $k_3$ and $k_4$ are constants. It is required that $k_1$ and $k_2$ be less than 1.0 in order to constrain $p_c$ and $p_m$ to the range of $\langle 0,1 \rangle$. The $p_c = k_3 \quad f' \le \bar{f}$ and $p_m = k_4 \quad f \le \bar{f}$ expressions are to prevent crossover and mutation probabilities from exceeding 1.0 for suboptimal solutions.
Authors of [10] also observe that $p_c$ and $p_m$ are zero for the solution with maximum fitness and that $p_c = k_1$ for $f' = \bar{f}$, while $p_m = k_2$ for $f = \bar{f}$. For further details and for information concerning setting the values of the constants refer to [10]. Some discussion concerning this approach is also provided in section 0.

### Self-adaptive Parameter Control

When using the self-adaptive parameter control approach, parameters such as mutation rate and crossover probability of each individual are part of its genome and are evolved with it. As stated in [9], the idea behind this is that a good parameter value will provide an evolutionary advantage to the individual. For further reference see [8] or [9].

## Genetic Programming

Genetic programming (GP) is a technique introduced by John Koza (see *Genetic Programming: On the Programming of Computers by Means of Natural Selection* [11]). It utilizes the previously outlined concepts to evolve computer programs. The main idea of Genetic Programming revolves around the way in which the individuals are represented, that is to say around the syntactic trees (also known as parse trees). The problem will be analysed more specifically in the following sections.

### Representation

It is obvious, that simple text-based representation of a programme is not especially suitable for genetic algorithms as using a naïve implementation of crossover and mutation over the text-based code would lead to syntactically incorrect programs.
The solution proposed by John Koza is to represent a program using a parse tree (see Fig. 2 and 3 for an instance), which is analogous to LISP S-expressions [1]. The syntactic tree is a graph with two types of nodes – non-terminals, which represent functions, and terminals, which represent variables and constants.
Figures 2 and 3 show examples of such trees with Fig. 2 displaying a tree that codes the expression $x.y + \ln x$ and Fig. 3 displaying a tree with more general mechanisms like conditional execution, assignment and return.

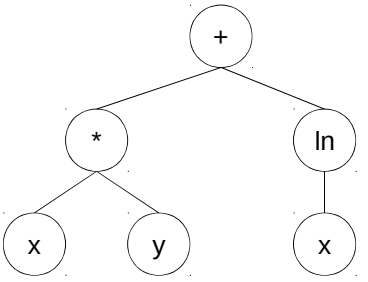


**Fig. 4. A simple example of a syntactic tree**

The program in Fig. 3 shows one of the possible ways to return values. The root node called PRG (the name is taken over from [1], where a PRG functor is used to express that several void-returning functors are called in a sequence) is a functor with an arbitrary number of inputs of type *void*, while the last input is of a pre-set type, which is identical to the return type of the program. That way after all processing is done by the void input subtrees, the result can be collected using the last input and returned.

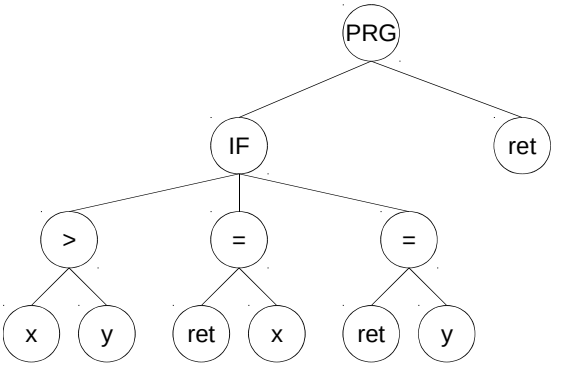


**Fig. 5 A more complex parse tree**

The program from figure 3 can be rewritten into the following C++ code (Listing 1):

*Listing 1 Code expressed by Fig. 3*

```
1. if(x > y) ret = x;
2. else ret = y;
3. return ret;
```

The representation proposed by Koza has one important property, known as the *closure property*, which requires that any valid tree generated from a set of terminals:

$$T = \{t_1, t_2, ..., t_n\}, \tag{21}$$

and a set of non-terminals:

$$NT = \{t_1, t_2, ..., t_m\}, \tag{22}$$

represents a valid program, which states that any non-terminal should be able to handle as an argument any data type and value returned from a terminal of non-terminal [12].
In contrast to this approach, several researches focus on the so-called *strongly typed genetic programming* [12], where nodes are allowed to have different incompatible return and argument types. In this case, type constraints have to be enforced, which introduces several fundamental differences. The most notable aspect is that when generating, crossing or mutating a tree care has to be taken to ensure that the return type of the node used as an input is compatible with the data type of the input itself.
The closure property can still be enforced in strongly typed genetic programming using dynamic typing. Non-terminals can be built so that they accept an argument of any type, but throw an exception if type id of the argument is not as expected.

## The Artificial Ant Problem

The artificial ant problem described by John Koza in [11] is essentially a trail-following task. The actor – an artificial ant – is supposed to navigate in an environment following an irregular path consisting of pieces of food which it collects. The ant has very limited sensing capabilities – it only sees a single tile right in front of it. John Koza successfully solves the problem by applying Genetic Programming‡‡.

This constraint, although a reasonable one – with many line-following agents this is in fact the case – makes the task of navigating along a non-trivial path rather difficult. It seems that even a human is generally unable to navigate the ant correctly when only seeing a single tile in front of the actor although this has not been tested on a wide range of subjects.

Concerning the application of GP to the problem, Koza uses the following set of terminals [11]:

$$T = \{MOVE,\quad RIGHT,\quad LEFT\}, \tag{23}$$

and the following non-terminals:

$$F = \{IF\text{ - }FOOD\text{ - }AHEAD,\quad PROGN2,\quad PROGN3\}. \tag{24}$$

The meaning of most of these is straight-forward – MOVE moves the actor forward by a single step, RIGHT and LEFT turn the actor in the respective directions. IF-FOOD-AHEAD is a functor with two arguments – the first is the then part and is executed if there is a piece of food in front of the actor, while the other is the else part. PROGN2 and PROGN3 are functors with 2 and 3 arguments respectively. PROGN represents a sequence of steps to be executed unconditionally, that is, PROGN2 and PROGN3 both execute each of its sub-trees.

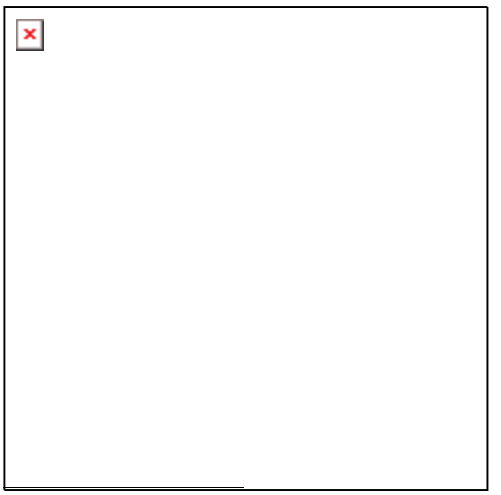

**Fig. 6. The Santa Fe trail**

‡‡ See [11] for detailed information about the solution.

The evaluation is based on simulation and the fitness is equal to the amount of food collected by the actor. It would normally be necessary to run several simulations for every individual to make sure that the solution works in general and not only on the single path on which it was tested. To avoid this Koza uses a trail known as the Santa Fe trail[§§] (Fig. 4), which is presumed to be sufficiently representative of the general trail following problem [11].

**Mode of Execution and Operators Used**

It is also necessary to mention the mode of execution used by Koza – the program generated by the evolutionary search is executed as fully as possible and then re-executed [11]. Both [11] and [1] limit the number of steps that a solution is allowed to perform to 400 so as to prevent running indefinitely for unfit individuals. The population size is set to 500 individuals and the maximum number of generations to 50 for both [11] and [1].
In our work we have set some additional requirements concerning the form of the solution – the evolved controller should, when executed, return the action that the ant is to execute next instead of calling functors that directly execute the action and wait for its completion. The set of terminals contains persistent variables and the controller has access to a pre-set number of its previous inputs and outputs.
Controllers based on such mode of execution seem to be much more difficult to evolve than those originally proposed by Koza. The search usually gets trapped in a local maximum from which it is often unable escape.
Let us provide the reader with some brief information concerning the terminals and non-terminals used in our work. The following components were utilized:

1. *VariableFunctor<NavAction>* – a terminal that acts as a variable of type NavAction (NavAction is an enumerated type representing the action that an actor can take like stay, forward, turn around, turn left, turn right).
2. *ConstFactory<NavAction>* – a factory that creates constant terminals of type NavAction.
3. *ConstFactory<TileType>* – a factory that creates constant terminals of type TileType (an enumerated type that represents various types of tiles in the map).
4. *ConstFunctor<void>* and *NumericConstFactory<bool>* – auxiliary terminals of type void and bool.
5. *IfAssign* – a non-terminal with 5 sub-nodes; the first is of type bool and expresses the condition. If the condition is true, value from sub-node 3 is assigned to variable from sub-node 2; if false value from sub-node 5 is assigned to variable from sub-node 4. Values and variables are of type NavAction.
6. *CompareFunctor<NavAction>* and *CompareFunctor<TileType>* – non-terminals that returns true if both of their inputs are equal and false if not.
7. Logic functors: *And*, *Or*, *Not*.
8. *PrgReturnFunctor(NavAction, N)* – a non-terminal used primarily as root functor of the tree – it has N sub-nodes returning void and one sub-node (the last one) returning NavAction. All sub-nodes are executed one by one and the return value of the last one is returned by the PrgReturnFunctor.

