The Joint Use of the Breadth-First Search Strategy and the Constraints Propagation in Smart Tables Handling

Alexander A. Zuenko
Kola Science Centre of the Russian Academy of Sciences, Institute for Informatics and Mathematical Modelling, Apatity, Russia,
zuenko@iimm.ru

ABSTRACT

Recently, in modeling and handling the qualitative relations in framework of Constraint Programming technology the so called table constraints: compressed tables, basic smart tables, smart tables are increasingly used. Smart tables are a generalization of all the types of table constraints listed above. In his earlier studies, the author proposes to subdivide smart tables into the structures of the *C*- and *D*-types. Smart tables of the *C*-type correspond to disjunctive normal forms of logical expressions with elementary unitary and binary predicates and smart tables of the *D*-type correspond to conjunctive normal forms of such expressions. In the Constraint Programming technology the conventional technique of search is to combine constraint propagation with backtracking depth-first search. In this study, a hybrid method is proposed that combines the breadth-first search strategy and the author's method of table constraints propagation. The method is designed to obtain all the solutions of the Constraint Satisfaction Problems modeled by one or several smart tables of the *D*-type. The method is based on the representation of a smart table of the *D*-type in the form of a join of several orthogonalized smart-tables of the *C*-type. A step of search consists of joining chosen pair of smart tables of the *C*-type, followed by constraint propagation. When choosing smart tables of the *C*-type for joining, the proposed heuristic allows us to determine the order of joining the orthogonalized smart tables, which helps to reduce the search space at the next steps of processing. In constraints propagating, computation is accelerated due to the reduction rules developed for the case of smart tables of the *C*-type.

CCS CONCEPTS

• **Mathematics of computing**; • **Discrete mathematics**; • **Combinatorics**; • **Combinatorial algorithms**;

KEYWORDS

Constraint programming technology, Constraint satisfaction problem, Table constraint, Qualitative constraints

ACM Reference Format:
Alexander A. Zuenko. 2021. The Joint Use of the Breadth-First Search Strategy and the Constraints Propagation in Smart Tables Handling. In *The 5th International Conference on Computer Science and Application Engineering (CSAE 2021), October 19–21, 2021, Sanya, China.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3487075.3487076

1 INTRODUCTION

The methods of logical analysis are applied in many Artificial Intelligence (AI) systems. They are intensively developed within the framework of such an AI direction as Constraint Programming, which is based on the declarative paradigm of knowledge representation in the form of a Constraint Satisfaction Problem (CSP) [1].

A *Constraint Satisfaction Problem* is described by a set of variables $X_1, X_2, \ldots, X_n$ and a set of constraints $C_1, C_2, \ldots, C_m$ [2, 3]. Each variable X_i has a non-empty domain D_i. Each constraint C_i on a subset of variables specifies allowable combinations of values for this subset. The solution of the CSP is an assignment of values to all variables $\{X_1=v_1, \ldots, X_n=v_n\}$, which satisfies all the constraints.

The paper considers the CSPs in which variables are discrete and defined on the finite domains. The paper deals with the processing of the so-called table constraints [4-14].

The simplest example of table constraints is relational tables. Any finite predicate can be represented as relational tables containing a truth set of the predicate. However, this explicit representation is not expedient for many relations because it results in the exponential growth of the constraint satisfaction procedures complexity. So, the attempts are taken up to develop a more compact table representation of qualitative relations, particularly, there were proposed the varieties of table constraints, such as *compressed tables* and *smart tables* [8-14].

The studies of prototypes have shown that the known types of table constraints are, first of all, oriented at modeling disjunctive normal forms of finite predicates and are in fact not suitable enough to model conjunctive normal forms of logical formulas.

The author's studies dealing with table constraints handling are [15-17].

A detailed review of related research can be found in [17].

In his earlier study the author proposes a new type of table constraints – smart tables of the *D*-type, which allows convenient modeling and efficient processing of the production rules, some types of logical expressions, as well as some types of global constraints [17]

In Constraint Programming the conventional technique of search is to jointly use the constraint propagation methods and backtracking depth-first search strategy based on specialized heuristics to select a variable and its value and reasonable algorithms to backtrack to the state that causes an invalid assignment. The feature of backtracking depth-first search methods is to expand, step by step, a partial solution to a complete one. If a partial solution is illegal then backtrack is implemented and the previous partial solution is expanded in an alternative direction.

