# Hybrid Search Methods Based on Table Representation of Non-quantitative Constraints Satisfaction Problems

Alexander Zuenko[a]

[a]*Institute for Informatics and Mathematical Modeling, Subdivision of the Federal Research Centre "Kola Science Centre of the Russian Academy of Sciences", Apatity, Russia*

**Abstract**

The paper presents two forms of compressed table constraint representation: the *C*- and *D*-systems. The hybrid methods have been developed to solve non-quantitative constraint satisfaction problems formalized by table structures introduced. To make search more effective under the increase of the solution space, the hybrid methods integrate the original methods, such as non-quantitative constraint propagation and local search, as well as structural constraint graph decomposition methods.

## 1. Introduction

In constraint programming, the generally accepted search methodology is based on a joint application of constraint propagation methods and backtracking methods. In so doing, use is made of both specialized heuristics in selection of the variable and its value, and intelligent strategies of backtracking to the state that caused the illegal assignment. The peculiarity of backtracking methods is that these methods allow step-by-step extension of a partial allowable solution to a complete one. If a partial solution turns out to be illegal, there is backtracking and the previous partial solution is extended to an alternative direction. In many practical applications related to processing of a large body of information, methods based on the search tree studies only, are not sufficiently effective compared to the local search methods which have been put to the fore now, as well as the hybrid methods combining the advantages of several search schemes [Blu11, Fei18, Nat15, Lim12, Jeg17].

A very important type of so-called global constraints in constraint programming technologies is table constraints. A table form is very suitable for different non-numerical (qualitative) relations of the subject domain to be described. However, an increase in the size of the tables leads to an exponential increase in the complexity of the procedures used in tables processing. In this connection, of special actuality for table constraints are the development of the ways to compactly represent these in the computer memory, and the development of effective methods

*Russian Advances in Artificial Intelligence: selected contributions to the Russian Conference on Artificial intelligence (RCAI 2020), October 10-16, 2020, Moscow, Russia*

zuenko@iimm.ru (A. Zuenko)

of their processing.

There is a wide spectrum of studies dealing with compact representation of table constraints in the literature [Che10, Kat07, Ver17, Wan16]. Some studies deal with the constraint satisfaction algorithms presented by compressed tables [Che10]. The present paper is a study extending a series of the author's publications dealing with the properties of compressed tables and the development of methods of inference on compressed tables [Zue14, Zue17, Zue18a, Zue18b].

In order to make the representation and processing of table constraints more effective, the present study suggests two types of compressed tables: *C*-system and *D*-system. No work prototypes dealing with applications of the structures similar to *D*-systems in solving the constraint satisfaction problems have been found by the author.

An effective search in large-size compressed tables was first organized by specially developed hybrid methods that integrate new methods of local search, non-quantitative constraint propagation, systematic search, as well as widely used methods of structural constraint graph decomposition. All the methods developed are based on a detailed analysis of the internal structure of the table constraint types proposed.

The hybrid methods developed within the approach proposed are classified into two categories: 1) the methods realizing the structural constraint graph decomposition in conjunction with constraint propagation; 2) the methods realizing constraints propagation in conjunction with local search.

## 2. Compressed representation of table constraints

For it to be solved by the constraint programming technology, any applied problem should be represented as a constraint satisfaction problem.

According to [Rus10], the *Constraint Satisfaction Problem* (CSP) consists of three components: <$X$, $D$, $C$>. $X$ – is a set of variables $\{X_1, X_2, ..., X_n\}$. $D$ – is a set of domains $\{D_1, D_2, ..., D_n\}$, where $D_i$ – is a variable domain $X_i$. $C$ – is a set of constraints $\{C_1, C_2, ..., C_m\}$, which specify the allowed combinations of variable values. Each domain $D_i$ defines a set of a values $\{v_1, ..., v_k\}$ for variable $X_i$. Each constraint is a couple <$scope$, $rel$>, where $scope$ – is a set of variables taking part in the constraint, $rel$ – is a relation specifying the allowed combinations of values which can be taken by the variables from $scope$.

The assignment which does not violate any constraint, is called a *legal* assignment. A *complete* assignment is that in which every variable participates. The *solution* of the CSP is a complete assignment satisfying all the constraints.

The constraints can be represented either by explicit enumeration of all the allowed combinations of values for the stated set of variables (an extensional way of specifying) or in a kind of an abstract relation supporting two operations, such as to test if a tuple is an element of the relation given, and to enumerate all the elements of the relation (an intentional way of specifying). Tables in relational databases can be an example of extensional relations. These relations play a very important role in the fields of CAD (computer aided design systems) and GIS (geographic information systems). The constraint satisfaction algorithms supporting extensional relations (table constraints) are widely applied in the fields mentioned above. Speaking about intentional relations we mean the relations specified parametrically. These are practically all

the relations over quantitative (continuous) domains.

Very often, $n$-ary relations specified extensionally, can be represented more compactly if compared to a full enumerating of their tuples.

Brought to attention is a mathematical apparatus for a "compressed" representation of $n$-ary relations, which uses two types of matrix-like structures. The first type is the $C$-systems. The $C$-system is a compressed table in which sets rather than separate elements act as cells.

Let's consider an example. Let there be the following relation: $\{(c, 1), (c, 2), (c, 4), (c, 5), (b, 2), (b, 4), (d, 1), (d, 5)\}$ on variables $X$, $Y$ with domains: $X - \{a, b, c, d\}$, $Y - \{1, 2, 3, 4, 5\}$. Presented below are a $C$-system, which compactly describes the relation, and an algebraic expression over sets that corresponds to a given $C$-system:

$$T[XY] = \begin{bmatrix} \{c\} & \{1,2,4,5\} \\ \{b\} & \{2,4\} \\ \{d\} & \{1,5\} \end{bmatrix} = \{c\} \times \{1,2,4,5\} \cup \{b\} \times \{2,4\} \cup \{d\} \times \{1,5\}.$$

Graphically, each row of the $C$-system corresponds to a subspace in the attribute space (Cartesian products of sets) and the whole $C$-system is a union of these subspaces.

The other type of compressed tables, which provides a compact representation of $n$-ary relations, is the $D$-system. This matrix is represented in reversed direct brackets. Given below is the $D$-system $P[XY]$, which is equivalent to the $C$-system $T[XY]$ because both systems describe the same initial relation:

$$P[XY] = \left] \begin{matrix} \{c,d\} & \{2,4\} \\ \{b,c\} & \{1,5\} \\ \emptyset & \{1,2,4,5\} \end{matrix} \right[ = (\{c,d\} \times \{1,2,3,4,5\} \cup \{a,b,c,d\} \times \{2,4\}) \cap$$
$$\cap(\{b,c\} \times \{1,2,3,4,5\} \cup \{a,b,c,d\} \times \{1,5\}) \cap (\{a,b,c,d\} \times \{1,2,4,5\}).$$

The $D$-system is a complex algebraic expression. Each row of the $D$-system is a subspace in the attribute space, which is more complicated in structure than the Cartesian product of sets. Each row of the $D$-system is a union of definite Cartesian products. The $D$-system specifies the intersection of subspaces correlated to each of its rows.