§§ It contains single gaps, double gaps, single, double and triple gaps at corners [11], etc.

## Adaptive Value-switching of Mutation Rate

### Motivation

Most of the existing parameter setting mechanisms, as presented in the previous section, either focus on setting GA-specific parameters such as length of the bit string (e.g. rule (8)), or are not adaptive (e.g. (8), (9) and (10)). The AGA adaptive mechanism described in [10] (formulas (13) and (14)) seems more fit to the task because it implements certain form of convergence detection based on comparison of the maximum and average fitness values. However this approach does little to solve the problem of getting trapped in a local optimum as the method does not discern between local and global optima.
Furthermore – as mentioned hereinbefore – equations (13) and (14) assign the best individual zero crossover and mutation probabilities, while assigning high probabilities to less fit individuals. The reasoning behind this is that the less fit individuals can safely be disrupted by high mutation rates and recombined by crossover (thus employing the solutions with sub-average fitness to search the space [10]), while the highly fit individuals should be preserved.
However, such approach has a very obvious downside which the authors do not seem to address – the highly fit individuals obviously contain the most excellent genetic material available and by disallowing mutation and crossover for these individuals the genetic code they carry becomes isolated and is not used to generate new solutions.

### Description of the AVSMR Mechanism

The idea that the most fit solutions should survive crossover and mutation unmodified is valid, yet that feature can be enforced by using elitism[***]. Keeping that in mind we propose a different adaptation scheme – called AVSMR (Adaptive Value-switching of Mutation Rate) - in order to address the other issues. The main idea is that the mutation probability should be increased to a high value when the search has become trapped in an extreme so as to provide the search process with new genetic material some of which may previously have been unavailable. To determine whether the search has become trapped the adaptive mechanism observes the change of average fitness in time.
To describe the solution in more detail – the algorithm works with 2 values of mutation probability – the normal value and the high value. The algorithm switches from the normal value to the high value once the trigger criterion activates.
The trigger criterion itself is based on a measure that we will herein term a *delta sum*:

$$\Delta S_i = \alpha.\Delta S_{i-1} + \frac{\bar{f}_i - \bar{f}_{i-1}}{\bar{f}_i}, \tag{25}$$

where $\Delta S_i$ is the delta sum in generation $i$ and $\bar{f}_i$ is the average fitness in generation $i$ and $\alpha$ is the feedback coefficient (the experiments have been carried out for $\alpha = 0.4$).
If the delta sum is lower than a pre-set value for a predefined number of generations, that is to say the increase of average fitness in the last few generations is low, indicating that the search

*** The best individual is copied to the next generation unmodified.

has become trapped[†††] – the mutation probability is set to its high value so as to provide the search with new genetic material. As mentioned before, when used in conjunction with elitism it is guaranteed that the best solution is not destroyed by the high mutation probability.
The mutation probability is reset back to its normal value when at least one of the following conditions is true:

1. the average fitness increases enough to produce a sufficiently large delta sum;
2. the maximum fitness increases;
3. mutation has been set to its high value for at least $n$ generations.

The *n*-generation limit is to ensure that the activation does not go on indefinitely (with the high mutation probability it is not very likely that the average fitness will increase enough to satisfy the first condition and maximum fitness may not increase as well).
It has been observed that average fitness typically decreases when the criterion activates because the search process is to a large extent disrupted by the high mutation probability. However after the *n*-generation limit forces the mutation rate back to its normal value, average fitness tends to increase rapidly, thus usually moving away from the local extreme.

**Experimental Results**

Several experiments have been carried out (the specific settings are attached in Appendix **Błąd! Nie można odnaleźć źródła odwołania.**) – Fig. 5 shows performance of the search algorithm with the AGA adaptive mechanism proposed in [10] with constants set according to recommendations. It also shows performance of the search algorithm without any adaptive mechanism and with the adaptive mechanism proposed in this paper. The maximum fitness value achieved is shown for each of the 5 runs displayed.

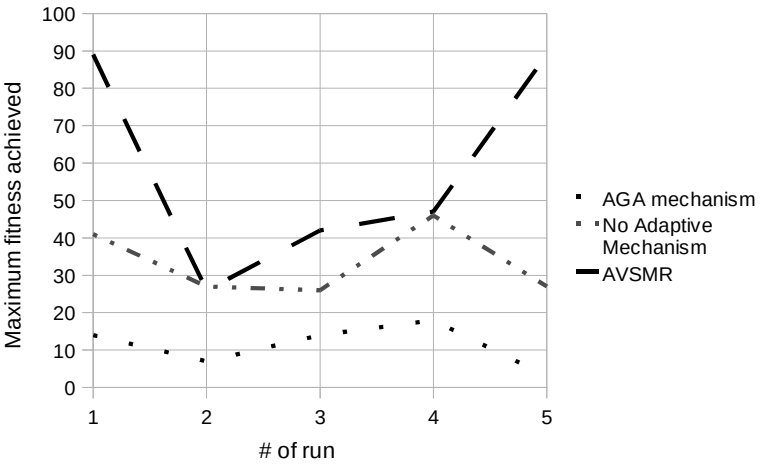


**Fig. 7. Comparison of the AGA Adaptive Mechanism and AVSMR**

††† This may also indicate convergence to the global maximum, it is, however, hardly possible to tell global and local maxima apart unless the algorithm is provided with additional problem-specific data.

As shown, search achieves suboptimal results when running with no adaptive mechanism. This can be ascribed to its inability to escape from local extremes. With no adaptive mechanism the search has not found the global optimum (fitness = 89) in any of the 5 runs.
As expected, the AGA mechanism has caused further deterioration and its results are even worse than those produced in the previous case.
The Value-switching adaptive mechanism proposed in this work improves the process of search – in 2 of the runs the global optimum is found, yet in certain cases not even the high mutation rate is guaranteed to help the search escape from the local maximum (runs 2, 3, 4).

**Further Suggestions**

It has been shown that the adaptive mechanism described in this work is able to effect considerable improvements and that it is able to some extent prevent getting trapped in local maxima. Further experiments should now be carried out in order to ascertain that the principle is valid for a wider range of tasks.
It has also become apparent that even with the high mutation rates it is not always guaranteed that the search will indeed escape from the local maximum. Value-switching, or piecewise continuous relationships for other parameters could perhaps help to alleviate the problem – this issue requires further investigation.

## The Simple Flood Mechanism

Seeing that the AVSMR mechanism described in the previous section is helpful in controlling the search process by helping it to escape from local extremes, yet not completely reliable and not always effective. To address these issues, we have developed another adaptive scheme supposed to provide even greater level of introducing new genetic material into the process.

**Simple Flood Mechanism**

The principle is very straight-forward – once a trapping is detected – a relatively small part of the population is selected – these individuals survive. The rest of the population is destroyed and replaced by newly generated individuals. This method is superior to AVSMR in that a large part of the population is guaranteed to be replaced and the newly generated individuals are generated in the same way that the initial population was.
The trigger criterion has been modified for this task. The first requirement is that the criterion only activates for a single generation at a time as it would probably be useless and possibly even counterproductive to activate the flood mechanism for several successive generations.
The new trigger criterion is still based on the average fitness $\bar{f}_i$ (where $i$ is the number of generation). The criterion stores average fitness $\bar{f}_i$ for $N$ generations ($N-1$ previous generations and the current one; $N = 7$ generations was used in the experiments). The mechanism cannot activate before the $\bar{f}_i$ for at least $N$ generations has actually been collected. Once that is true, the mechanism activates if the following holds:

$$\sum_{i=j}^{j-(N-2)} \bar{f}_i - \bar{f}_{i-1} < \Theta, \tag{26}$$

where $j$ is the number of current generation and $\Theta$ is an activation threshold. It is also possible to interpret the threshold as a relative parameter in which case we can rewrite the equation as follows:

$$\frac{\sum_{i=j}^{j-(N-2)} \bar{f}_i - \bar{f}_{i-1}}{\bar{f}_j} < \Theta \cdot \tag{27}$$

All experiments were carried out using (21).
It is also important to note that once the mechanism activates, the array storing the previous value of average fitness is cleared so it is guaranteed that the mechanism does not activate for the next $N$ generations.
Although the approach seems straight-forward and similar in concept to AVSMR, experimental results point out an important issue. As obvious from Fig. 6, the results achieved by the Simple Flood Mechanism are significantly worse than those produced by the AVSMR – they are in fact worse than those produced by the system when using no adaptive mechanism.

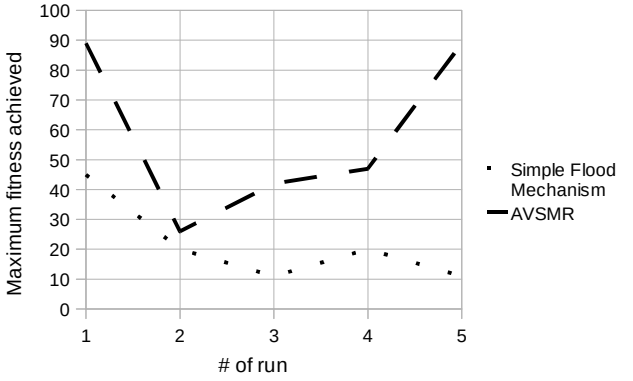


**Fig. 8. Comparison of AVSMR and the Simple Flood Mechanism**

The reason behind this is very simple – although we do introduce new genetic material into the process, the newly generated individuals will generally have very low fitness (usually 0, 3, or 4 at most). Therefore if we apply fitness-proportionate selection to these in the next generation, almost every newly generated individual will be discarded. The survivors on the other hand will now dominate the population. This is especially true later in the evolutionary process when fitness score of the best individual will tend to be vastly greater than that of any randomly generated individual. At this point the next generation will be formed almost exclusively by the best individual, which will almost in every case aggravate the problem of getting trapped in a local extreme instead of solving it.

