

KNOWLEDGE-BASED AND CP -DRIVEN METHODOLOGY FOR DEDICATED DSS DESIGN

Zbigniew Banaszak

*Technical University of Koszalin, Department of Computer Science and Management,
Poland, banaszak@tu.koszalin.pl*

Izabela Tomczuk-Piróg

*Opole University of Technology, Department of Management and Production
Engineering, Opole, Poland, itomczuk@po.opole.pl*

Paweł Sitek

*Technical University of Kielce, Department of Electrical and Computer Engineering,
Poland, sitek@tu.kielce.pl*

Abstract— An analysis of unified framework standing behind a methodology in object oriented decision support system design has been analysed. First of all the consistency of the assumed knowledge bases describing an object (enterprise) and requests (standard options supporting a decision maker), respectively are examined. Then the knowledge base representation is transformed into representation of so called constraint satisfaction problem (CSP). Possible ways of the CSP decomposition as well as possibility of different programming languages application lead to a problem of searching for a distribution strategy which gives a possibility to interact in an on-line mode.

1. INTRODUCTION

Decision making supported by task-oriented software tools plays a pivotal role in modern enterprises; the commercially available ERP systems are not able to respond in an interactive on-line/real-time mode. A new generation of DSS (Decision Support System) that enable a fast prototyping of production flows in multi-project environment as well as an integrated approach to a layout planning, production routing, batch-sizing and scheduling problems is needed. In that context, the constraint logic programming techniques allowing declarative representation of a decision making problem provide quite an attractive alternative [6,7]. The contribution includes therefore modelling of decision making and searching strategies development.

2. PROBLEM STATEMENT

Given are knowledge-based representation of small and medium size enterprises and knowledge-based of context-oriented queries. The *SME*'s specification includes parameters describing parameters such as the number of resources available, their efficiency, capacity; time constrained resources availability etc., as well as relations linking particular workstations, pallets, tools, manpower etc. Due to the prevailing unique character of work orders in *SME*'s, it is necessary to be able to evaluate quickly and precisely the possibility of balancing production capacity of a company with the requirements set in agreement with the customer [1]. Solution to a makespan-feasible problem permits investigating the effect of a new work order impact on the performance of a manufacturing system. The queries encompassing the standard options of *SME* management are, in turn, specified by data relevant to a production order requirements and the enterprise capability. The objective is to find a *DSS* allowing one to respond to any question related to the *SME* considered in an interactive mode. The problem we are facing deals with a question, whether there is a way which enables evaluating a possibility of relevant *DSS* design for the commercially available programming languages. The graphical illustration of the problem considered is presented in Fig. 1.

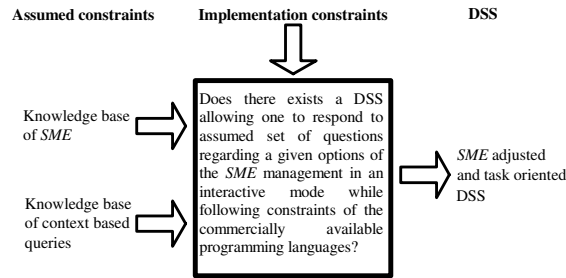


Figure 1 - Illustration of the problem statement

3. KNOWLEDGE-BASED REPRESENTATION

It is assumed that any system can be specified in terms of knowledge base composed of facts and rules determining the system's properties and relations linking them respectively. Formally, *RW* knowledge base is defined as a pair: $RW = \langle \alpha, F(\alpha) \rangle$, where, $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_N)$ – is a sequence of elementary formulas specifying system's properties; α_i – is the *i*-th assertion (specified in terms of binary logic), and $a_i = w(\alpha_i) \in \{0, 1\}$ is a logic value of the assertion α_i

$F(\alpha) = \{F_1(\alpha), F_2(\alpha), \dots, F_k(\alpha)\}$ – is a sequence of facts specifying relations among properties (in terms of logic operators: conjunction, disjunction, negation, and implication); $F_i(a)$ – refers to a binary value of expression $F_i(\alpha)$.

So $a = (a_1, a_2, \dots, a_m)$ corresponds to any $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_N)$. Consequently (a_1, a_2, \dots, a_m) refers to a sequence of values associated with $w(\alpha)$.