An empty component ("$\emptyset$" in the description of $C$- and $D$-systems) is a dummy component that does not contain values. Another dummy component is the full component – "$*$" – a designation for the entire range of possible values (domain) of an attribute.

As in the case of conventional tables, relational algebra operations can be applied to the $C$- and $D$-systems. These operations are implemented without decomposition of the $C$- and $D$-systems into conventional tables, using specialized theorems. In addition, the operation of complement of $n$-ary relations is widely applied to the $C$- and $D$-systems, which in fact is not used in the relational databases. More details about operations with $C$- and $D$-systems can be found in [Kul15]. Represented here is the theorem which is to be used in the presentation below.

**Theorem 1.** *The result of join of two C-systems can be presented as a C-system composed of rows produced from joining of each vector-row of the first C-system with each vector-row of the second C-system.*

Now, change to the description of the hybrid search methods for the proposed types of compressed tables.

# 3. Structural decomposition of the constraint graph and constraint propagation

We have developed hybrid method that integrates the author's methods of non-quantitative constraints propagation [Zue17, Zue18a] and the known methods of structural decomposition of the constraint graph [Dec89, Mig01]. It is known that if the graph of the constraint satisfaction problem (sub-problem) has a tree structure, then, in order to reach its solution, it is enough to apply only the constraint propagation methods.

The peculiarity of the method developed is in that a set of solutions of each sub-problem are being reached in a form of a $C$-system, i.e., sub-spaces are being found in the space of legal assignments per algorithm step, which substantially reduces the time necessary to carry out computational procedures.

The method proposed consists of three basic stages:

1. The graph of constraints of the initial problem is partitioned into loosely-coupled or independent sub-problems, the graphs of constraints of which are trees. At this stage well known methods of structural decomposition of the constraint graph are applied (for instance, the tree decomposition or cycle cutset methods).
2. Each sub-problem is solved by the well known algorithm applied in solving the constraint satisfaction problem with a tree structure and by the author's methods of non-quantitative constraint propagation. The solution of each sub-problem is being reached in a form of the $C$-system.
3. Based on Theorem 1, the sub-problems solution are joined into a solution of a common problem defined also by the $C$-system.

The representation of a set of solutions as the $C$-system substantially economizes the memory resources and accelerates computational procedures in solution searching in comparison with explicit enumeration of all the possible solutions. At the same time, it is not difficult to explicitly enumerate all the elementary solutions on the basis of their representation in a form of the $C$-system because the $C$-system possesses a simple characteristic function.

Thus, the method developed integrates two basic components: a) the component realizing the algorithms of propagation; b) the component realizing the structural decomposition of the initial problem. Considered below are the principles of these two components functioning.

## 3.1. Structural decomposition algorithms

Consider the basic approaches which the structural algorithms are based on. To put it another way, let's define the ways allowing us to use the structure of the problem, which is represented as a constraint graph, to accelerate the search of the solution.

The structure or the topology of CSPs can be defined by different graph structures: (primary) constraint graph, constraint hypergraph, dual constraint graph.

The primal constraint graph of the CSP $(V, D, C)$ – is a non-oriented graph $G = (V, E)$, the nodes $V$ of which correspond to the variables of the CSP, with two nodes being combined by the edge in graph $G$ if the corresponding variables participate in the same constraint. The primary graph will be taken as the constraint graph hereinafter the paper.

The constraint graph can be used to partition the problem into a complex of simpler, in terms of computational complexity, sub-problems. If the CSP composition allows us to distinguish sub-problems fully independent of each other, it is possible to solve these separately, and the solutions can be combined into a solution of the initial problem.

To partition the CSPs into independent sub-problems, we can consider the connected components of the constraint graph. Suppose that each sub-problem of the CSP contains $c$ variables of the total number of variables ($n$). In this case, the number of sub-problems is $n/c$, and to solve each sub-problem, the volume of work is $d^c$, where $d$ is the size of the variables domain. Thus, the total volume of work is measured by the value $O(d^c \times n/c)$, which linearly depends on $n$; if the decomposition were absent, the total volume of work would be measured by the value $O(d^n)$, which exponentially depends on $n$. So, the totally independent sub-problems are very attractive. However these are rather rare.

In most cases the sub-problems of any CSP are connected with each other by variables. In the simplest case, the constraint graph is a tree: any two variables are connected not more than by one way. The algorithms of a solution of a CSP having a tree structure, possess low computational complexity (CSP can be solved in linear time) [Mig01].

As there is an effective algorithm for trees, one should consider the question if there is any possible way to reduce more common constraint graphs to tree structures.

There are two basic ways of solving this problem: one of these is based on eliminating the nodes, the other is based on merging the nodes with each other and on forming the super-nodes. The first approach envisages the assignment of values to some variables so that the variables left form a tree. The first approach is based on the well known methods, such as the cycle cutset method [Dec90] and the method of detecting the connected components of the constraint graph [Dec87, Fre82]. The typical methods implemented in the second approach include the method of tree decomposition [Dec89, Dec87, Fre85] and the method of tree-clustering [Dec89].

Structural decomposition methods are also effectively applied in solutions of CSPs that contain non-binary constraints, particularly, global constraints [Tho16].

The methods of structural decomposition mentioned above possess the following common features. Each sub-problem of a common CSP is solved independently: if any of these has no solution, the whole problem has no solution either. If all the sub-problems are solved, there is an attempt to reach a global solution.

### 3.2. Methods for propagation of constraints represented as $C$-systems

Given below are the statements allowing realization of the equivalent transformations of the constraints for the case when constraints are represented as a set of the $C$-systems [Zue17]. The aim of the transformations is to bring the CSP to a simpler form, which contains a lesser number of the $C$-systems, rows of the $C$-systems, columns (attributes) of the $C$-systems, values in the attributes domains etc.

*Statement 1* (***S1***). If all the rows (tuples) of the $C$-system are empty, that is, each row contains at least one empty component, the $C$-system is empty (the corresponding CSP is inconsistent).

*Statement 2* (***S2***). If all the components of an attribute (a column of the $C$-system) are full, this attribute can be eliminated from the $C$-system (all the components in the corresponding column are eliminated), and a couple "the eliminated attribute – its domain" is preserved in the

vector of the partial solution.

*Statement 3* (***S3***). If the domain of an attribute of the $C$-system contains values not occurred in the corresponding column, these values are eliminated from the given domain.

*Statement 4* (***S4***). If the row of the $C$-system contains at least one empty component (the row is empty), the row is eliminated.

*Statement 5* (***S5***). If the component of an attribute contains the value not belonging to the corresponding domain, this value is eliminated from the component.

*Statement 6* (***S6***). If one row of the $C$-system completely dominates (contains component-wise) the other row, the dominated row is eliminated from the $C$-system.