**Flood Mechanism with Low-pressure Scaling and the New-Blood Mechanism**

There are several ways to alleviate the problem that the Simple Flood Mechanism faces. The objective is – in any case – to create such scheme in which the newly generated individuals mate with the survivors so as to make use of their potentially useful code.

This paper proposes two different ways to achieve this:

1. apply a fitness scaling function with low selection pressure to the GA for several generations following the flood – this mechanism will be referred to as *Flood Mechanism with Low-pressure Scaling (FMLPS)*;
2. once the mechanism activates create only such mating pairs in which at least one individual is newly generated – this mechanism will be referred to as the *New Blood Mechanism*.

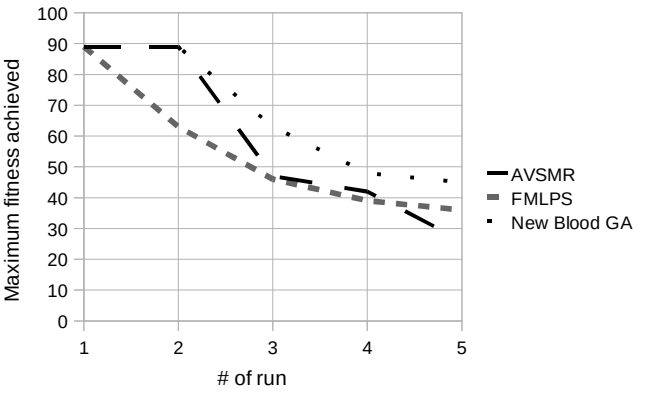


**Fig. 9. Comparison of the AVSMR, FMLPS the New Blood Mechanism**

The experimental results are shown in Fig. 7. FMLPS uses power scaling of 0.3 as the low-pressure scaling. To make the comparison easier, the values are now ordered by fitness rather than by the number of run. This shows that AVSMR is still superior to FMLPS (although FMLPS is – in contrast to the Simple Flood Mechanism – significantly better than vanilla GP). The New Blood GA on the other hand is definitely superior to AVSMR – although it still gets trapped in local extremes, the maximum fitness values achieved are greater than those achieved by the AVSMR.

The influence that some of the parameters such as the number of survivors, or the selection pressure applied by the low-pressure scaling have on the process of search should be subjected to a more systematic investigation. Combining the proposed adaptive mechanisms with some of the concepts introduced by the AGA mechanism could also prove useful – e.g. instead of decreasing the selection pressure using a scaling function the spread of the best individual's copies through the population immediately after the flood could be inhibited by techniques similar to those utilized in AGA.

## Conclusion

It is well known that search processes based on genetic algorithms and genetic programming are prone to getting trapped in local maxima when exploring highly complex spaces. As shown in the paper, search process based on the standard genetic programming approach fails to find the global optimum when applied to the modified version of the artificial ant problem.

This paper investigates the problem and proposes several adaptive mechanism, which should help the search process to escape from local extremes. As shown, the results are considerably better than those of the standard Genetic Programming approach.

Although the results are better, even the proposed algorithms cannot always guarantee that the process will indeed escape from every local maximum it encounters. This stems mainly from the high order of stochasticity that the algorithm is subject to as well as from the size of the searched space.
Related techniques such as adaptive value-switching, or piecewise continuous relationships for other parameters of the search algorithm might provide further improvements. The influence that some of the flood mechanism related parameters (such as the number of survivors, or the selection pressure applied by the low-pressure scaling) have on the process of search may also provide an interesting area for further investigation.

## REFERENCES


1. HYNEK, J.: *Genetické algoritmy a genetické programování*. Grada Publishing, a. s. Praha, 2008. ISBN 978-80-7300-218-3 [In Czech.]
2. ALBA, E., COTTA, C.: *Evolutionary Algorithms*. 2004. http://www.lcc.uma.es/%7Eccottap/papers/eas.pdf
3. MITCHELL, M.: *An Introduction to Genetic Algorithms*. A Bradford Book The MIT Press. Cambridge, Massachusetts, 1999. ISBN 0−262−13316−4
4. SADJADI, F. A.: *Comparison of fitness scaling functions in genetic algorithms with applications to optical processing*. Optical Information Systems II, Proceedings of SPIE: Vol. 5557, 2004.
5. BANERJEE, A.: *Fitness Scaling*. [quot. 11-20-2010]. http://www.cse.unr.edu/~banerjee/scaling.htm
6. LAROSE, D. T.: *Data Mining Methods and Models*. John Wiley & Sons. New Jersey, 2006. ISBN 978-04-7166-656-1
7. BUSETTI, F.: *Genetic algorithms overview*. 2001. http://www.vit.ac.in/academicresearch/res701/RES701DUMP%5CEvolutionary%20Algorithms%5Cgaweb.pdf
8. EIBEN, Á. E., ROBERT, H., MICHALEWICZ, Z.: *Parameter Control in Evolutionary Algorithms*. IEEE Transactions of Evolutionary Computation: 3, 1999. http://www.gpa.etsmtl.ca/cours/sys843/pdf/Eiben1999.pdf
9. THIERENS, D.: *Adaptive mutation rate control schemes in genetic algorithms*. Proceedings of the 2002 IEEE World Congress on Computational Intelligence: Congress on Evolutionary Computation, 2002.

http://dynamics.org/~altenber/UH_ICS/EC_REFS/GP_REFS/CEC/2002/GP_WCCI_2002/7315.PDF

10. SRINIVAS, M., PATNAIK, L. M.: *Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms*. IEEE Transactions on Systems, Man and Cybernetics: 24, 1994. http://eprints.iisc.ernet.in/archive/00006971/02/adaptive.pdf
11. KOZA, J. R.: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. The MIT Press. Cambridge, Massachusetts, 1998. ISBN 0-262-11170-5
12. MONTANA, D. J.: *Strongly Typed Genetic Programming*. Evolutionary Computation: 3, 1995. http://vishnu.bbn.com/papers/stgp.pdf

## [1] Selected Experimental Data

**Maximum number of generations:** 150.
**Elitism:** 3.
**Population size:** 500.
**Maximum depth:** 10.
**Tree generator used:** GrownTreeGenerator.
**Functors used:**

1. VariableFunctor<NavAction>,
2. ConstFactory<NavAction>(ACT_STAY, ACT_TURN_AROUND),
3. ConstFactory<TileType>(tile_empty, tile_wall),
4. CompareFunctor<NavAction>,
5. CompareFunctor<TileType>,
6. ConstFunctor<void>,
7. NumericConstFactory<bool>(0, 1),
8. IfAssign,
9. Logic functors: And, Or, Not,
10. PrgReturnFunctor(NavAction, 10).

**Formal parameters used:**

1. 3 x TileType (tile in front of the actor now and in past 2 turns).
2. 3 x NavAction (previous outputs of the program).

**Selection method:** FitnessRoulette.

**1. No Adaptive Mechanism**

**Crossover probability:** 1.0.
**Mutation probability:** 0.2.
**Results:**

| # of run | max. fitness achieved |
|---|---|
| 1 | 41 |
| 2 | 27 |
| 3 | 26 |
| 4 | 46 |
| 5 | 27 |

**2. The AGA Adaptive Mechanism**

**Crossover probability:** variable.
**Mutation probability:** variable.
**Additional information:** Uses the AGA mechanism with k1 = 1.0, k2 = 0.5, k3 = 1.0, k4 = 0.5.
**Results:**

| # of run | max. fitness achieved |
|---|---|
| 1 | 14 |
| 2 | 7 |
| 3 | 14 |
| 4 | 18 |
| 5 | 3 |

**3. The AVSMR Adaptive Mechanism**

**Fitness scaling:** none.
**Crossover probability:** 1.0.
**Mutation probability:** Basic mutation probability of 0.2; can be increased to 0.8 by the adaptive mechanism.
**Additional information:** Uses the AVSMR mechanism.

**Results:**

| # of run | max. fitness achieved |
|---|---|
| 1 | 89 |
| 2 | 26 |
| 3 | 42 |
| 4 | 47 |
| 5 | 89 |

### 4. The FMLPS Adaptive Mechanism

**Crossover probability:** 1.0.
**Mutation probability:** 0.2.
**Additional information:** Uses the FMLPS mechanism with 20 survivors, power scaling of 0.3, $N = 7$; $\Theta = 0.01$.
**Results:**

| # of run | max. fitness achieved |
|---|---|
| 1 | 89 |
| 2 | 39 |
| 3 | 63 |
| 4 | 36 |
| 5 | 46 |

### 5. The New Blood Adaptive Mechanism

**Crossover probability:** 1.0.
**Mutation probability:** 0.2.
**Additional information:** Uses the New Blood mechanism with 20 survivors, $N = 7$; $\Theta = 0.01$.
**Results:**

| # of run | max. fitness achieved |
|---|---|
| 1 | 89 |
| 2 | 48 |
| 3 | 89 |
| 4 | 45 |
| 5 | 63 |

Pavol SEMANČO[*]

# MINIMIZING MAKESPAN IN GENERAL FLOW-SHOP SCHEDULING PROBLEM USING A GA-BASED IMPROVEMENT HEURISTIC

**Abstract**

*In the paper an improvement heuristic is proposed for permutation flow-shop problem based on the idea of evolutionary algorithm. The approach employs constructive heuristic that gives a good initial solution. GA-based improvement heuristic is applied in conjunction with three well-known constructive heuristics, namely CDS, Gupta's algorithm and Palmer's Slope Index. The approach is tested on benchmark set of 10 problems range from 4 to 25 jobs and 4 to 30 machines. The results are also compared to the best-known lower-bound solutions.*

## 1. INTRODUCTION

A flow-shop production introduces a manufacturing system where $n$ jobs are processed by $m$ machines in the same order. The problem of finding an optimal schedule is referred to as flow-shop scheduling problem (FSSP). In a permutation flow-shop scheduling problem, denoted as PFSSP, the same sequence, or permutation, of jobs is maintained throughout (Pinedo, 2008). The objective of the flow-shop scheduling problem is to meet optimality criterion of minimizing the makespan, total flow time or total weighted flow time. This paper investigates an optimal job sequence for flow-shop scheduling benchmark problem with objective to minimize the makespan. The general scheduling problem for a classical flow shop gives rise to $(n!)^m$ possible schedules (Gupta 1975). For flow-shop scheduling problem Johnson (1954) proposed algorithm that optimally solves a 2-machine flow-shop problem. It was later demonstrated that m-machine flow-shop scheduling problem (FSSP) is strongly NP-hard for $m \geq 3$ (Garey et al., 1976). Permutation FSSP also has to meet standard requirements like a job cannot be processed by two or more machines at a time and a machine cannot process two or more jobs at the same time.