In any system description, the following categories can be distinguished:

$\alpha_x = \{ \alpha_{x1}, \alpha_{x2}, \dots, \alpha_{xk} \}$ – a set of elementary formulas specifying the so called input system variables, $\alpha_{xi} \in \{ \alpha_1, \alpha_2, \dots, \alpha_N \}$,

$\alpha_y = \{ \alpha_{y1}, \alpha_{y2}, \dots, \alpha_{yp} \}$ – a set of elementary formulas specifying the so called output system variables, $\alpha_{yi} \in \{ \alpha_1, \alpha_2, \dots, \alpha_N \}$,

$\alpha_w = \{ \alpha_{w1}, \alpha_{w2}, \dots, \alpha_{wr} \}$ – a set of auxiliary elementary formulas specifying the system functioning, $\alpha_{wi} \in \{ \alpha_1, \alpha_2, \dots, \alpha_N \}$.

Of course, $\alpha_x \cup \alpha_y \cup \alpha_w = \alpha$, $\alpha_x \cap \alpha_y = \emptyset$, $\alpha_x \cap \alpha_w = \emptyset$, $\alpha_y \cap \alpha_w = \emptyset$, and $a_x = \{ a_{x1}, a_{x2}, \dots, a_{xk} \}$, $a_y = \{ a_{y1}, a_{y2}, \dots, a_{yp} \}$, $a_w = \{ a_{w1}, a_{w2}, \dots, a_{wr} \}$, so $a = (a_{x1}, a_{x2}, \dots, a_{xk}) \wedge (a_{y1}, a_{y2}, \dots, a_{yp}) \wedge (a_{w1}, a_{w2}, \dots, a_{wr})$ corresponds to $\alpha = (\alpha_{x1}, \alpha_{x2}, \dots, \alpha_{xk}) \wedge (\alpha_{y1}, \alpha_{y2}, \dots, \alpha_{yp}) \wedge (\alpha_{w1}, \alpha_{w2}, \dots, \alpha_{wr})$.

$F_x(\alpha_x) = \{ F_{x1}(\alpha_x), F_{x2}(\alpha_x), \dots, F_{xr}(\alpha_x) \}$ – a set of input facts, i.e. assertions describing properties of system input,

$F_y(\alpha_y) = \{ F_{y1}(\alpha_y), F_{y2}(\alpha_y), \dots, F_{yr}(\alpha_y) \}$ – set of output facts, i.e. assertions describing properties of the system output.

Of course, besides of the real objects such as SMEs the above representation can be applied to any other objects, e.g., constraints, specifications, etc.

4. CONSTRAINT SATISFACTION PROBLEM REPRESENTATION

The declarative character of Constraint Programming languages and a high efficiency in solving combinatorial problems creates an attractive alternative for the currently available (employing operation research techniques) systems of computer-integrated management [2]. The Constraint Satisfaction Problem $CSP = ((X, D), C)$ consists of a set of variables $X = \{ x_1, x_2, \dots, x_n \}$, their domains $D = \{ D_i \mid D_i = [d_{i1}, d_{i2}, \dots, d_{ij}, \dots, d_{im}] \}$, $i = \{ 1, \dots, n \}$, and a set of constraints $C = \{ C_i \mid i = \{ 1, \dots, L \} \}$. A solution is such an assignment of the variable values that all the constraints are satisfied.

As the general CSP is NP-complete, constraint propagation is usually incomplete. This means that some but not all the consequences of the set of constraints are deducted. In particular, constraint propagation can not detect all inconsistencies. Consequently, one needs to perform some kind of search to determine if the CSP instance at hand has a solution or not. The solution of that problem can be the proper knowledge-based representation and decomposition of CSP . The CSP problem decomposes in natural way into subproblems, in particular to elementary subproblems, which are not further decomposed. The elementary problems can be seen as problems encompassed by constraints.

In the above considerations it should be noted that a $CSP = ((X, D), C)$ can be decomposed into a set of: elementary subproblems, loosely coupled subproblems, dependent subproblems of strongly coupled problems.

In general case any CSP may be decomposed (Fig. 2), either into a set of loosely coupled problems or into a set of strongly coupled problems. Possible ways of CSP decomposition enable taking into account the real life constraints following from: a way of a problem specification (i.e., a set of elementary problems recognized); a programming language implemented (some structures of dependent problems may or

may not be accepted by *CP/CCP/CLP* packages); a way of a *CSP* resolution (e.g., the loosely coupled subproblems can be computed independently within a multiprocessor environment); a searching strategy applied (the order of subproblems resolution results in a *CSP* makespan).