A part of the statements makes it possible to eliminate the values from the domains and the components of the attributes (***S3, S5***) or the columns-attributes (***S2***), and a part of them makes it possible to eliminate extra rows (***S4, S6***) from consideration. The indicator of successful completion of searching is the elimination of all the rows and columns from the $C$-system, with no empty rows formed. To put it another way, in this case, the resulting state is characterized only by a complex of non-empty reduced domains in the tuple of the solution. The indicator of inconsistency of the CSP is the emptiness of the $C$-system (***S1***). The similar statements for the $D$-system are given below.

### 3.3. An example of the method realization

Let the following correspondences "attribute – attribute domain" be specified: $A$–$\{a_1,a_2,a_3,a_4,a_5,a_6\}$, $B$ – $\{b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9\}$, $C$ – $\{c_1, c_2, c_3, c_4, c_5, c_6\}$, $D$ – $\{d_1, d_2, d_3, d_4, d_5, d_6\}$, $E$ – $\{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\}$, $F$ – $\{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9, f_{10}, f_{11}, f_{12}, f_{13}, f_{14}\}$. Also specified is the following set of constraints written in a form of the $C$-systems (their semantics is not essential in the example):

$$Constraint\ R_1[AB]\quad \begin{bmatrix} \{a_1\} & \{b_1, b_2\} \\ \{a_2\} & \{b_3, b_4, b_5\} \\ \{a_3\} & \{b_6\} \\ \{a_4\} & \{b_7\} \\ \{a_5\} & \{b_8\} \\ \{a_6\} & \{b_9\} \end{bmatrix}$$

$$Constraint\ R_2[BC]\quad \begin{bmatrix} \{b_1\} & \{c_1\} \\ \{b_2, b_3, b_6\} & \{c_2\} \\ \{b_4\} & \{c_3\} \\ \{b_5, b_7\} & \{c_4\} \\ \{b_8\} & \{c_5\} \\ \{b_9\} & \{c_6\} \end{bmatrix}$$

$$Constraint\ R_3[DC]\quad \begin{bmatrix} \{d_1\} & \{c_1\} \\ \{d_2\} & \{c_2\} \\ \{d_3\} & \{c_3\} \\ \{d_4\} & \{c_4\} \\ \{d_5\} & \{c_5\} \\ \{d_6\} & \{c_6\} \end{bmatrix}$$

$$Constraint\ R_4[ED]\quad \begin{bmatrix} \{e_1\} & \{d_1\} \\ \{e_2\} & \{d_2, d_3\} \\ \{e_3\} & \{d_2\} \\ \{e_6\} & \{d_4\} \\ \{e_7\} & \{d_5, d_6\} \end{bmatrix}$$

$$Constraint\ R_5[CF]\quad \begin{bmatrix} \{c_1\} & \{f_1, f_2\} \\ \{c_2\} & \{f_3, f_4, f_5\} \\ \{c_3\} & \{f_6, f_7\} \\ \{c_4\} & \{f_8, f_9, f_{10}\} \\ \{c_5\} & \{f_{11}, f_{12}\} \\ \{c_6\} & \{f_{13}, f_{14}\} \end{bmatrix}$$

$$Constraint\ R_6[FE]\quad \begin{bmatrix} \{f_1, f_6\} & \{e_1\} \\ \{f_2, f_3, f_4\} & \{e_2\} \\ \{f_4, f_6\} & \{e_3\} \\ \{f_5, f_7\} & \{e_4\} \\ \{f_8, f_9, f_{11}\} & \{e_5\} \\ \{f_9, f_{11}, f_{14}\} & \{e_6\} \\ \{f_{10}, f_{13}\} & \{e_7\} \\ \{f_{12}, f_{14}\} & \{e_8\} \end{bmatrix}$$

The task is to calculate the join of the present relations, i.e., it is necessary to find: $R_1[AB] \bowtie R_2[BC] \bowtie R_3[DC] \bowtie R_4[ED] \bowtie R_5[CF] \bowtie R_6[FE]$.

Figure 1 shows the graph of the CSP. Figure 2 shows an example of the structural decomposition of the graph of the initial CSP, which has been considered earlier (Figure 1). The initial CSP is partitioned into sub-problems by the assignment of the values to the variable $C$.

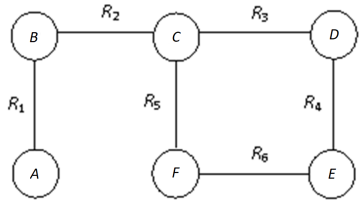


**Figure 1:** The graph of the initial CSP

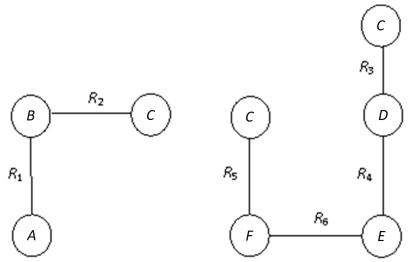


**Figure 2:** The effect of the structural decomposition after the assignment of the values to the variable $C$

In this case, in order to solve the initial problem, it is necessary to consider two sub-problems:

1. $R_2[BC] \bowtie R_1[AB]$;
2. $R_3[DC] \bowtie R_4[ED] \bowtie R_6[FE] \bowtie R_5[CF]$.

It is possible, in advance, to calculate the corresponding multi-place relation for each sub-problem of the CSP, which results from the process of structural decomposition. Then, these relations can be joined into one.

Let's suppose that the value $c_1$ is assigned to the variable $C$, then the domain of the variable $C$ becomes a singleton set $\{c_1\}$.

Let's consider the first sub-problem of CSP ($R_2[BC] \bowtie R_1[AB]$). The graph of the sub-problem is a tree, hence, the solution of the sub-problem can be reached using only the methods of non-quantitative constraint propagation developed earlier, not using the backtracking methods. First, constraint $R_2[BC]$ is activated and, based on Statements ***S1-S5***, "tuned" to a new domain of variable $C$. After ***S5*** being applied, all the rows, but the first one, of the initial $C$-system $R_2[BC]$ are empty. Then, according to ***S4***, constraint $R_2[BC]$ takes the form of $[\{b_1\}\ \{c_1\}]$. Now, according to ***S3***, we may conclude that a new domain of variable $B$ is singleton $\{b_1\}$. Then

constraint $R_1[AB]$ is activated and, being "tuned" to a new domain of variable $B$, takes the form of $[\{a_1\}\{b_1\}]$. So, we have finished moving from constraint $R_2[BC]$ to constraint $R_1[AB]$ .

The movement in the opposite direction can be represented as a result of "reduced" constraints joining. We have the following vector-row in the scheme $[ABC]$: $[\{a_1\}\{b_1\}\{c_1\}]$.

Now, we still think that variable $C$ takes the value $c_1$, and will consider the second subproblem: $R_3[DC] \bowtie R_4[ED] \bowtie R_6[FE] \bowtie R_5[CF]$.