The method proposed in the study combines breadth-first search strategy and the author‘s method of table constraint propagation. The method is designed to solve constraint satisfaction problems represented by one or several smart tables of the D-type.

2 THE TABLE CONSTRAINTS FOR KNOWLEDGE REPRESENTATION

The paper [17] considers the following types of table constraints: typical relational table, compressed table, basic smart table, smart table. Smart tables are a generalization of all the types of the table constraints listed above. Smart tables are proposed to be subdivided into the C- and D-type structures. Smart tables can contain in their scheme not just simple but also composite attributes, the values of which are relations from a predefined set.

Smart tables of the C-type correspond to disjunctive normal forms (DNFs) of logical expressions with elementary unitary and binary predicates and smart tables of the D-type correspond to conjunctive normal forms (CNFs) of such expressions. The origin of the terms *smart table of the C-type* and the *D-type* is due to the fact that the former type of tables consists of smart tuples of the C-type (from *Conjunction*), which are used to represent conjunctions of unitary and binary predicates and the latter type of tables consists of smart tuples of the D-type (from *Disjunction*), which model disjunctions of elementary predicates.

In the paper two dummy components are used to represent the content of smart tables: a complete component (denoted by “$*$”) is a set equal to the domain of the corresponding (according to its location in the tuple) attribute; an empty set – $\varnothing$.

A smart table of the C-type is represented as a matrix enclosed in square brackets.

Example 1. *Let’s consider a constraint that describes the adaptable conditions for the formation of the price of a product when solving the problem of purchasing planning for a certain company. In a natural language the constraint under consideration looks like: either the price of product 1 is $ 35, while more than 15 units of product 1 and the same number of units of product 2 must be purchased, or the price of product 1 is $50, otherwise.*

In terms of logical formulas this constraint can be expressed as follows:

$$(X_1 \geq 15) \wedge (X_1 = X_2) \wedge (C = 35) \vee (X_1 < 15) \wedge (C = 50) \vee$$
$$\vee (X_1 \neq X_2) \wedge (C = 50)\,,$$

where:
X_1 is the number of units of product 1, $X_1 \in [0; 111]$;
X_2 is the number of units of product 2, $X_2 \in [0; 250]$;
C is product price, $C \in \{35, 50\}$.
Let’s represent the constraint as a smart table of the C-type:

$$T[X_1, X_2, C, X_1X_2] = \begin{bmatrix} \geq 15 & * & \{35\} & = \\ < 15 & * & \{50\} & * \\ * & * & \{50\} & \neq \end{bmatrix}.$$

Each row of the smart table corresponds to some disjunct of the above logical formula. The scheme of this smart table contains a composite attribute X_1X_2 whose domain is a set of binary relations $\{=, \neq\}$.

Example 1 shows that using smart tables of the C-type, it is convenient to represent DNFs of the finite predicates (in particular, a true set of finite predicates).

Using smart tables of the D-type the CNFs of the finite predicates are modeled. The content of a smart tables of the D-type is enclosed in inverted square brackets. Smart tables of the D-type make it easy to calculate the complement of smart tables of the C-type relative to the given universum: it is required to take the complement of each component of the smart table. For example, the complement of the smart table $T\ [X_1, X_2, C, X_1X_2]$ is a smart table of the D-type:

$$\overline{T}[X_1, X_2, C, X_1X_2] = \left]\begin{matrix} < 15 & \emptyset & \{50\} & \neq \\ \geq 15 & \emptyset & \{35\} & \emptyset \\ \emptyset & \emptyset & \{35\} & = \end{matrix}\right[.$$

A smart table of the D-type $\bar{T}$ can be represented as the following logical expression:

$$[(X_1 < 15) \vee (X_1 \neq X_2) \vee (C = 50)] \wedge\ [(X_1 \geq 15) \vee (C = 35)] \wedge$$
$$\wedge [(X_1 = X_2) \vee (C = 35)],$$

Since smart tables are a “compressed” representation of multiplace relations (typical tables), all the operations of relational algebra can be applied to them. We have already considered the application of the operation of complement earlier. Further computations will use the operation of a join of relations ($\oplus$). Note that when performing complement and a join operations, there is no need to decompose smart tables into a set of elementary tuples.

Any smart table of the D-type can be represented as a join of several diagonal smart tables of the C-type, each of which corresponds to some row of the initial smart table of the D-type.