The optimization of FSSP employs the three major types of scheduling algorithm (exact, approximation and heuristic). However, the most common type of scheduling algorithms for NP-hard FSSP is heuristic that produces near-optimal or optimal solutions in reasonable time. The heuristics can be further classified as constructive heuristic and improvement heuristic (or meta-heuristic). The improvement heuristic in contrast to constructive heuristic starts with a initial schedule trying to find an improved schedule. In this paper, the improvement-heuristic

[*] Ing. Pavol Semančo, Technical University of Kosice, Slovakia, email: pavol.semanco@tuke.sk

approach is proposed incorporating the idea of evolution. If no improvement occurs for a certain number of iterations, the algorithm backtracks to the last best result. GA-based improvement heuristic is performed by predetermined number of iterations and report of the best result.

The rest of the paper is organized as follows. The next section reviews the relevant scheduling literature for the flow-shop scheduling heuristics algorithms. In the section, namely GA-based improvement heuristic, the formal description of GA approach is covered. The Section, "Computational Experiments," discusses results obtained from the experiment. The summary of the paper and possible future research ideas are presented in the section, namely Summary and Conclusions.

## 2. RESEARCH BACKGROUND

The model of flow-shop scheduling problem with makespan ($C_{max}$) as an objective function can be specified according to 3-filed classification $\alpha/\beta/\gamma$. The first filed, namely $\alpha$, stands for machine environment. For the flow-shop scheduling the machine environment is denoted as *Fm*, where *m* is the number of the machines. The $\beta$-field specifies the job constraints like for permutation of jobs the *prmu* abbreviation is used. The last field determines the optimally criterion like makespan ($C_{max}$). Based on this 3-field classification the general flow-shop scheduling problem can be denoted as $Fm/prmu/C_{max}$. This notation was firstly suggested by Conway et al. (1967) and until now is handy.

Hejazi and Saghafian (2005) introduced a comprehensive review of alogorithms for flow-shop scheduling problems with makespan criterion. Approaches solving flow-shop scheduling problem range from heuristics, developed, for example, by Palmer (1965), Campbell et al. (1970), Dannenbring (1977) to more complex techniques such as Branch and Bound (Brucker, 1994), Tabu Search (Gendreau, 1998), Genetic Algorithm (Murata et al., 1996), Shifting Bottleneck procedure (Balas and Vazacopoulos, 1998), Ant Colony Algorithm (Blum and Sampels, 2004) and others.

The flow-shop sequencing problem is one of the most well-known classic production scheduling problems. Focusing on the PFSSP with $C_{max}$ objective function, first classical heuristics was proposed by Page (1961). Palmer (1965) adopted his idea and proposed the slope index to be utilized for the m-machine *n*-job permutation flow shop sequencing problem. A simple heuristic extension of Johnson's rule to m-machine flow shop problem has been proposed by Campbell et al. (1970). This extension is known in the literature as the CDS (Campbell, Dudek, and Smith) heuristic. Another method to obtain a minimum makespan is presented Gupta (1972). A significant approach to solving the FSSP proposed Nawaz et al. (1983), in which they point out that a job with larger total processing time should have higher priority in the sequence.

One of the important factors that are quite frequently discussed in FSSP is the setup time (see, for instance, Allahverdi et al., 2008). The setup time represents the time required to shift from one job to another on the given machine. In the flow-shop environment, the setup time is included in the processing times of each job (Hendizadeh et al., 2007).

Modern approaches designated for larger instances are known as meta-heuristics. Approaches that combine different concepts or components of more than one meta-heuristic are named as hybrid meta-heuristic algorithms (Zobolas et al., 2009). Heuristic methods for make-span minimization have been applied, for example, by Ogbu et al. (1990) using Simulated Annealing (SA) and by Taillard (1990) applying Tabu Search (TS) algorithm. Nagar

et al. (1996) proposed a combined Branch-and-Bound (BB) and Genetic Algorithm (GA) based procedure for a flow shop scheduling problem with objectives of mean flow time and make-span minimization. Similarly, Neppalli et al. (1996) were used genetic algorithms in their approach to solve the 2-machine flow shop problem with objectives of minimizing make-span and total flow time. An atypical method based on an Artificial Immune System (AIS) approach, which was inspired from vertebrate immune system, has been presented by Engin and Doyen (2004). They used the proposed method for solving the hybrid flow shop scheduling problem with minimizing $C_{max}$. Obviously, there are plenty of other related approaches to this problem that are identified in survey studies, such as that of Ribas et al. (2010).

## 3. GA-BASED IMPROVEMENT HEURISTIC

Genetic algorithm (GA) forms one of the categories of local search method that operate with a set of solutions. GA is inspired by well-known Darvin's theory about the evolution. GA-based heuristic is started with a set of solutions, also referred to as population. Solutions (or in terms of genetic algorithm, chromosomes) from initial population are taken to form a new population with hope that the new population will be better than the old one. The selection of solutions is performed by a "survival of the fittest" principle to ensure that the overall quality of solutions increases from one generation to the next. This is repeated until some condition (for example number of generations or improvement of the best solution) is satisfied. The framework of proposed GA-based heuristic (GAH) is introduced below.

### NOTATION OF GAH ALGORITHM

The following notation was used:
$G$ number of generations
$P$ population size
$F(s)$ fitness function
$C_{max}$ makespan
$s$ solution represented by a job sequence
$s_i$ initial solution
$p_c$ crossover probability parameter
$p_m$ mutation probability parameter
$c$ chromosome string
$c_p$ parent chromosome
$c_o$ offspring

## GA OPERATORS

The most important parts of the genetic algorithm are genetic operators, referred to as encoding, selection, crossover and mutation operator that impact the whole performance. Proposed GA-based improvement heuristic employs permutation encoding of chromosomes, where each chromosome is a string of numbers (genes), which represents number in a sequence.

For the selection of best chromosomes the roulette wheel method was used. Proposed GAH employs also a method, called elitism, before roulette wheel selection to ensure that at least one best solution is copied without changes to a new population, so the best solution found can survive to end of run.

The crossover operator is carried out with a crossover probability. Crossover selects genes from parent chromosomes and creates a new offspring. It randomly selects a crossover point and everything before this point is copied from the first parent. Then the second parent is scanned and if the scanned gene is not yet in the offspring, it is appended. This method is also called as Single point crossover.

Mutation is also done randomly for each gene and it depends upon another parameter called mutation probability. In this method inversion mutation is adopted where one gene is selected at random and exchanged with another gene mutually. Basically it is an order changing where two numbers are exchanged.

## PSEUDO CODE OF GA FOR MINIMIZING THE MAKESPAN

In the paper GAH is used to search for solution of minimal make-span. Figure 1 introduces the pseudo code of proposed GA-based improvement heuristic in conjunction with constructive heuristic. The constructive heuristic gives a good initial solution to be improved by GA-based heuristic. The objective of the fitness function is to minimize a makespan. The best solution is represented by minimal makespan.

*Step 1* Find initial solution ($s_i$) by selected constructive heuristic
*Step 2* Generate initial population ($P$) based on initial solution and randomness
*Step 3* Apply selection with elitism
*Step 4* Apply crossover with crossover probability ($c_p$)
*Step 5* Apply mutation with mutation probability ($m_p$)
*Step 6* Compute the fitness value for new offspring
*Step 7* Evaluate and save the best chromosome
*Step 8* Go to *Step 2* until the generation value reaches $G$

**Fig. 1. Pseudo code of proposed algorithm**

## 4. COMPUTATIONAL EXPERIMENTS

The experiment was run with objective of minimizing makespan on benchmark dataset that has 10 instances. The dataset ranges from 20 to 500 jobs and 5 to 20 machines.

The CDS, Palmer's Slope Index, Gupta's algorithms and GAH were coded in PHP script, running on a PC with 1.6 GHz Intel Atom and 1GB of RAM. All PHP-coded algorithms has user-friendly interface with eventuality to select whether to run each heuristic itself or all together. It has also an option to draw a Gantt chart. Table 1 contains the input parameters of GAH approach for the experiment purposes.

**Table 1. GA constraints**

| Parameter | Value |
|---|---|
| P | 20 |
| G | 500 |
| $p_c$ | 0.6 |
| $p_m$ | 0.05 |
| *F(s)* | *makespan* |

RESULTS

Results of GA-based heuristic are represented by use of percentage improvement from solution of constructive heuristic and gap from lower-bound solution (LB).