Considering constraint $R_3[DC]$ and taking into account a new domain of variable $C$, we conclude that the constraint is converted into vector-row $[\{d_1\}\{c_1\}]$, and the domain of variable $D$ is reduced to a singleton set $\{d_1\}$. Being activated, constraint $R_4[ED]$, is converted into vector-row $[\{e_1\}\{d_1\}]$, and the domain of variable $E$ is reduced to $\{e_1\}$. After that constraint $R_6[FE]$ is analyzed, being reduced to vector-row $[\{f_1, f_6\}\{e_1\}]$. The domain of variable $F$ is reduced to $\{f_1, f_6\}$. Finally, constraint $R_5[CF]$ is activated, which, taking into account all the specified domains, is converted to vector-row $[\{c_1\}\{f_1\}]$.

Moving the opposite direction results in a vector-row in the scheme $[CDEF]$ : $[\{c_1\}\{d_1\}\{e_1\}\{f_1, f_6\}]$. Thus, we have implemented an elementary algorithm step. All the rest steps for the five assignments left are made in the same way. The results of all the steps can be classified into two $C$-systems.

| | $A$ | $B$ | $C$ |
|---|---|---|---|
| 1 | $\{a_1\}$ | $\{b_1\}$ | $\{c_1\}$ |
| 2 | $\{a_1\}$ | $\{b_2\}$ | $\{c_2\}$ |
| 3 | $\{a_2\}$ | $\{b_3\}$ | $\{c_2\}$ |
| 4 | $\{a_3\}$ | $\{b_6\}$ | $\{c_2\}$ |
| 5 | $\{a_2\}$ | $\{b_4\}$ | $\{c_3\}$ |
| 6 | $\{a_2\}$ | $\{b_5\}$ | $\{c_4\}$ |
| 7 | $\{a_4\}$ | $\{b_7\}$ | $\{c_4\}$ |
| 8 | $\{a_5\}$ | $\{b_8\}$ | $\{c_5\}$ |
| 9 | $\{a_6\}$ | $\{b_9\}$ | $\{c_6\}$ |

,

| | $C$ | $D$ | $E$ | $F$ |
|---|---|---|---|---|
| 1 | $\{c_1\}$ | $\{d_1\}$ | $\{e_1\}$ | $\{f_1\}$ |
| 2 | $\{c_2\}$ | $\{d_2\}$ | $\{e_2\}$ | $\{f_3, f_4\}$ |
| 3 | $\{c_2\}$ | $\{d_2\}$ | $\{e_3\}$ | $\{f_4\}$ |
| 4 | $\{c_4\}$ | $\{d_4\}$ | $\{e_6\}$ | $\{f_9\}$ |
| 5 | $\{c_6\}$ | $\{d_6\}$ | $\{e_7\}$ | $\{f_{13}\}$ |

Of all the $C$-systems represented, the first one (from left to right) describes a set of solutions for the first sub-problem of the CSP, the second one – for the second sub-problem of the CSP.

Now the two $C$-systems can be joined (by the common variable $C$) to obtain the resulting $C$-system describing all the solutions of the initial problem. The two $C$-systems are being joined for polynomial time according to Theorem 1.

Next, we consider the second of the developed hybrid methods.

## 4. Constraint propagation and local search

Constraint propagation methods and local search methods are known to possess low computational complexity. Due to it, of promising perspective is the integration of methods of these classes into hybrids. However, it is known that methods of constraint propagation use partial assignments (only some variables are instantiated) in order, based on specified values of some variables, to make conclusions about reducing the domains of the variables left, and methods

of local search use complete assignments (all the variables take values) which either are taken as allowed or are modified in a special way to produce allowed assignments.

The paper proposes a hybrid method integrating the following components:

1. the author's methods of non-quantitative constraint propagation (methods of inference on the *D*-systems) to reduce the solution space;
2. the method of local search for partial assignments, which is based on application of heuristics with minimum conflicts for quick transition from elimination of sub-spaces containing no solution;
3. Tabu search [Ste97] to avoid repeated analyzing states already studied.

### 4.1. Inference on the *D*-systems

We state, without proving, the statements used to organize inference on the *D*-systems [Zue14]:

*Statement 1'* (***S1'***). If at least one row of the *D*-system is empty (contains all the empty components), the *D*-system is empty (the corresponding system of constraints is inconsistent, the CSP has no solution).

*Statement 2'* (***S2'***). If all the components of an attribute are empty, the attribute can be eliminated from the *D*-system (all the components in the corresponding column are eliminated).

*Statement 3'* (***S3'***). If in the *D*-system there is a row (tuple) containing exactly one non-empty component, all the values not included into the component, are eliminated from the corresponding domain.

*Statement 4'* (***S4'***). If a row of the *D*-system contains at least one full component, it is eliminated (the corresponding constraint can be eliminated from the constraint system).

*Statement 5'* (***S5'***). If a component of an attribute of the *D*-system contains a value not belonging to the corresponding domain, the value is eliminated from the component.

Statements 1'-5' allow elimination of all the "extra" values from some components, from domains of variables (attributes), elimination of rows and/or columns of the constraint matrices, reducing the search space and accelerating reaching the solution of the CSP. Based on the statements above, the well known arc and node consistency algorithms were modified for cases of non-quantitative constraints [Zue18a].

### 4.2. Local search based on the analysis of compressed tables

The local search methods are the basic methods used in solution of complex CSPs. The idea is to start the local search from choosing a randomly generated candidate in the solution of the problem (complete assignment) and improve it, step by step, for example, by reducing the number of unsatisfied constraints (conflicts). The local search algorithms differ in methods of finding this improvement. One of the disadvantages of local algorithms is their incompleteness, i.e. the search may stop at a local optimum, which in fact is not a global solution.

The paper [Zue18b] presents an approach to organize the local search process for the solution of the CSPs, which is based on a compressed representation of the qualitative constraints in a form of the *D*-systems and on an application of the statements about revealing the sub-spaces in the spaces legal and/or illegal assignments.

In organizing the local search procedures on the $D$-systems, a notion of a conflict is interpreted as follows: a conflict means a situation when in the process of assignment of the values to all the variables of the $D$-system there is an empty row formed in the $D$-system. An empty row is a row completely composed of empty components. In terms of the systematic search methods, the presence of at least one empty row means that the constraint is inconsistent and backtracking is necessary. As part of the proposed approach to organizing local search, it is necessary to calculate the number of the conflicts so that to make a decision about further progress to neighboring assignments.

Now, let's clarify a notion of a neighboring state. A neighboring state is a complete assignment which differs from the current one only in the value of one variable.