The above observation leads to a reference model concept of a *CSP* decomposition, [3]. So, since each subproblem corresponds to a standard constraint problem structure: (*a set of decision variables*), (*a set of variable domains*), (*a set of constraints*), hence some AND/OR – like graph representation can be used both in the analysis of the *CSP* programming and its resolution. It should be noted that any knowledge base can be represented in terms of $CSP = ((a,D), \{F(a)=1\})$, where $D = \{D_i \mid D_i = \{0,1\}, i = 1..N\}$, $F(a)=1$ a sequence of facts: ($F_1(a)=1, F_2(a)=1, \dots, F_K(a)=1$).

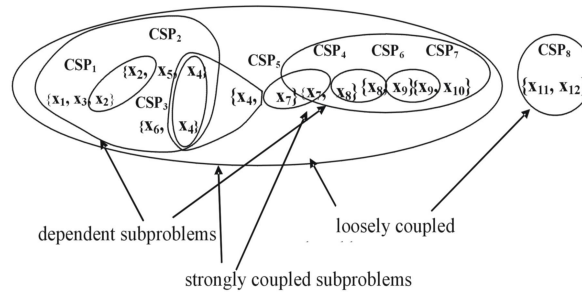


Figure 2 - $CSP = ((X,D),C)$ decomposition into loosely and strongly coupled as well as dependent subproblems

5. FEASIBLE SOLUTIONS

The question considered regards $F_x(\alpha_x)$ the following implication: $F_x(\alpha_x) \Rightarrow F_y(\alpha_y)$. In other words, the question is: What are α_x and $F_x(\alpha_x)$, if either, ensuring the system property $F_y(\alpha_y)$. Consider the knowledge base $RW' = \langle \alpha', F'(\alpha) \rangle$ corresponding to a system considered and the knowledge base stating a question $RW'' = \langle \alpha'', F''(\alpha) \rangle$, e.g. regarding a given system's property. The resultant knowledge base $RW = \langle \alpha, F(\alpha) \rangle$ (see Fig. 3) provides a framework for the considered problem statement: Is there an $F_x(\alpha_x)$, ensuring $F_y(\alpha_y)$? More precisely, in order to determine the feature $F_x(\alpha_x)$ a set of facts following feature $F_y(\alpha_y)$ while do not following $\neg F_y(\alpha_y)$ have to be determined. The scheme of the searching procedure is shown in Fig. 3. It means the knowledge base $RW1$ including the conditions implying $F_y(\alpha_y)$ as well as the knowledge base $RW2$ not including the conditions implying $F_y(\alpha_y)$ are refined from the knowledge base considered RW . The knowledge bases obtained enable to determine the final knowledge base $RW3$, i.e. modified $RW1$ (not including elementary formulas and facts included in $RW2$). In order to implement the above procedure in terms of logic-algebraic method the sets of binary values S_a, S_x , and S_y following a, a_x , and a_y , (while corresponding to $F(\alpha), F_x(\alpha_x)$, and $F_y(\alpha_y)$) have to be defined due to formulas: $S_a = \{a: F(a) = 1\}$, $S_x = \{a_x: F_x(a_x) = 1\}$, $S_y = \{a_y: F_y(a_y) = 1\}$.

Assumption that all the RW facts are true implies that among α sequences there are also sequences for which $F(a) = 1$ holds, see $S_a = \{a: F(a)=1\}$. The associated set S_a guaranteeing the facts describing the system are true can be treated as RW. Searching for the set S_x representing $F_x(\alpha_x)$ which can be treated as RW3 requires two subsets, i.e. S_{x1} corresponding to RW1 while following $F_y(a_y)=1$, and S_{x2} corresponding to RW2 while following $F_y(a_y)=0$.