The paper will refer to the 3-heuristic GAH versions, namely P-GAH (Palmer-GAH), CDS-GAH and G-GAH (Gupta-GAH). Table 2 summarizes the results for all 10 instances and also shows percentage improvement of GAH over constructive heuristic. Table 1 also introduces the best-known lower bounds and percentage gap from the best-known bound for the best GAH result. In the table the results are displayed for Palmer alone, CDS alone, NEH alone, P-GAH, CDS-GAH and G-GAH.

**Table 2. Makespans and improvements for 10 benchmark problems**

| No. | Problem Size | LB | Gupta | | | CDS | | | Palmer | | | Best GAH | % gap from LB |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Single pass | G-GAH | % Imprv GAH | Single pass | CDS-GAH | % Imprv GAH | Single pass | P-GAH | % Imprv GAH | | |
| 1. | 4x4 | 156 | 157 | 156 | 0.64 | 156 | 156 | 0.00 | 157 | 156 | 0.64 | 156 | **0.00** |
| 2. | 5x4 | 51 | 51 | 51 | 0.00 | 51 | 51 | 0.00 | 53 | 51 | 3.77 | 51 | **0.00** |
| 3. | 6x5 | 7.7 | 7.7 | 7.7 | 0.00 | 7.7 | 7.7 | 0.00 | 8.35 | 7.7 | 7.78 | 7.7 | **0.00** |
| 4. | 7x7 | 65 | 65 | 65 | 0.00 | 67 | 65 | 2.99 | 75 | 65 | 13.33 | 65 | **0.00** |
| 5. | 8x7 | 69 | 69 | 66 | 4.35 | 66 | 66 | 0.00 | 70 | 69 | 1.43 | 66 | **-4.55*** |
| 6. | 10x12 | 93 | 106 | 97 | 8.49 | 104 | 100 | 3.85 | 104 | 96 | 7.69 | 96 | 3.13 |
| 7. | 12x12 | 104 | 111 | 110 | 0.90 | 114 | 107 | 6.14 | 115 | 108 | 6.09 | 107 | 2.80 |
| 8. | 15x18 | 141 | 163 | 150 | 7.98 | 153 | 149 | 2.61 | 146 | 142 | 2.74 | 142 | 0.70 |
| 9. | 23x25 | 219 | 264 | 233 | 11.74 | 259 | 232 | 10.42 | 241 | 225 | 6.64 | 225 | 2.67 |
| 10. | 30x25 | 249 | 285 | 260 | 8.77 | 271 | 258 | 4.80 | 274 | 261 | 4.74 | 258 | 3.49 |

LB – Best-known lower bound solution
Single pass – makespan of constructive heuristic
* new lower-bound solution

Overall neither of 3-heuristic GAH versions performed significantly better, although all of them gave feasible improved solutions. For flow-shop scheduling problem sizes range from 4 to 7 machines and jobs, GAH matched the best-known lower bound solutions. for 24 of the 30 problems and found a new upper bound for one problem. For the fifth problem the new lower bound was found by the GA-based improvement heuristic.

Average computational times (CPU) for each size of the problem are summarized and depicted in Figure 2. The computation times of course vary by the size of the problem. The variance, within three versions of GAH was not significant.

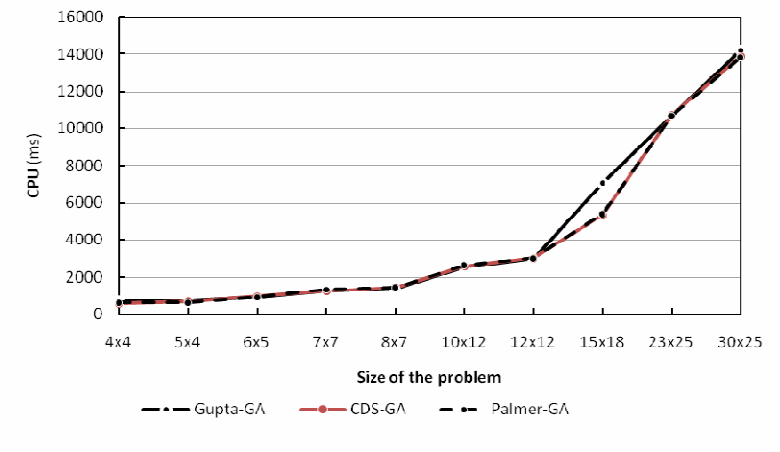


**Figure 2. Computational times of GAH algorithm for each size of the problem.**

## 5. SUMMARY AND CONCLUSIONS

In presented study, the scheduling problem with sequence-dependent operations was dealt. The main idea is to minimize the make-span time and thereby reducing the idle time of both jobs and machines since these criteria are often applied for operational decision-making in scheduling. Under above mentioned consideration an improvement heuristic based on evolutionary algorithm (GAH) is proposed and applied to the permutation flow-shop scheduling problem. The GA-based heuristic approach uses a constructive heuristic to get an initial solution that tries to find improvements iteratively.

The GAH algorithm was used to improve upon heuristics, namely, Palmer, CDS and Gupta. For all three heuristics, GAH showed significant improvements. The best improvements were compared well with the best-known lower bounds. The average gap from the best-known lower bound was 0.82% for all ten problems.

Future research should look at this heuristic for the more difficult flow-shop scheduling problems involving sequence-dependent setup times. Different objective functions can also be tested.

## REFERENCES

1. ALLAHVERDI. A.. NG. C.T.. CHENG. T.C.E.. & KOVALYOV. M.Y. (2008). *A survey of scheduling problems with setup times or costs*. European Journal of Operational Research. 187. 985-1032.
2. BALAS. E.. & VAZACOPOULOS. A. (1998). *Guided local search with shifting bottleneck for job shop scheduling*. Management Science. 44 (2). 262-275.
3. BLUM. C.. & M. SAMPELS. (2004). *An ant colony optimization algorithm for shop scheduling problems*. Journal of Mathematical Modelling and Algorithms. 3(3). 285-308.
4. BRUCKER. P.. JURISCH. B.. & SIEVERS. B. (1994). *A branch and bound algorithm for the job shop scheduling problem*. Discrete Applied Mathematics. 49(1). 109-127.

5. CAMPBELL. H.G.. DUDEK. R.A.. & SMITH. M.L. (1970). *A heuristic algorithm for the n job. m machine sequencing problem.* Management Science. 16(10). 630-637.
6. DANNENBRING. D.G. (1977). *An evaluation of flow shop sequencing heuristics.* Management Science. 23(11). 1174-1182.
7. ENGIN. O.. & DOYEN. A. (2004). *A new approach to solve hybrid flow shop scheduling problems by artificial immune system*. Future Generation Computer Systems. 20. 1083-1095.
8. GAREY. M.R.D.. JOHNSON. D.S.. & SETHI. R. (1976). *The complexity of flowshop and jobshop scheduling*. Mathematics of Operations Research. 1. 117-129.
9. GENDREAU. M.. LAPORTE. G.. & SEMET. F. (1998). *A tabu search heuristic for the undirected selective travelling salesman problem*. European Journal of Operational Research. 106(2-3). 539-545. Elsevier.
10. GUPTA. J.N.D. (1972). *Heuristic algorithms for multistage flowshop scheduling problem.* AIIE Transactions. 4 (1). 11-18.
11. GUPTA. J.N.D. (1975). *Analysis of combinatorial approach to flowshop scheduling problems*.
12. HEJAZI. S.R.. & SAGHAFIAN. S. (2005). *Flowshop scheduling problems with makespan criterion: a review*. International Journal of Production Research. 43(14). 2895-2929.
13. HENDIZADEH. S.H.. ELMEKKAWY. T.Y.. & WANG. G.G. (2007). *Bi-criteria scheduling of a flowshop manufacturing cell with sequence dependent setup time*.. European Journal of Industrial Engineering. 1. 391-413.
14. JOHNSON. S. M. (1954). *Optimal two and three stage production schedules with set-up times*. Naval Research Logistics Quarterly. 1. 61-68.
15. NAGAR. A.. HERAGU. S.S.. & HADDOCK. J. (1996). *A combined branch-and-bound and genetic algorithm based approach for a flowshop-scheduling problem.* Annal. Oper. Res.. 63. 397-414.
16. NAWAZ. M.E.. ENSCORE. I.. & HAM. I. (1983*). A heuristic algorithm for the m machine. n job flow shop sequence problem*. OMEGA. 11 (1). 91-95.
17. NEPPALLI. V.R.. CHEN. C.L. & GUPTA. J.N.D. (1996). *Genetic algorithms for the two-stage bicriteria flowshop problem*. Eur. J. Oper. Res.. 95. 356–373.
18. OGBU. F.A.. & SMITH. D.K. (1990). *The application of the simulated annealing algorithm to the solution of the n/m/Cmax owshop problem*. Computers & Operations Research. 17. 3243-253.
19. PALMER. D. S. (1965). *Sequencing jobs through a multi-stage process in the minimum total time - a quick method of obtaining a near optimum.* Opers Res. Q.. 16. 101-107.
20. PINEDO. M. (2008). *Scheduling: Theory. algorithms and Systems*. Prentice Hall. New Jersey: Springer.
21. RIBAS. R.. LEISTEN. J.M. (2010). *Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective*. Computers and Operations Research. 37(8).1439-1454.
22. ZOBOLAS. G. I.. TARANTILIS. C. D.. & IOANNOU. G. (2009) *Minimizing makespan in permutation flow shop scheduling problems using a hybrid metaheuristic algorithm*. Computers and Operations Research. 36 (4). 1249-1267.