The local search process is reduced to the change from the current state to the neighboring one until the constraint ($D$-system) is satisfied or until the number of trials is exhausted. Of all the possible neighboring states, the one allowing to reach the solution by the fastest way, is chosen. To do that, it is recommended to apply the following heuristics. Of all the variables of the conflict set, it is recommended to choose a one which participates in the greatest quantity of conflicts. The variable is included into variables of the conflict set if the assignment in this attribute is one of the reasons of the formation of at least one empty row. It is recommended to choose the value the most often met in the corresponding column.

The paper [Zue18b] also deals with the way of accelerating the local search methods on the compressed tables proposed. This acceleration is based on applying the consequences of the theorem presented below.

The theorem is based on the partial order relations, which are determined for the values inside the domain of each attribute. So let's first remind that, if in the $D$-system, with the scheme $S$ and a set of the numbers of tuples, one chooses an attribute $X$ ($X$ is included into the scheme $S$) with the domain $D_X$, then, for the values from $D_X$, the partial order relation "$\preceq$" is introduced as follows: $a \preceq b$, if and only if $\{n_o\} \subseteq \{n_p\}$, where: $a, b \in D_X$; and $\{n_o\}$ and $\{n_p\}$ are the sets of the numbers of the rows of the $D$-system whose components of the attribute $X$ contain the values $a$ and $b$, respectively.

To put it another way, one value dominates the other within one domain if it occurs in the given column in all the rows where the first one is found, and in some other rows. Figure 3 shows the Hasse diagrams for the values of the attributes $X$ and $Y$ of a $D$-system.

Thus for the attribute $Y$ the value $g$ dominates the value $e$ (i.e., $g \succeq e$), as it, like $e$, is present in row 3 in the second column but is present in row 1 in the second column where $e$ is absent. As a whole, as for the attribute $Y$, we have $e, f, h \preceq g$. For attribute $X$ : $d \preceq a, b$.

**Theorem 2.** If in the $D$-system with the scheme $S$ for an attribute $X$ ($X$ belongs to the scheme $S$) with the domain $D_X$ there is: $a \preceq b$, then for any couple of complete assignments $\{(X_1, c_1), ..., (X_{p-1}, c_{p-1}), (X_p, a), (X_{p+1}, c_{p+1}), ..., (X_n, c_n)\}$ and $\{(X_1, c_1), ..., (X_{p-1}, c_{p-1}), (X_p, b), (X_{p+1}, c_{p+1}), ..., (X_n, c_n)\}$ the number of conflicts generated by the second assignment is not greater than that generated by the first assignment.

*Consequence 2.1.* If a value $b$ of the attribute $X_p$, belongs to the illegal assignment, then, in replacing the value $b$ to some other value $a$, with the value $a \preceq b$, we also obtain an illegal assignment.

*Consequence 2.2.* If a value $a$ of the attribute $X_p$, belongs to the legal assignment, then, in replacing the value $a$ to some other value $b$, with the value $a \preceq b$, we also obtain a legal

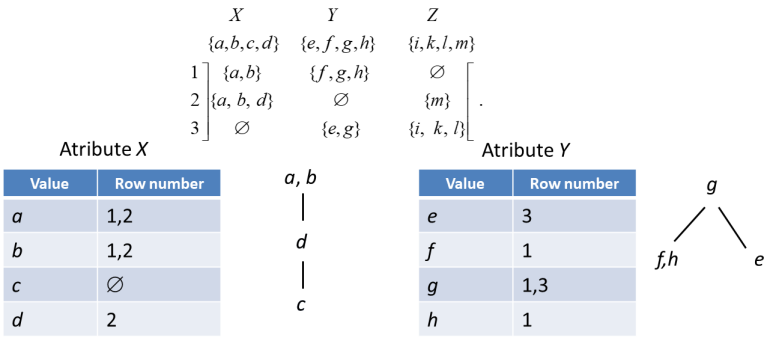


**Figure 3:** The partial order relation on the values of the attributes domains of the initial $D$-system

assignment.

Considered below are the typical cases when the theorem and the consequences mentioned above are capable to accelerate searching.

Firstly, it is possible to generate the assignments containing dominating values in each attribute. Applied to our example, the following assignment: $\{(X, a), (Y, g), (Z, l)\}$ is generated. It satisfies the $D$-system shown in Figure 3, so it is the solution.

Secondly, if a solution is reached, then, based on the *Consequence 2.2*, it is possible to extend it to a sub-space in the legal assignments space. If the assignment $\{(X, a), (Y, g), (Z, l)\}$ is the solution, then, taking into account the partial order relations in the variables domains, we obtain a sub-space of legal assignments $X - \{a, b\}$, $Y - \{g\}$, $Z - \{i, k, l\}$. In particular, one of the solutions derived is $\{(X, b), (Y, g), (Z, l)\}$.

Thirdly, if there are conflicts in the current state, then, in changing to the neighboring state in the given conflict attribute, the priority should be given to one of the dominating values allowable at the stage of a choice. The heuristics proposed earlier and dictating to choose the values that are found in columns more often, is a peculiar approximation of the given rule.

Finally, if an illegal assignment is found, it can be extended, based on *Consequence 2.1*, to an illegal assignments sub-space. Let an illegal assignment $\{(X, c), (Y, g), (Z, l)\}$ be found at the current stage, which generates a conflict in the second row. Then, taking into account the partial order relations in the variables domains, we have the following sub-space in the illegal assignment space: $X - \{c\}$, $Y - \{e, f, h, g\}$, $Z - \{i, k, l\}$.

## 4.3. The scheme of the hybrid method based on the local search

The hybrid method developed is based on the constraint satisfaction problem representation as the $D$-system, using the conflict calculation heuristics to modify the current inconsistent partial assignment. It is suggested to model the assignment vector as the $C$-type vector-row whose components can contain not only a particular value but a set of allowed values of the given variable. At the stage of extending the current partial assignment, a variable with minimum domain is chosen, and then a choice is made of the variable value that is more often found in the corresponding column. In fact, the method developed is a modification of the well known tabu

decision-repair method [Jus02, Cha04] for the problem of processing the compressed tables proposed.

The scheme of the method is as follows:

1. A partial order is designated for the domain of each variable on the values of variables (like in the local search algorithm proposed above).
2. Partial assignments are extended iteratively until a) an illegal assignment is found (change to step 3 of the algorithm); b) an legal assignment is found (change to step 4) or c) the number of trials is exhausted (change to step 5). In so doing, use is made of the following heuristics to choose a variable and the values of the variable: a variable with minimum domain is chosen, then, a choice is made of the variable value which is the most often found in the corresponding column. In the course of carrying out assignment, the $D$-system is reduced in accordance with the reduction rules proposed earlier.
3. In case an inconsistent partial (or complete) assignment is found, the local search procedure is carried out instead of backtracking. The inconsistent partial assignment based on *Consequence 2.1*, is formed up to the sub-space in the illegal partial assignments space for the variable set specified. The found conflicts (assignments resulting in empty rows formation in the $D$-system) are recorded in special memory formed on a queue principle. It is expedient to limit the queue by a small number of elements.
   Based on the quantity of conflicts (empty rows of the $D$-system), choice is made of the variable participating in the maximum number of conflicts. The partial assignment is modified by calculation of the addition to the partial assignment component, which corresponds to the conflict variable chosen. Change to step 2.
4. The legal assignment is extended, based on *Consequence 2.2*, to the domain in the legal assignments space.
5. The end.