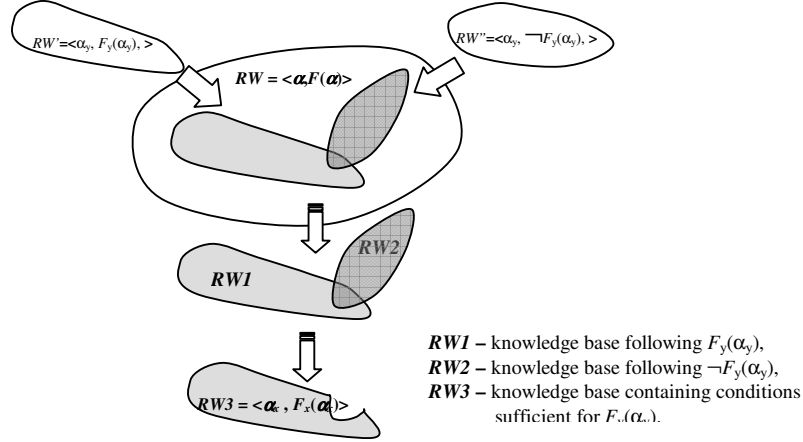


Figure3 - Sufficient conditions refinement

Finally, $S_x = S_{x1} \setminus S_{x2}$, where S_{x1} , and S_{x2} are determined for a_x from equations:

$$\text{for } S_{x1}: \begin{cases} F(a) = 1 \\ F_y(a_y) = 1 \end{cases} \tag{1}$$

$$\text{for } S_{x2}: \begin{cases} F(a) = 1 \\ F_y(a_y) = 0 \end{cases} \tag{2}$$

where: $F(a)=1$ stands for the set of facts: $\{F_1(a)=1, F_2(a)=1, \dots, F_K(a)=1\}$

6. METHODOLOGY FOR INTERACTIVE DEDICATED DECISION SUPPORT SYSTEM DESIGN

The proposed methodology consists of two stages (see Fig. 5). According to the first one the CLP_L including the sufficient conditions (i.e. guaranteeing a solution there exists) is provided. As the input data the SME based knowledge base KB_{SME} and the request based knowledge base KB_R are considered. Of course, the different request based knowledge bases may result in different sets of sufficient conditions. This observation provides a way of the sufficient conditions refinement, i.e. a way of DSS adjustment. The CLP_L extended for other kinds of variables and constraints (so called algebraic ones) results in CLP problem. So, due to the second stage a programming languages as well as decision variables substitution strategy

guaranteeing interactive usage of the *DSS* designed is provided [3]. Both stages are based on the *CP/CCP/CLP* languages. The key point of the methodology proposed regards the procedure for *CSP_L* design. In order to illustrate this procedure partially introduced in the Section 5, let us consider the following example.

Consider a controller composed of two switches P_1 and P_2 . The room's temperature is controlled by relevant set up switches

- If P_1 is turned on and P_2 is turned off, then the room is heated to 20 °C.
- If P_1 is turned off and P_2 is turned on, then the room is heated to 30 °C.
- If both P_1 and P_2 are turned on, then the room is heated to 40 °C.

The question is: What switches set up guarantee the room temperature between 20 and 40 °C?

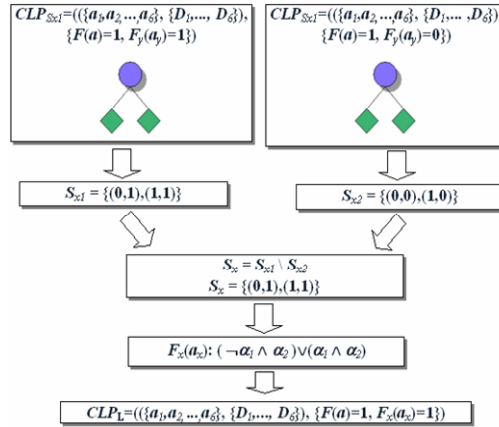
The controller's knowledge base representation is: $RW = \langle \alpha, F(\alpha) \rangle$:

$\alpha = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5)$, where: $F(\alpha) = (F_1(\alpha), F_2(\alpha), F_3(\alpha), F_4(\alpha))$, where:
 α_1 : „switch P_1 is turned on”, $F_1(\alpha): \alpha_1 \wedge (\neg \alpha_2) \Leftrightarrow \alpha_3$
 α_2 : „switch P_2 is turned off”, $F_2(\alpha): (\neg \alpha_1) \wedge \alpha_2 \Leftrightarrow \alpha_4$
 α_3 : „the room is heated to 20 °C”, $F_3(\alpha): \alpha_1 \wedge \alpha_2 \Leftrightarrow \alpha_5$
 α_4 : „the room is heated to 30 °C”, $F_4(\alpha): (\neg \alpha_1) \wedge (\neg \alpha_2) \Leftrightarrow \alpha_6$
 α_5 : „the room is heated to 40 °C”
 α_6 : „the room is not heated”

Elementary input and output formulas are: $\alpha_x = (\alpha_1, \alpha_2)$, $\alpha_y = (\alpha_3, \alpha_4, \alpha_5, \alpha_6)$

Required output property is: $F_1(\alpha_y) = \alpha_4 \vee \alpha_5$

The Fig. 4 illustrates the logic-algebraic method based procedure for the *CSP_L* design. Note that the procedure follows the scheme of sufficient conditions refinement shown in Fig. 3.