Daniel GĄSKA[*], Antoni ŚWIĆ[**]

# THE STANDARDIZED AUDIT OF SAFETY AND THE RELIABILITY OF ERP SYSTEMS

**Abstract**

*The paper presents the possibility of the realization of the evaluation of the security of the Enterprise Resource Planning (ERP)systems following the regulations specified by European and Polish norms which relate to the safety of computer systems (information systems) in enterprises with the special regard to the ERP systems. It also introduces the possibility of creating the security system programme and the actions executed during the evaluation.*

## 1. INTRODUCTION

ERP systems are characterized by the modular structure, that is each system contains several modules which create the complete entity. The modules can work in various configurations, which means that the firm does not have to buy the whole system. It is enough to buy the chosen modules which will co-operate with each other, thus exchanging the introduced information.

Assuring the safety to the computer resources of firms is currently one of most popular services on the IT market. However, the majority the services aiming at the evaluation and the improvement of the computer safety in the firm do not take into account the regulations specified by Polish and European norms. Thanks to the introduction of the norms into the process of the evaluation of the computer safety of the firm it will be possible to compare various ERP systems in relation to the safety. The standardized process of the evaluation of the safety will give us the true representation of the system and its protections.

The safety of the ERP system is the necessary element to ensure the correct functioning of the whole enterprise. Because all the elements of the enterprise are integrated with the ERP system, the possibility of maintaining the safety of the system

[*] M.Sc. Eng. Daniel Gąska Lublin University of Technology, Institute of Technological Information Systems, Lublin, Poland, e-mail: d.gaska@pollub.pl
[**] D.Sc. Eng. Assoc Prof. Antoni Świć Lublin University of Technology, Institute of Technological Information Systems, Lublin, Poland, e-mail: a.swic@pollub.pl

seems to be the essential element in the context of the utilization of ERP systems in the enterprises which introduced the system [1].

## 2. AUDITING ERP SYSTEMS ACCORDING TO THE DIRECTIVES OF SACA

ISACA (Information Systems Audit and Control Association) is an international association of the people in charge of the issues concerning the audit, control, safety and other aspects of the management of the information systems.

It is one the ways of introducing the reliable evaluation of the information systems and, in particular, of the ERP systems. The association proposes solutions which enable the execution of the audit of the information system realized on the basis of standards specified in SISA Standards for Information Systems Auditing [4].

It is imperative that the organization's system management fully understand and support the IS auditor's role(s) as it relates to the ERP system or implementation project .The IS Auditing Guideline should be reviewed and considered within the context of the ERP system and related initiatives of the organization (Fig. 1.).

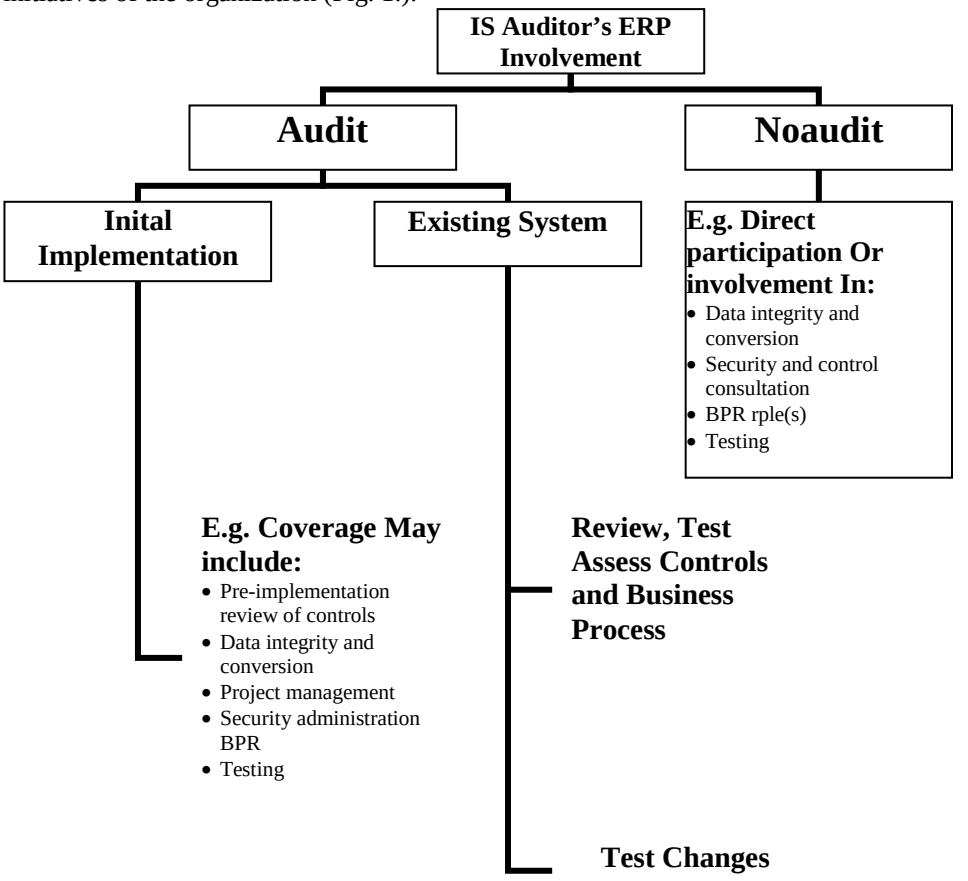


**Fig. 1. IS Auditor's ERP Involvement [3]**

ERP Knowledge and Skill Requirements [3]

| | **ERP System** | **Implementation Project** |
|---|---|---|
| Background knowledge of the IS auditor | An understanding of financial and management controls and control risks generally<br>A thorough understanding of the application of professional IS auditing standards<br>A thorough understanding of IT related controls and control risks in the following areas:<br>• IT environment<br>• Applications/processing<br>An understanding of client/server architectures<br>An understanding of operating systems and database management systems<br>A general understanding of ERPs and their design and deployment philosophies, including their effect on the audit trail<br>An understanding of the ERP modules and how they are configured, integrated and deployed<br>An understanding of security and authorization concepts in an ERP setting | An understanding of project management practices and controls generally<br>An understanding of project management practices and controls in the area of IT<br>An understanding of IT-related systems development methodologies and standards, including change management<br>An understanding of business process reengineering principles and application of such |
| Skills of the IS auditor | A seasoned IS audit professional who is able to focus on the key areas of control risk in an ERP setting<br>An understanding of computer-assisted audit techniques (CAATs) and how to apply them in an ERP setting. An ability to recognise where additional skills/expertise (such as financial and regulatory) are required | Experience in the review and assessment of implementation projects |

| | | |
|---|---|---|
| How to Acquire skills | Certification as a professional auditor<br>Certification as a professional IS auditor, such as CISA ERP learning opportunities especially as part of the end-user community<br>Practical, on-the-job experience<br>Self study, research, Internet, etc. | Enroll in specialist training courses focusing on Practical, on-the-job experience<br>Self study, research, Internet, etc. |

While carrying out the audit of the ERP systems, you should consider the most important areas of the system. Fig. 2 shows which areas you should examine more exactly.

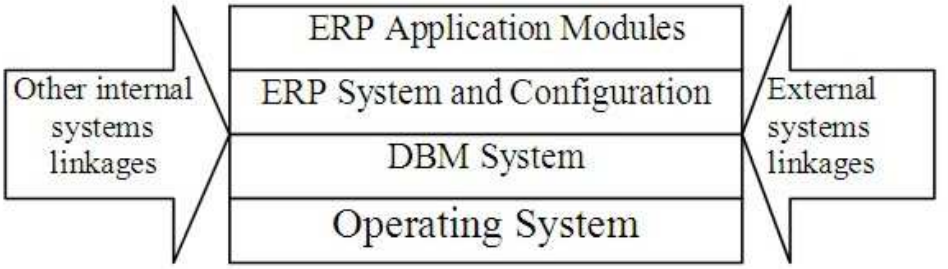


**Fig. 2. General Elements of and Questions on ERP System Implementation**

## 3. THE FEATURES OF THE SECURITY

Every component feature of the security depends on the architectural organization of the modules of the ERP systems and on the properties of the security of these modules.

Every component feature on the level of the system can depend on several component features on the level of the module [6].

The security of the ERP systems cannot be described by one feature. Some of the features can be expressed as probability, other features are deterministic some elements can be introduced quantitatively, whereas other aspects can only be described qualitatively.

The examples of the analysis of the security of ERP systems on the level of modules can be situations in which:

- thearchitecture of the system contains redundancy, the readiness of the system depends on the features of the integrity of the redundant modules;
- if the architecture contains the mechanisms of the protection of the system, the protection of the system depends on the features of the readiness of the modules which realize the mechanism of the protection;
- if the architecture contains modules controlling internal passing of the information between the various parts of the system, then the security of the system depends on the features of the protection of these modules.

In order to realize the evaluation of the security of the ERP systems, the program of this evaluation needs to be defined. This is possible after defining the aims of the evaluation of security, the requirements of the system and the specification of the system. Figure 1 represents three elements of the full analysis of the system, with the third element of the analysis being the evaluation of the security of the ERP systems [13].

It is important to remember that the information given in the document relating to the system requirements (SRD) and in the document relating the specification of the system (SSD) must be complete and exact to make the evaluation of the system possible.

If it turns out that at any phase of carrying out the evaluation, some information is missing or is incomplete, the consultation with the authors of SRD and SSD is required. By asking them detailed questions, it will be possible to receive the required information. It is important that the received additional information is specified in suitable documents [15].