The method proposed has been successfully utilized in solving a problem of planning of localizing the consequences of on-land oil products spills at one of the subdivisions of the JSC "Apatit", the Kirovsk Branch of FosAgro.

Consider the basic scheme of the method proposed (with no additional possibilities for acceleration, i.e., taking no account of the relations of domination), with an "$N$-Queens" problem taken as an example.

## 4.4. An example of the hybrid search method application

In our case, the chessboard is $4 \times 4$ in size. The problem is to find one of the possible variants of four queens arrangement on the chessboard so that they are not threatened by each other. The similar formulation of the problem has been considered earlier in [Zue18c], where we proposed to formalize the problem by using the $D$-system and made an assessment of the backtracking methods developed by the authors. In the present study the "$N$-Queens" problem is solved by a new hybrid method.

Let's match variable $X_i$ with the $i$-th horizontal. In this case the domain of each variable (attribute) is $\{a, b, c, d\}$. Now let's set a constraint "two queens arranged on horizontals 1 and

2, are not threatened by each other" in the form of the $D$-system:

$$\begin{array}{c|cc} & X_1 & X_2 \\ & \{a,b,c,d\} & \{a,b,c,d\} \\ \hline 1 & \{b,c,d\} & \{c,d\} \\ 2 & \{a,c,d\} & \{d\} \\ 3 & \{a,b,d\} & \{a\} \\ 4 & \{a,b,c\} & \{a,b\} \end{array}$$

All the names of variables and their current domains are enumerated in the title of the $D$-system.

The first row of the $D$-system, particularly, formalizes that if a queen is on field $a_1$ (the place where the first horizontal and the first vertical are intersected), then, on the second horizontal, the other queen may occupy only fields $c_2$ or $d_2$. In the language of logic, it is expressed as follows:

$$\begin{array}{l} (X_1 \in \{a\}) \longrightarrow (X_2 \in \{c,d\}) \\ \text{or } \neg(X_1 \in \{a\}) \vee (X_2 \in \{c,d\}) \\ \text{or } (X_1 \in \{b,c,d\}) \vee (X_2 \in \{c,d\}). \end{array}$$

If we compare different pairs of horizontals, we can write down all the constraints on the inter-relative arrangement of all the 4 queens. Thus the CSP described can be expressed in the

form of the $D$-system:

| | $X_1$ | $X_2$ | $X_3$ | $X_4$ |
|---|---|---|---|---|
| | $\{a, b, c, d\}$ | $\{a, b, c, d\}$ | $\{a, b, c, d\}$ | $\{a, b, c, d\}$ |
| 1 | $\{b, c, d\}$ | $\{c, d\}$ | $\emptyset$ | $\emptyset$ |
| 2 | $\{a, c, d\}$ | $\{d\}$ | $\emptyset$ | $\emptyset$ |
| 3 | $\{a, b, d\}$ | $\{a\}$ | $\emptyset$ | $\emptyset$ |
| 4 | $\{a, b, c\}$ | $\{a, b\}$ | $\emptyset$ | $\emptyset$ |
| 5 | $\emptyset$ | $\{b, c, d\}$ | $\{c, d\}$ | $\emptyset$ |
| 6 | $\emptyset$ | $\{a, c, d\}$ | $\{d\}$ | $\emptyset$ |
| 7 | $\emptyset$ | $\{a, b, d\}$ | $\{a\}$ | $\emptyset$ |
| 8 | $\emptyset$ | $\{a, b, c\}$ | $\{a, b\}$ | $\emptyset$ |
| 9 | $\emptyset$ | $\emptyset$ | $\{b, c, d\}$ | $\{c, d\}$ |
| 10 | $\emptyset$ | $\emptyset$ | $\{a, c, d\}$ | $\{d\}$ |
| 11 | $\emptyset$ | $\emptyset$ | $\{a, b, d\}$ | $\{a\}$ |
| 12 | $\emptyset$ | $\emptyset$ | $\{a, b, c\}$ | $\{a, b\}$ |
| 13 | $\{b, c, d\}$ | $\emptyset$ | $\{b, d\}$ | $\emptyset$ |
| 14 | $\{a, c, d\}$ | $\emptyset$ | $\{a, c\}$ | $\emptyset$ |
| 15 | $\{a, b, d\}$ | $\emptyset$ | $\{b, d\}$ | $\emptyset$ |
| 16 | $\{a, b, c\}$ | $\emptyset$ | $\{a, c\}$ | $\emptyset$ |
| 17 | $\emptyset$ | $\{b, c, d\}$ | $\emptyset$ | $\{b, d\}$ |
| 18 | $\emptyset$ | $\{a, c, d\}$ | $\emptyset$ | $\{a, c\}$ |
| 19 | $\emptyset$ | $\{a, b, d\}$ | $\emptyset$ | $\{b, d\}$ |
| 20 | $\emptyset$ | $\{a, b, c\}$ | $\emptyset$ | $\{a, c\}$ |
| 21 | $\{b, c, d\}$ | $\emptyset$ | $\emptyset$ | $\{b, c\}$ |
| 22 | $\{a, c, d\}$ | $\emptyset$ | $\emptyset$ | $\{a, c, d\}$ |
| 23 | $\{a, b, d\}$ | $\emptyset$ | $\emptyset$ | $\{a, b, d\}$ |
| 24 | $\{a, b, c\}$ | $\emptyset$ | $\emptyset$ | $\{b, c\}$ |

Start solving from constructing a partial assignment. Assume that attribute $X_1$ is chosen and its value $a$ is considered. This partial assignment can be expressed as the $C$-system $Ass1[X_1] = [\{a\}]$. Now it is necessary to "tune" the initial $D$-system to a new domain of attribute $X_1$, i.e., a set $\{a\}$. When tuning, rows 2, 3, 4, 14, 15, 16, 22, 23, 24 are eliminated from the $D$-system according to ***S4'***.