For example:

$$\overline{T}[X_1, X_2, C, X_1X_2] = \left]\begin{matrix} < 15 & \emptyset & \{50\} & \neq \\ \geq 15 & \emptyset & \{35\} & \emptyset \\ \emptyset & \emptyset & \{35\} & = \end{matrix}\right[=$$
$$= \begin{bmatrix} < 15 & * & * & * \\ * & * & \{50\} & * \\ * & * & * & \neq \end{bmatrix} \oplus \begin{bmatrix} \geq 15 & * & * & * \\ * & * & \{35\} & * \end{bmatrix} \oplus$$
$$\oplus \begin{bmatrix} * & * & \{35\} & * \\ * & * & * & = \end{bmatrix} = \begin{bmatrix} < 15 & * & \{35\} & * \\ \geq 15 & * & \{50\} & = \\ * & * & \{35\} & \neq \end{bmatrix}.$$

A diagonal smart table is a square smart table in which non-dummy components can be located only on the main diagonal. In the expression above, the rows containing empty components are eliminated from the diagonal smart tables of the C-type.

The resulting smart table of the C-type describes in a “compressed” form a true set of CNF which is presented by the initial smart table of the D-type. Elementary substitutions are easy to obtain by considering each row of smart table of the C-type separately. Since the smart table under our consideration is a complement of smart table T, it describes prohibited combinations of numbers of products and their prices. In particular, from the first row of the resulting smart table of the C-type it follows that product 1 cannot be purchased at a price of $ 35 if its quantity is less 15 units.

To illustrate how a join of smart tables of the C-type is performed, consider the join of smart tables 2 and 3 in the expression above:

$$\begin{bmatrix} \geq 15 & * & * & * \\ * & * & \{35\} & * \end{bmatrix} \oplus \begin{bmatrix} * & * & \{35\} & * \\ * & * & * & = \end{bmatrix} = \begin{bmatrix} \geq 15 & * & \{35\} & * \\ \geq 15 & * & * & = \\ * & * & \{35\} & * \\ * & * & \{35\} & = \end{bmatrix}.$$

When joining two smart tables of the C-type join of each row of the first operand and each row of the second operand is performed. When joining rows, for components-sets corresponding to common attributes of both operands, an intersection operation is performed.

Thus, generally, the problem of finding all true sets of CNF of the finite predicate is solved. However, in this form the solution of the problem turns out to be too time-consuming. The methods for reducing complexity of this transformation are briefly described below.

One of the acceleration methods can be an orthogonalization based on Poretsky ratio known from mathematical logic:

$$A \vee B = A \vee \overline{A}B.$$

In particular, the equations are true:

$$\begin{bmatrix} \geq 15 & * & * & * \\ * & * & \{35\} & * \end{bmatrix} = \begin{bmatrix} \geq 15 & * & * & * \\ < 15 & * & \{35\} & * \end{bmatrix}$$

$$\begin{bmatrix} * & * & \{35\} & * \\ * & * & * & = \end{bmatrix} = \begin{bmatrix} * & * & \{35\} & * \\ * & * & \{50\} & = \end{bmatrix}.$$

Now let's find join of these two orthogonalized smart tables of the C-type:

$$\begin{bmatrix} \geq 15 & * & * & * \\ < 15 & * & \{35\} & * \end{bmatrix} \oplus \begin{bmatrix} * & * & \{35\} & * \\ * & * & \{50\} & = \end{bmatrix} = \begin{bmatrix} \geq 15 & * & \{35\} & * \\ \geq 15 & * & \{50\} & = \\ < 15 & * & \{35\} & * \end{bmatrix}.$$

As a result of joining the two orthogonalized smart tables under consideration we obtained fewer rows in the resulting table then when they were joined without preliminary orthogonalization.

In general, the order of joining diagonal smart tables of the C-type significantly increases the speed of the computational process.

Note specially that:

$$\begin{bmatrix} \geq 15 & * & \{35\} & * \\ \geq 15 & * & * & = \\ * & * & \{35\} & * \\ * & * & \{35\} & = \end{bmatrix} = \begin{bmatrix} \geq 15 & * & \{35\} & * \\ \geq 15 & * & \{50\} & = \\ < 15 & * & \{35\} & * \end{bmatrix}.$$

These smart tables model one and the same set of elementary tuples and are equivalent in this sense. In other words, one and the same set of elementary tuples can be modeled by smart tables of the C-type, which may look absolutely differently. Two equivalent smart tables of the C-type can be obtained from one another as a result of equivalent transformations. Some of these transformations underlie the proposed method of inference on smart tables of the C-type (the propagation method for the case of smart tables of the C-type), the purpose of which is to reduce the search space in polynomial time.