## 4. EVALUATION CRITERIA ERP SYSTEMS ACCORDING TO THE DIRECTIVES OF INTERNATIONAL STANDARD ISO/IEC 15408

Information held by ERP system is a critical resource that enables organizations to succeed in their mission. Additionally, individuals have a reasonable expectation that their personal information contained in ERP products or systems remain private, be available to them as needed, and not be subject to unauthorized modification. ERP products or systems should perform their functions while exercising proper control of the information to ensure it is protected against hazards such as unwanted or unwarranted dissemination, alteration, or loss. The term ERP security is used to cover prevention and mitigation of these and similar hazards.

Many consumers of ERP lack the knowledge, expertise or resources necessary to judge whether their confidence in the security of their ERP products or systems is appropriate, and they may not wish to rely solely on the assertions of the developers. Consumers may therefore choose to increase their confidence in the security measures of an ERP product or system by ordering an analysis of its security (i.e. a security evaluation) [10].

The Common Criteria (CC) with international standard ISO/IEC 15408 can be used to select the appropriate ERP security measures and it contains criteria for evaluation of security requirements.

The Common Criteria (CC) with international standard ISO/IEC 15408 plays an important role in supporting techniques for consumer selection of ERP security requirements to express their organizational needs. The Common Criteria (CC) with international standard ISO/IEC 15408 is written to ensure that evaluation fulfils the needs of the consumers as this is the fundamental purpose and justification for the evaluation process.

Consumers can use the results of evaluations to help decide whether an evaluated product or system fulfils their security needs. These security needs are typically identified as a result of both risk analysis and policy direction. Consumers can also use the evaluation results to compare different products or systems. Presentation of the assurance requirements within a hierarchy supports this need.

The Common Criteria (CC) gives consumers — especially in consumer groups and communities of interest — an implementation-independent structure termed the Protection Profile (PP) in which to express their special requirements for ERP security measures in a Target of Evaluation (TOE).

In order to achieve greater comparability between evaluation results, evaluations should be performed within the framework of an authoritative evaluation scheme that sets the standards, monitors the quality of the evaluations and administers the regulations to which the evaluation facilities and evaluators must conform.

The Common Criteria (CC) does not state requirements for the regulatory framework. However, consistency between the regulatory frameworks of different evaluation authorities will be necessary to achieve the goal of mutual recognition of the results of such evaluations. Figure 3 depicts the major elements that form the context for evaluations.

Use of a common evaluation methodology contributes to the repeatability and objectivity of the results but is not by itself sufficient. Many of the evaluation criteria require the application of expert judgment and background knowledge for which consistency is more difficult to achieve.

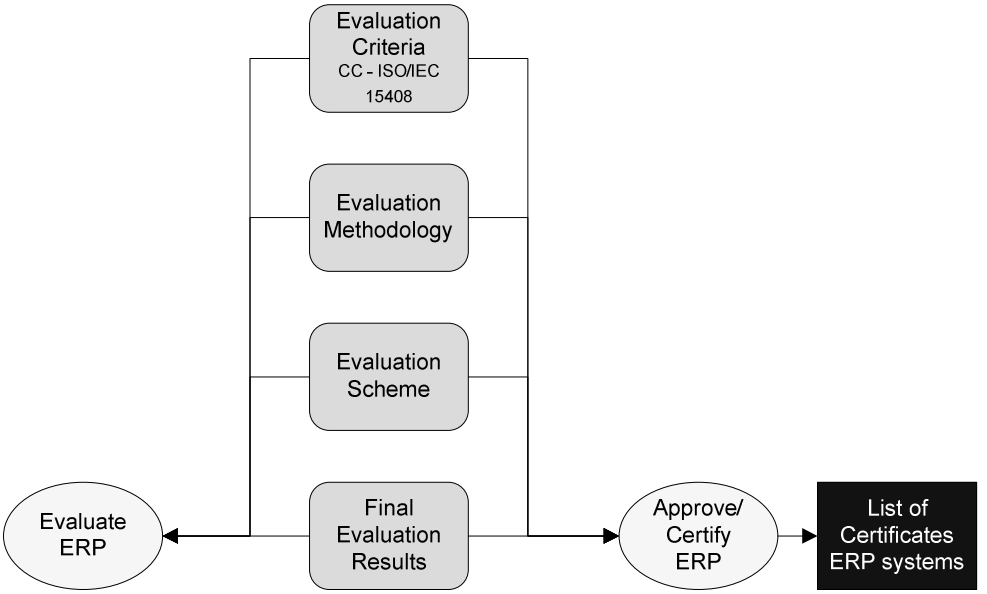


**Fig. 3. Evaluation context**

In order to enhance the consistency of the evaluation findings, the final evaluation results could be submitted to a certification process. The certification process is the independent inspection of the results of the evaluation leading to the production of the final certificate or approval. The certificate is normally publicly available. It is noted that the certification process is a means of gaining greater consistency in the application of ERP security criteria.

Security is concerned with the protection of assets from threats, where threats are categorized as the potential for abuse of protected assets. All categories of threats should be considered; but in the domain of security greater attention is given to those threats that are related to malicious or other human activities. Figure 3 illustrates high level concepts and relationships.

Safeguarding assets of interest is the responsibility of owners who place value on those assets. Actual or presumed threat agents may also place value on the assets and seek to abuse assets in a manner contrary to the interests of the owner. Owners will perceive such threats as

potential for impairment of the assets such that the value of the assets to the owners would be reduced. Security specific impairment commonly includes, but is not limited to, damaging disclosure of the asset to unauthorized recipients (loss of confidentiality), damage to the asset through unauthorized modification (loss of integrity), or unauthorized deprivation of access to the asset (loss of availability) [10].

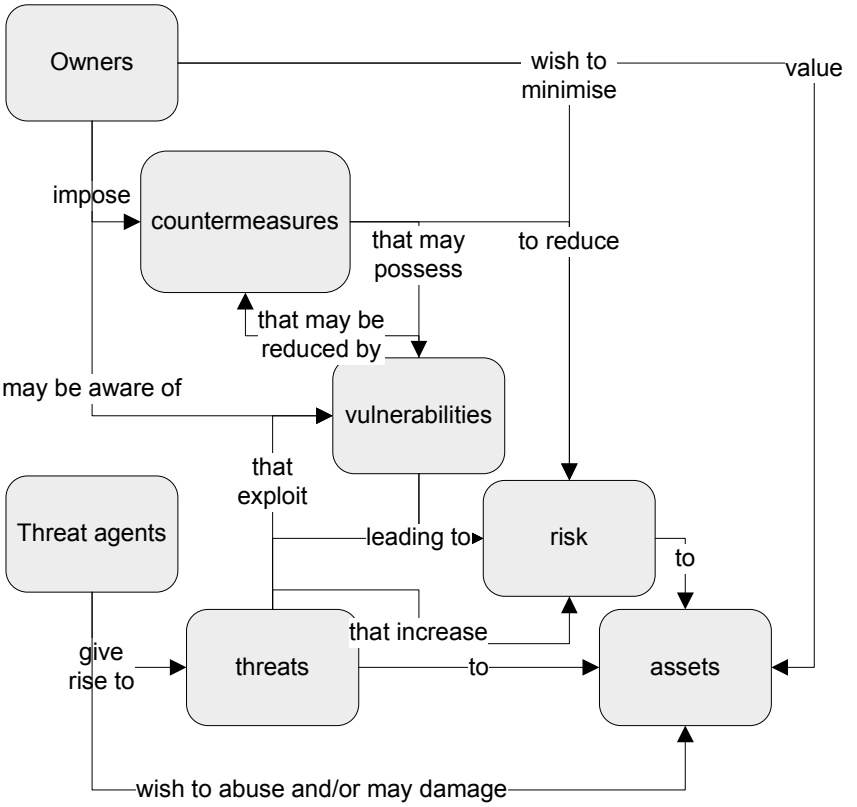


**Fig. 4. Security concepts and relationships**

The owners of the assets will analyze the possible threats to determine which ones apply to their environment. The results are known as risks. This analysis can aid in the selection of countermeasures to counter the risks and reduce it to an acceptable level.

Countermeasures are imposed to reduce vulnerabilities and to meet security policies of the owners of the assets (either directly or indirectly by providing direction to other parties). Residual vulnerabilities may remain after the imposition of countermeasures. Such vulnerabilities may be exploited by threat agents representing a residual level of risk to the assets. Owners will seek to minimize that risk given other constraints.