The rest of the $D$-system:

$$\begin{array}{c|ccc} & X_2 & X_3 & X_4 \\ & \{a,b,c,d\} & \{a,b,c,d\} & \{a,b,c,d\} \\ \hline 1 & \{c,d\} & \emptyset & \emptyset \\ 5 & \{b,c,d\} & \{c,d\} & \emptyset \\ 6 & \{a,c,d\} & \{d\} & \emptyset \\ 7 & \{a,b,d\} & \{a\} & \emptyset \\ 8 & \{a,b,c\} & \{a,b\} & \emptyset \\ 9 & \emptyset & \{b,c,d\} & \{c,d\} \\ 10 & \emptyset & \{a,c,d\} & \{d\} \\ 11 & \emptyset & \{a,b,d\} & \{a\} \\ 12 & \emptyset & \{a,b,c\} & \{a,b\} \\ 13 & \emptyset & \{b,d\} & \emptyset \\ 17 & \{b,c,d\} & \emptyset & \{b,d\} \\ 18 & \{a,c,d\} & \emptyset & \{a,c\} \\ 19 & \{a,b,d\} & \emptyset & \{b,d\} \\ 20 & \{a,b,c\} & \emptyset & \{a,c\} \\ 21 & \emptyset & \emptyset & \{b,c\} \end{array}.$$

Rows (tuples) 1, 13, 21 contain one non-empty component each. "Tune" the $D$-system to new domains: $X_2 - \{c,d\}$, $X_3 - \{b,d\}$, $X_4 - \{b,c\}$. According to ***S5'***, eliminate "redundant" values from the components. According to ***S4'***, eliminate rows 1, 5, 6, 9, 11, 13, 17, 18, 21 containing full components.

We obtain:

$$\begin{array}{c|ccc} & X_2 & X_3 & X_4 \\ & \{c,d\} & \{b,d\} & \{b,c\} \\ \hline 7 & \{d\} & \emptyset & \emptyset \\ 8 & \{c\} & \{b\} & \emptyset \\ 10 & \emptyset & \{d\} & \emptyset \\ 12 & \emptyset & \{b\} & \{b\} \\ 19 & \{d\} & \emptyset & \{b\} \\ 20 & \{c\} & \emptyset & \{c\} \end{array}.$$

The analysis of row 7 has shown that the domain of variable $X_2$ should be reduced to $\{d\}$. Then taking into account a new domain of variable $X_2$, from the analysis of row 8, it follows that domain $X_3$ becomes equal to $\{b\}$, and the analysis of row 20 allows reduction of the domain of variable $X_4$ to $\{c\}$. That is, we obtain the following complete assignment $\{(X_1, a), (X_2, d), (X_3, b), (X_4, c)\}$. However, the values of the variables are in conflict with row 10 of the $D$-system specifying the problem. It means that a conflict $\{(X_1, a), (X_3, b)\}$ is found: because row 10 of the initial $D$-system contains non-empty components only in positions $X_1$ and $X_3$.

Hence, partial assignment $Ass1[X_1] = [\{a\}]$, which caused the conflict, gets in a queue of tabu states. The feature of the method is that the propagation process is going on even under the conditions of the conflict found earlier.

At this moment the component implementing the local search, starts functioning. Among all the variables included into the partial assignment, the variable participating in the maximum number of conflicts, is chosen. As in the partial assignment $Ass1[X_1] = [\{a\}]$ there is only one variable $X_1$ participating in the only conflict, a new partial assignment is obtained by taking the complement of the component $\{a\}$: $Ass2[X_1] = [\{b, c, d\}]$.

Then "tune" the initial $D$-system to a newly formed partial assignment. Having "tuned" the $D$-system to a new domain $\{b, c, d\}$, we have:

| | $X_1$ | $X_2$ | $X_3$ | $X_4$ |
|---|---|---|---|---|
| | $\{b, c, d\}$ | $\{a, b, c, d\}$ | $\{a, b, c, d\}$ | $\{a, b, c, d\}$ |
| 2 | $\{c, d\}$ | $\{d\}$ | $\emptyset$ | $\emptyset$ |
| 3 | $\{b, d\}$ | $\{a\}$ | $\emptyset$ | $\emptyset$ |
| 4 | $\{b, c\}$ | $\{a, b\}$ | $\emptyset$ | $\emptyset$ |
| 5 | $\emptyset$ | $\{b, d\}$ | $\{c, d\}$ | $\emptyset$ |
| 6 | $\emptyset$ | $\{a, d\}$ | $\{d\}$ | $\emptyset$ |
| 8 | $\emptyset$ | $\{a, b\}$ | $\{a, b\}$ | $\emptyset$ |
| 9 | $\emptyset$ | $\emptyset$ | $\{b, c, d\}$ | $\{c, d\}$ |
| 10 | $\emptyset$ | $\emptyset$ | $\{a, c, d\}$ | $\{d\}$ |
| 11 | $\emptyset$ | $\emptyset$ | $\{a, b, d\}$ | $\{a\}$ |
| 12 | $\emptyset$ | $\emptyset$ | $\{a, b, c\}$ | $\{a, b\}$ |
| 14 | $\{c, d\}$ | $\emptyset$ | $\{a, c\}$ | $\emptyset$ |
| 15 | $\{b, d\}$ | $\emptyset$ | $\{b, d\}$ | $\emptyset$ |
| 16 | $\{b, c\}$ | $\emptyset$ | $\{a, c\}$ | $\emptyset$ |
| 17 | $\emptyset$ | $\{b, d\}$ | $\emptyset$ | $\{b, d\}$ |
| 18 | $\emptyset$ | $\{a, d\}$ | $\emptyset$ | $\{a, c\}$ |
| 20 | $\emptyset$ | $\{a, b\}$ | $\emptyset$ | $\{a, c\}$ |
| 22 | $\{c, d\}$ | $\emptyset$ | $\emptyset$ | $\{a, c, d\}$ |
| 23 | $\{b, d\}$ | $\emptyset$ | $\emptyset$ | $\{a, b, d\}$ |
| 24 | $\{b, c\}$ | $\emptyset$ | $\emptyset$ | $\{b, c\}$ |

.

Then it is necessary to extend partial assignments. Choose one of the variables containing a minimum quantity of values in the domain to extend the partial assignment. Let variable $X_2$ be chosen. In the column which corresponds to variable $X_2$, the value a occurs 6 times, and the values $b$ and $d$ occur 5 times each. Hence, we choose the value $a$.

Then consider partial assignment $Ass3[X_1X_2] = [\{b, c, d\}\{a\}]$. As a result of the initial $D$-system "tuning", which specifies the conditions of the problem, we obtain specified values of all the variables, not violating any constraints: $Ass4[X_1X_2X_3X_4] = [\{c\}\{a\}\{d\}\{b\}]$. Hence, we have reached the solution of the problem: the first queen is on field $c_1$, the second one is on field $a_2$, the third one is on field $d_3$, and the fourth one is on field $b_4$ (Figure 4).

## 5. Conclusions

For the first time, the hybrid methods have been developed to solve non-quantitative constraint satisfaction problems formalized by the $C$- and $D$-systems. The peculiar feature of the methods proposed is that these methods do not use backtracking strategies

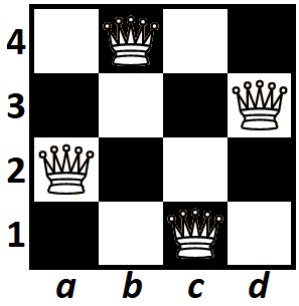


**Figure 4:** One of the possible solutions of the "4-Queens" problem

resulting often in the exponential increase of computation time, under the increase in the size of the problem.