Below there are the reduction rules for the case of smart tables of the C-type that allow us to remove some "redundant" values from the attribute domains, components of smart tables and also exclude entire rows and columns from consideration.

Statement 1' (**S1'**). If all the rows of the smart table of the C-type are empty, that is, they contain at least one empty component each, then the C-system is empty (the corresponding CSP is inconsistent).

Statement 2' (**S2'**). If all the components of some attribute (column of the smart table of the C-type) are complete, then this attribute can be removed from the smart table of the C-type (all the components placed in the corresponding column are removed) and the pair "removing attribute – its domain" is stored in the partial solution vector.

Statement 3' (**S3'**). If a domain of some attribute of smart table of the C-type contains values that do not occur in the corresponding column, then these values are removed from this domain.

Statement 4' (**S4'**). If a row of a smart table of the C-type contains at least one empty component (the row is empty), then the row is removed from smart table.

Statement 5' (**S5'**). If a component of some attribute of a smart table of the C-type contains a value that does not belong to the corresponding domain, then this value is removed from the component.

Statement 6' (**S6'**). If in a row of a smart table of the C-type a component corresponding to a simple attribute that forms some composite attribute is reduced, then in this row the component corresponding to the mentioned composite attribute must be modified, taking into account the new value of the component of the simple attribute.

Statement 7' (**S7'**). If in a row of a smart table of the C-type, a component corresponding to a composite attribute is reduced then in this row the components of corresponding simple attributes must be modified, taking into account the newly obtained component of a composite attribute.

Statement 8' (**S8'**). If in a smart table of the C-type the domain of a simple attribute that forms some composite attribute is reduced, then the domain of the composite attribute must be modified, taking into account the new domain of the simple attribute.

Statement 9' (**S9'**). If the domain of a composite attribute is reduced, then the domains of corresponding simple attributes also must be modified, taking into account the newly obtained domain of a composite attribute.

Similar reduction rules for the case of the D-type smart tables are given in [17]. Next, we proceed to the description of the proposed method.

3 THE METHOD PROPOSED

In CSPs, where it is necessary to reach all the solutions, it is reasonable to jointly use the breadth-first search strategy and the author's methods of non-numerical constraints propagation. In the present studies, a case is considered when the CSP is represented in a form of a single smart table of the D-type or several smart tables of the D-type. The hybrid method proposed is based on the representation of the smart table of the D-type in a form of a join of several orthogonalized smart tables of the C-type. We call the smart tables of the C-type an orthogonalized smart tables of the C-type, whose tuples (rows) are orthogonal pair wise. The method allows us to determine the order of joining orthogonalized smart tables, which would contribute to reducing the computation. The problem is to some extent similar to the problem of multiplication of numerical matrices where the arrangement of brackets in the expression, i.e.

determination of the order of matrices multiplication, significantly effects the total number of the operations.

At each step of search a pair of smart tables of the C-type is proposed to be chosen so that to reduce the search space as much as possible. The following heuristic is used in choosing the smart tables for joining:

$$J(K[S], T[R]) = \ |K[S]| \times |T[R]| \times |S \backslash R| \times |R \backslash S|,$$

where: $K[S]$ – is the first of the smart tables pair chosen; $T[R]$ – is the second of the smart tables pair chosen at the current step of search; S – the scheme (a set of attributes) of the smart table of the C-type K; R – the scheme (a set of attributes) of the smart table of the C-type T; $|K[S]|$ is the number of (elementary or non-elementary) tuples of the given smart table of the C-type $K[S]$.

This heuristic is borrowed from [18], where it is applied to join typical relational tables.

After joining the pair of the smart tables of the C-type, it may turn out that some values of the corresponding variable are absent in one of the columns of the resulting smart table of the C-type. Thus, after each joining, it is necessary to carry out the constraints propagation, applying statements **S1'**-**S8'**, i.e. the reduction rules described earlier for the case of the smart tables of the C-type. When the "redundant" values are removed, all the smart tables are still orthogonalized.