Legend:

$a_i = w(\alpha_i)$ – decision variable determining logic value of the formulae α_i ,

$D_i = \{0, 1\}$ – binary domain of the variable a_i ,

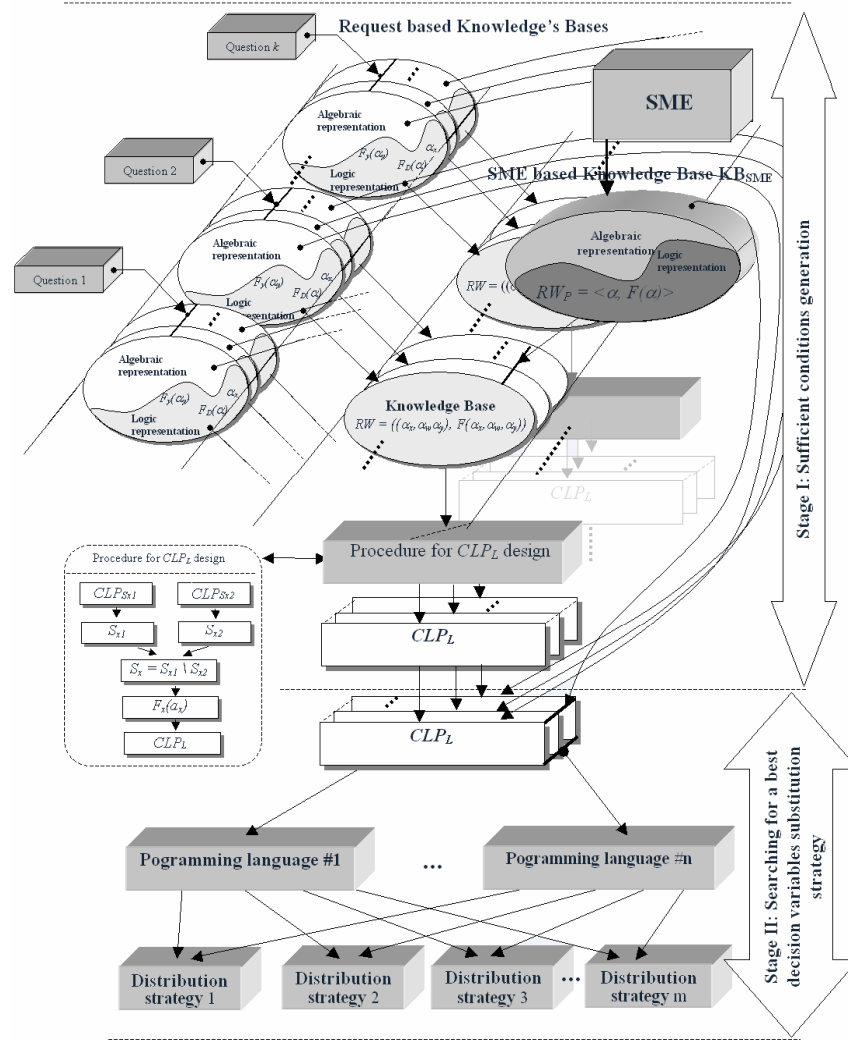
$F(a)=1$ – the constraint guaranteeing all facts F_i hold: $(F_1(a)=1, F_2(a)=1, F_3(a)=1, F_4(a)=1)$,

$F_y(a_y)=1$ – the constraint guaranteeing, the output fact is true,

$F_y(a_y)=0$ – the constraint guaranteeing, the output fact is false,

$F_x(a_x)$ – the input fact, i.e. resultant sufficient condition.