The first hybrid method proposed integrates the author's methods of non-quantitative constraint propagation and the existing algorithms of structural constraint graph decomposition. The main computational difficulty connected with application of this method is in the component implementing the structural constraint graph decomposition, but there are the approximated ("greedy") algorithms allowing this decomposition to be rather quickly implemented. Each sub-problem with a tree structure, takes polynomial time to be solved by the author's non-quantitative constraint propagation algorithms. The method suits well to the situations when there is a series of CSPs with graphs of the similar structure. For instance, the situation like this often occurs in analyzing typical queries to the data storage.

The second hybrid method integrates the following components: the author's methods of non-quantitative constraint propagation to reduce the solution space; the method of local search on partial assignments to quickly quit the sub-spaces containing no solution; the tabu search method to avoid repeated analyzing the states already studied. This method is to be applied to the situations when it is necessary to reach the solution using large bodies of information, with no a priori information on the CSP graph structure. The method suits well the problems connected with planning and scheduling.

Specialized structures applied in representation and processing the table constraints make it possible to reduce memory consumption for the problem representation, and the detailed analysis of the *C*- and *D*-systems properties underlies the ways of the computational procedures acceleration.

## Acknowledgments

The work was supported by the Russian fund of basic researches (RFFI) – grants 17-29-07021-ofi_m, 18-07-00615-a, 20-07-00708-a.

## References


[Blu11] C. Blum, J. Puchinger, G. Raidl, A. Roli. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6):4135–4151, 2011.

[Cha04] K. Chatzikokolakis, G. Boukeas, P. Stamatopoulos. Construction and Repair: A Hybrid Approach to Search in CSPs. *Methods and Applications of Artificial Intelligence. SETN 2004. Lecture Notes in Computer Science*, 3025:342–351, 2004.

[Che10] K. C. Cheng, R. H. Yap. An MDD-based generalized arc consistency algorithm for positive and negative table constraints and some global constraints. *Constraints*, 15(2):265–304, 2010.

[Dec87] R. Dechter, J. Pearl. Network-based heuristics for constraint satisfaction problems. *Artificial Intelligence*, 34(1):1–38, 1987.

[Dec89] R. Dechter, J. Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, 38(3):353–366, 1989.

[Dec90] R. Dechter. Enhancement schemes for constraint processing: Backjumping, learning and cutset decomposition. *Artificial Intelligence*, 41(3):273–312, 1990.

[Fei18] A. A. Feitosa Neto, A. M. P. Canuto, J. C. Xavier-Junior. Hybrid Metaheuristics to the Automatic Selection of Features and Members of Classifier Ensembles. *Information*, 9:268, 2018.

[Fre82] E. C. Freuder. A sufficient condition for backtrack-free search. *Journal of the ACM*, 29(1):24–32, 1982.

[Fre85] E. C. Freuder. A sufficient condition for backtrack-bounded search. *Journal of the ACM*, 32(4):755–761, 1985.

[Jeg17] P. Jégou, C. Terrioux. Combining restarts, nogoods and bag-connected decompositions for solving CSPs. *Constraints*, 22:191–229, 2017.

[Jus02] N. Jussien, O. Lhomme. Local search with constraint propagation and conflict-based heuristics. *Artificial Intelligence*, 139:21–45, 2002.

[Kat07] G. Katsirelos, T. Walsh. A Compression Algorithm for Large Arity Extensional Constraints. *Principles and Practice of Constraint Programming – CP 2007. Lecture Notes in Computer Science*, 4741:379–393, Springer, Berlin, Heidelberg, 2007.

[Kul15] B. Kulik, A. Zuenko, A. Fridman. Deductive and defeasible reasoning on the basis of a unified algebraic approach. *Scientific and Technical Information Processing*, 42(6):402–410, Allerton Press Inc. 2015.

[Lim12] K. Limtanyakul, U. Schwiegelshohn. Improvements of constraint programming and hybrid methods for scheduling of tests on vehicle prototypes. *Constraints*, 17:172–203, 2012.

[Mig01] I. Miguel, Q. Shen. Solution techniques for constraint satisfaction problems: foundations. *Artificial Intelligence Review*, 15:243–267, 2001.

[Nat15] M. Nattaf, C. Artigues, P. Lopez. A hybrid exact method for a scheduling problem with a continuous resource and energy constraints. *Constraints*, 20:304–324, 2015.

[Rus10] S. Russel, P. Norvig. *Artificial Intelligence: A Modern Approach. 3rd edition.* Prentice Hall, 2010.

[Ste97] O. Steinmann, A. Strohmaier, T. Stutzle. Tabu search vs. random walk. *KI-97: Advances in Artificial Intelligence*, 337–348 ,Springer Verlag, Berlin, Germany, 1997.

[Tho16] E. Thorstensen. Structural decompositions for problems with global constraints. *Constraints*, 21:198–222, 2016.

[Ver17] H. Verhaeghe, C. Lecoutre, Y. Deville, P. Schaus. Extending Compact-Table to Basic Smart Tables. *Principles and Practice of Constraint Programming. 23rd International Conference, CP 2017*, 297–307, Melbourne, VIC, Australia, 2017.

[Wan16] R. Wang, W. Xia, R. Yap, Z. Li. Optimizing Simple Tabular Reduction with a bitwise representation. *Proceedings of IJCAI 2016*, 787–793, 2016.

[Zue14] A. A. Zuenko. Vyvod na ogranicheniyakh s primeneniem matrichnogo predstavleniya konechnykh predikatov [Constraint inference based on the matrix representation of finite predicates]. *Iskusstvennyi intellekt i prinyatie reshenii [Artificial Intelligence and Decision Making]*, 3:21–31, 2014. (In russian)

[Zue17] A. A. Zuenko, P. A. Lomov, A. G. Oleinik. Application of constraint propagation techniques to speed up processing of queries to ontologies. *SPIIRAS Proceedings*, 1(50):112–136, 2017.

[Zue18a] A. A. Zuenko. Matrix-like Structures for Representation and Processing of Constraints Over Finite Domains. *Proceedings of the Third International Scientific Conference "Intelligent Information Technologies for Industry" (IITI'18, Volume 2), Advances in Intelligent Systems and Computing (AISC)*, 875:428–438, Springer Nature Switzerland AG, 2019.

[Zue18b] A. A. Zuenko. Local Search in Solution of Constraint Satisfaction Problems Represented by Non-Numerical Matrices. *Proceedings of the 2nd International Conference on Computer Science and Application Engineering (CSAE '18)*, 1–5, ACM New York, NY, USA, 2018.

[Zue18c] A. Zuenko, Y. Oleynik. Programming of Algorithms of Matrix-Represented Constraints Satisfaction by Means of Choco Library. *Proceedings of the Third International Scientific Conference "Intelligent Information Technologies for Industry" (IITI'18, Volume 2), Advances in Intelligent Systems and Computing (AISC)*, 875:439–448, Springer Nature Switzerland AG, 2019.