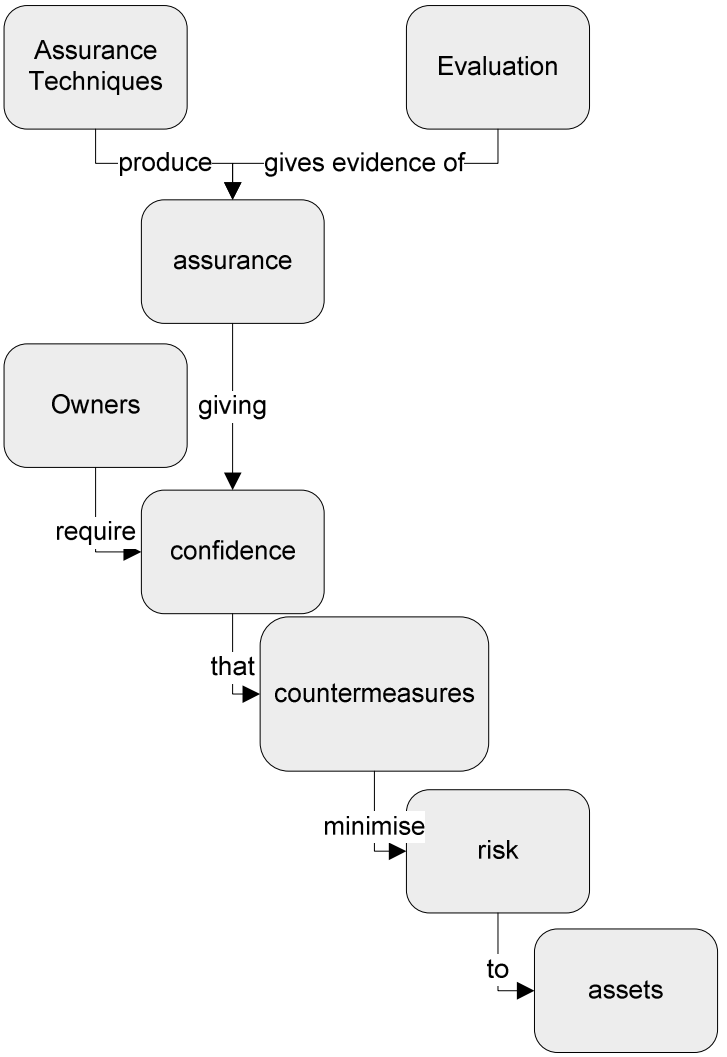


**Fig. 5. Evaluation concepts and relationships**

Owners will need to be confident that the countermeasures are adequate to counter the threats to assets before they will allow exposure of their assets to the specified threats. Owners may not themselves possess the capability to judge all aspects of the countermeasures, and may therefore seek evaluation of the countermeasures. The outcome of evaluation is a statement about the extent to which assurance is gained that the countermeasures can be trusted to reduce the risks to the protected assets. The statement assigns an assurance rating of the countermeasures, assurance being that property of the countermeasures that gives grounds for confidence in their proper operation. This statement can be used by the owner of the assets in deciding whether to accept the risk of exposing the assets to the threats. Figure 5 illustrates these relationships [11].

Owners of assets will normally be held responsible for those assets and should be able to defend the decision to accept the risks of exposing the assets to the threats. This requires that the statements resulting from evaluation are defensible. Thus, evaluation should lead to objective and repeatable results that can be cited as evidence.

Many assets are in the form of information that is stored, processed and transmitted by ERP products or systems to meet requirements laid down by owners of the information. Information owners may require that dissemination and modification of any such information representations (data) be strictly controlled. They may demand that the ERP product or system implement ERP specific security controls as part of the overall set of security countermeasures put in place to counteract the threats to the data ERP systems are procured and constructed to meet specific requirements and may, for economic reasons, make maximum use of existing commodity ERP products such as operating systems, general purpose application components, and hardware platforms. ERP security countermeasures implemented by a system may use functions of the underlying ERP products and depend upon the correct operation of ERP product security functions. The ERP products may, therefore, be subject to evaluation as part of the ERP system security evaluation [11].

## 4.1. COLLECTING THE INFORMATION FOR EXECUTING THE EVALUATION OF THE SECURITY

Before beginning the realization of the audit of the security of the ERP system, it is necessary to execute the review of the system in order to relate the system to its mission. The system should be decomposed into modules and elements. It is also necessary to remember that the process of decomposing leads to demonstrative schemes/patterns and additional descriptions.

It is recommended that while realizing the process of decomposing of the ERP system the description should include:

- all modules of interface to the process, to the application, to the database and to external systems;
- communication channels which in the large measure decide about the security of the system;
- processing modules connected with the application;
- the interaction of the modules;
- existence of the divisions and the distances between the divisions of the firm.

After completing the process of decomposingit is important to know that the majority of ERP systems are based on module architecture which where he separate modules freely combined.

In order to conduct the evaluation of the system, it is essential to extract the necessary information from SRD and SSD documents.

It is recommended to combine the requirements specified in SRD and the level of security assured by the system, as specified in SSD, and the comparison between them in order to arrive at the precise quantitative and qualitative definition and the range of their value, if this can be applied, in following casess:

- thelimits of the ERP systems;
- thekind of threats and their ways of spreading;
- the influencing conditions which can create the threat inside the system;
- the ways of reducing the risk of the situations which may pose the threat;

- the ways of reducing the risk of the situations of connecting various phenomena which, in turn, may pose the threat;
- the allocation of the security of modules and the elements of the system;
- the way in which various modules and the elements of the system interact and the possibility losing the security which can happen as the result of the interaction;
- matters which are outside the range of the system;
- thegenerally accessible knowledge and the range within which the security of the system is to be evaluated.

## 4.2. ACTIONS EXECUTED DURING THE EVALUATION OF THE SECURITY OF THE SYSTEM

The list of the actions to be realized during the evaluation process comes from the reduced list of the objects of the evaluation broadened by the subjects included in the evaluation in which we should consider:

- thekind of analysis and defining of the proprieties required for the justification of the evaluation of the security;
- thelevel of the priority of every action which is part o the evaluation of the security;
- theknowledge and skills necessary for the execution of the required analysis and the definition of the properties;
- limitations in the schedule of the evaluation of the security, resulting from the long time of marking the different proprieties of the system;
- theavailability of the chosen staff;
- tools and services necessary for the execution of required analyses and delimitation of the propriety of the system;
- estimation of the cost and duration of every analysis and the definition of the properties of the system.

It is often necessary to combine several techniques which will be complimentary and will make it possible to define the security of the system realizing earlier planned actions.

The programme of the evaluation of the security of the ERP systems should contain such elements as:

- the object of the evaluation;
- the criteria which need to be taken into consideration;
- the actions taken during the evaluation;
- therequired increase of the level of the confidence;
- the schedule of the evaluation in which you should consider the long time of the duration of some investigations.

## 4.3. TECHNIQUES OF DEFINING THE PROPRIETIES OF THE SYSTEM FOR FURTHER EVALUATION

Chosen techniques could be either analytic, using only the documentation of the system or experimental, requiring the access to the realized system [2].

The results received with the help of the alternative techniques of defining proprieties can be quantitative or qualitative, or can also be the combination of both kinds.

Various methods of defining properties can be applied, but it is recommended that in each case, the report of the evaluation contained the reference he documents describing the applied methods.

The following steps should be executed with reference to each kind of the threat:

- check if the threat exists and if it does, check if there is the accessible certification and if it is valid in the working conditions specified in SRD or if it follows the regulation;
- if the satisfying certification is not available it is recommended to execute the suitable analysis of the risk.

The experimental techniques of defining the proprieties of the system are the supplement of the analytic techniques.

Every time the analytic techniques cannot guarantee the evaluation of the security level of the system, the execution of experimental defining of the proprieties, in order to evaluate those aspects which do not have complete data.

## 5. THE REPORT OF THE EVALUATION OF ERP SYSTEMS

The report of the evaluation of the safety of the ERP system should also contain the following information:

- the compilation of the data from the document relating to system requirements and the document relating to the specification of the system of, for example the requirements of safety, working conditions, service, etc.;
- theanalysis of the system, its modular and functional structure, the risks to which the system was subjected, elements and components and the relationship between them, etc.;
- thelist of actions recommended for further evaluation of the analysis and further investigations.

## 8. SUMMARY

Summing up should affirm, that the performance of the standardized programme of the opinion of the safety of the ERP systems will make possible the creation such conditions the work, which will give the tool to the enterprise thanks which what level of the safety of the ERP systems the qualification possible will be he is in the enterprise. Does the introduce methodology give the answer what to the safety of one of the most important links of the firm.

## REFERENCES


1. *Benjamin B. Bae, Ph.D., and Paul Ashcroft, Ph.D.*: Implementation of ERP Systems. Accounting and Auditing Implications, ISACA Journal Volume 5, 2004.
2. *Standards*, Guidelines and Procedures for Auditing and Control Professionals, Published by Information Systems Audit and Control Association, February 2007.
3. *Steve Maguire*, Writing solid code: Microsoft's techniques for developing bug-free C programs, Published by Microsoft Press Washington 1993.
4. *PN-I-13335-1*: Technika informatyczna. Wytyczne do zarządzania bezpieczeństwem systemów informatycznych. Pojęcia i modele bezpieczeństwa systemów informatycznych.

5. *PN-I-02000*: Technika informatyczna. Zabezpieczenia w systemach informatycznych. Terminologia.
6. *PN-ISO 10011-1*: Wytyczne do audytowania systemów jakości. Audytowanie.
7. *ISO/IEC 17799*: Technologie informacyjne. Zasady postępowania w zarządzaniu bezpieczeństwem informacji.
8. *PrPN-I-13335-2*: Technika informatyczna. Wytyczne do zarządzania bezpieczeństwem systemów informatycznych. Aspekty zarządzania i planowania.
9. *PrPN-I-1335-3*: Technika informatyczna. Wytyczne do zarządzania bezpieczeństwem systemów informatycznych. Techniki bezpieczeństwa.
10. *ISO/IEC 15408-1*: Information technology - Security techniques - Evaluation criteria for IT security - Part 1: Introduction and general model.
11. *ISO/IEC 15408-2*: Information technology - Security techniques - Evaluation criteria for IT security - Part 2: Security functional components.
12. *ISO/IEC 15408-3*: Information technology - Security techniques - Evaluation criteria for IT security - Part 3: Security assurance components.
13. *PN-EN 61069*: Pomiary i sterowanie procesami przemysłowymi. Wyznaczanie właściwości systemu w celu jego oceny: Część 2: Metodologia oceny.
14. *PN-EN 61069*: Pomiary i sterowanie procesami przemysłowymi. Wyznaczanie właściwości systemu w celu jego oceny: Część 5: Ocena niezawodności systemu.
15. *PN-EN 61069*: Pomiary i sterowanie procesami przemysłowymi. Wyznaczanie właściwości systemu w celu j**e**go oceny: Część 7: Ocena bezpieczeństwa systemu.