The values of the function $\mathcal{J}(K[S],\ T[R])$ can differ significantly depending on the interpretation of the concept "tuple of smart table of the C-type". The following can be considered as tuples of a smart table of the C-type: a) smart tuples (rows of the smart table); b) elementary tuples obtained when transforming a smart table into a typical relational table.

When considering case b), it should be taken into account that if the diagonal smart table of the C-type is orthogonalized, then it is easy to calculate the cardinality of the set of elementary tuples of the given smart table without decomposing it into a typical relational table. To do that, the cardinalities of all the non-empty components placed in a row should be multiplied, and the results of multiplications obtained for each row, should be summed.

Let's set the task to estimate which of the interpretations of the concept "tuple of smart table of the C-type" allows us to better choose a pair of joined smart tables at the step of search.

Example 2. *Consider the peculiarities of the method application on the example of the smart table of the D-type. So, we have the smart table specified in* $\mathbf{S}=X_1 \times X_2 \times X_3 \times X_4=\{1, 2, 3, 4, 5\}^4$:

$$\begin{array}{r} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \left]\begin{array}{cccc} \{2,3\} & \emptyset & \{1,2,3\} & \emptyset \\ \{2\} & \{1,2,5\} & \{4\} & \emptyset \\ \emptyset & \{1,2,4\} & \{3,4\} & \{3,5\} \\ \{1,2\} & \{3,4,5\} & \emptyset & \{3\} \\ \emptyset & \emptyset & \{1,5\} & \{4\} \\ \emptyset & \{2,3\} & \{1,2,3\} & \{1\} \\ \{2,4,5\} & \emptyset & \{3,5\} & \{3,5\} \\ \{5\} & \emptyset & \{2,3,4\} & \{1,3,4\} \\ \{1,3\} & \emptyset & \{4,5\} & \emptyset \end{array}\right[.$$

This smart table of the D-type can be represented as a joining the following smart tables of the C-type:

$$K_1\,[X_1X_3] \oplus K_2\,[X_1X_2X_3] \oplus K_3\,[X_2X_3X_4] \oplus$$
$$\oplus K_4\,[X_1X_2X_4] \oplus K_5\,[X_3X_4] \oplus K_6\,[X_2X_3X_4] \oplus$$
$$\oplus K_7\,[X_1X_3X_4] \oplus K_8\,[X_1X_3X_4] \oplus K_9\,[X_1X_3]\,,$$

where:

$$K_1\,[X_1X_3] = \begin{bmatrix} \{2,3\} & * \\ \{1,4,5\} & \{1,2,3\} \end{bmatrix},$$

$$K_2\,[X_1X_2X_3] = \begin{bmatrix} \{2\} & * & * \\ \{1,3,4,5\} & \{1,2,5\} & * \\ \{1,3,4,5\} & \{3,4\} & \{4\} \end{bmatrix},$$

$$K_3\,[X_2X_3X_4] = \begin{bmatrix} \{1,2,4\} & * & * \\ \{3,5\} & \{3,4\} & * \\ \{3,5\} & \{1,2,5\} & \{3,5\} \end{bmatrix},$$

$$K_4\,[X_1X_2X_4] = \begin{bmatrix} \{1,2\} & * & * \\ \{3,4,5\} & \{3,4,5\} & * \\ \{3,4,5\} & \{1,2\} & \{3\} \end{bmatrix},$$

$$K_5\,[X_3X_4] = \begin{bmatrix} \{1,5\} & * \\ \{2,3,4\} & \{4\} \end{bmatrix},$$

$$K_6\,[X_2X_3X_4] = \begin{bmatrix} \{2,3\} & * & * \\ \{1,4,5\} & \{1,2,3\} & * \\ \{1,4,5\} & \{4,5\} & \{1\} \end{bmatrix},$$

$$K_7\,[X_1X_3X_4] = \begin{bmatrix} \{2,4,5\} & * & * \\ \{1,3\} & \{3,5\} & * \\ \{1,3\} & \{2,4,5\} & \{3,5\} \end{bmatrix},$$

$$K_8\,[X_1X_3X_4] = \begin{bmatrix} \{5\} & * & * \\ \{1,2,3,4\} & \{2,3,4\} & * \\ \{1,2,3,4\} & \{1,5\} & \{1,3,4\} \end{bmatrix},$$

$$K_9\,[X_1X_3] = \begin{bmatrix} \{1,3\} & * \\ \{2,4,5\} & \{4,5\} \end{bmatrix}.$$

In this case, an orthogonalization is made which is aimed at calculation of elementary tuples in each of the smart table of the C-type enumerated.