Figure 4 - Illustration of the logic-algebraic method procedure for the *CSP_L* design



Legend:

$a = a_1' a_2' \dots a_n'$, $a = (a_1, a_2, \dots, a_n)$, $a_i = v(\alpha_i)$, $D = \{0,1\}$, $C = \{C_P, C_D, C_W\}$, $C_P = \{C_1, C_2, \dots, C_k\}$, $C_D = \{C_{D1}, C_{D2}, \dots, C_{DH}\}$, $C_i = (F_i(a)=1)$, $C_D = (F_D(a)=1)$, $CLP_{S1} = ((a, D), \{C_P, C_D, F_i(a_i)=1\})$, $CLP_{S2} = ((a, D), \{C_P, C_D, F_j(a_j)=0\})$, $S = \{a_i \in a : F_i(a_i)=1\}$ - set of sequences a_i , following the constraint $F_i(a_i)=1$, $F_i(a_i)$ - sufficient conditions treated as an input fact, $C_W = (F_i(a_i)=1)$ - the constraint following from the sufficient conditions obtained, $CLP_L = ((a, D), \{C_P, C_D, C_W\})$ - problem specification including the sufficient conditions $CLP = ((x, D), \{C_P, C_D, C_W, C_N\})$ - problem specification including the sufficient conditions and extended for algebraic, ones. $x = x_N' a$ - decision variables vector, where x_N - belongs to the domain D_X of so called algebraic variables

Figure 5 - Methodology for interactive dedicated decision support systems design

7. CONCLUSIONS

A *CP/CCP/CLP* – based modelling framework driven by the logic-algebraic method provides a good platform for development of the task oriented *DSS*. The discussion provided has shown the versatility of *CP/CCP/CLP* paradigm for the decision making problems. Possible applications of logic-algebraic method to the examination of sufficient conditions ensuring assumed system's properties as well as the consistency checking techniques greatly reducing the search space and supported by *CP/CCP/CLP* prove their efficiency for resolution of the project-driven manufacturing tasks.

Therefore, the proposed approach can be considered as a contribution to design automation of interactive and task oriented *DSS*. That is especially important in the context of a cheap and user-friendly decision support for the *SME*'s.

In that context, the *CP/CCP/CLP* can be considered as a well-suited framework for development of decision-making software supporting small and medium size enterprises at the stage of production process planning [8].

Further research focuses on the development of the task oriented searching strategies, implementation of which could interface a decision maker with a user-friendly intelligent support system.

8. REFERENCES

1. Banaszak, Z., Zaremba, M., Muszyński, W., 2005, CP-based decision making for SME. Preprints of the 16th IFAC World Congress, 3 – 8 July, 2005, Prague, Czech Republic, Eds P. Horacek, M. Simandl, P.Zitek, DVD
2. Banaszak, Z., Zaremba, M., 2004, CLP-based project-driven manufacturing. Prep. of the 7th IFAC Symposium on Cost Oriented Automation, June 6-9, 2004, Gatineau, Quebec, Canada, pp. 269-274.
3. Banaszak, Z., Józefczyk, J., 2005, Towards CLP-based task oriented DSS for SME, Applied Computer Science and Production Management, Vol.1, No.1: 161-180.
4. Barták R., 2003, Constraint-based scheduling: An introduction for newcomers, Preprints of the 7th IFAC Workshop on Intelligent Manufacturing Systems, 6-8 April, 2003, Budapest, Hungary, pp. 75-80.
5. Mika, Marek; Waligóra G. Węglarz J., 2003, Metaheuristic approach to the multi-mode resource-constrained project scheduling problem with discounted cash flows and progress payment. In: Project-Driven Manufacturing. Banaszak Z., Józefowska J. Eds, WNT, Warsaw, pp. 34-47.
6. Rossi F., 2000, Constraint (Logic) programming: A Survey on Research and Applications, K.R. Apt et al. (Eds.), New Trends in Constraints, LNAI 1865, Springer-Verlag, Berlin, pp. 40-74.
7. Tomczuk I., Banaszak Z., 2004, Constraint programming approach for production flow planning, Proc. of the 6th Workshop on Constraint Programming for Decision and Control, pp. 47-54.
8. Van Hentenryck P., Perron L., Puget J., 2000, Search and Strategies in OPL, ACM Transactions on Computational Logic, Vol. 1, No. 2, pp. 1-36.
9. Van Hentenryck, P., Constraint Logic Programming, Knowledge Engineering Review, Vol. 6, 1991, pp. 151–194.
10. Wallace M., 2000, Constraint Logic Programming, Ed. Kakas A.C., Sadri F., Computat. Logic, LNAI 2407, Springer-Verlag, Berlin, Heidelberg, pp. 512-532.
11. Wallace M., Practical applications of constraint programming, Constraints Journal, Vol. 1(1), 1996, pp. 139-168.