The elementary step of the search is in that, at first, joining two smart tables of the C-type is implemented. The two smart tables of the C-type are chosen in accordance with the heuristic mentioned above. Then the constraints propagation procedure is carried out.

For the two mentioned interpretations of the concept of "tuple of a smart table of the C-type", the values of the heuristic function for each pair of smart tables joined at the first step of the search are presented in Tables 1 and 2. The actual number of elementary tuples that is obtained when joining a pair of smart tables at the first step of the search is shown in Table 3.

Analyzing Tables 1 and 2, one can conclude that with both interpretations of the concept "tuple of a smart table of the C-type", the value of the heuristic function "null" corresponds to the same pairs of joined smart tables. However, the calculation of Table 1 requires much more arithmetic operations. One more argument in favor of using the number of its smart tuples as the cardinality of a smart table, rather than elementary tuples, is that the minimum (maximum) value in Table 2 corresponds to the minimum (maximum) value in Table 3.

Table 1: The Values of the Heuristic Function at the First Step of the Search for the Case with Elementary Tuples

	K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_8
K_2	0							
K_3	4066	9951						
K_4	3838	9393	10807					
K_5	247	2418	0	2626				
K_6	3838	9393	0	10201	0			
K_7	0	9951	11449	10807	0	10807		
K_8	0	10137	11663	11009	0	11009	0	
K_9	0	0	3424	3232	208	3232	0	0

Table 2: The Values of the Heuristic Function at the First Step of the Search for the Case with Smart Tuples

	K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_8	K_9
K_1									
K_2	0								
K_3	12	9							
K_4	12	9	9						
K_5	4	12	0	12					
K_6	12	9	0	9	0				
K_7	0	9	9	9	0	9			
K_8	0	9	9	9	0	9	0		
K_9	0	0	12	12	4	12	0	0	

Table 3: The Actual Number of Elementary Tuples That Is Obtained When Joining a Pair of Smart Tables at the First Step of the Search

	K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_8	K_9
K_1									
K_2	69								
K_3	514	411							
K_4	269	345	415						
K_5	47	229	51	259					
K_6	601	361	85	403	50				
K_7	78	399	469	433	55	415			
K_8	83	424	491	441	49	449	93		
K_9	10	60	346	328	44	280	62	68	

Applying the heuristic proposed above, choose for joining the following smart tables: $K_1[X_1X_3]$ and $K_9[X_1X_3]$ having the same schemes. Calculate the result of their joining:

$$K_1\,[X_1X_3] \oplus K_9\,[X_1X_3] =$$

$$\begin{bmatrix} \{2,3\} & * \\ \{1,4,5\} & \{1,2,3\} \end{bmatrix} \oplus \begin{bmatrix} \{1,3\} & * \\ \{2,4,5\} & \{4,5\} \end{bmatrix} = \begin{bmatrix} \{3\} & * \\ \{2\} & \{4,5\} \\ \{1\} & \{1,2,3\} \end{bmatrix}.$$

Then, according to Statements **S1’**-S**8’**, one can reduce the domain of variable X_1 to the set {1, 2, 3}, “having summed” the values of the first column of the smart table $K_1[X_1X_3] \oplus K_9[X_1X_3]$. Now, one should “tune” the rest smart tables of the C-type (K_1 – K_8) to a new domain of variable X_1. At the current stage of search we have:

Current domains: X_1 – {1, 2, 3}, X_2 – {1, 2, 3, 4, 5}, X_3 – {1, 2, 3, 4, 5}, X_4 – {1, 2, 3, 4, 5}.

Current constraints:

$$K_1\,[X_1X_3] \oplus K_9\,[X_1X_3] = \begin{bmatrix} \{3\} & * \\ \{2\} & \{4,5\} \\ \{1\} & \{1,2,3\} \end{bmatrix},$$

$$K_2\,[X_1X_2X_3] = \begin{bmatrix} \{2\} & * & * \\ \{1,3\} & \{1,2,5\} & * \\ \{1,3\} & \{3,4\} & \{4\} \end{bmatrix},$$

$$K_3\,[X_2X_3X_4] = \begin{bmatrix} \{1,2\} & * & * \\ \{3\} & \{3,4\} & * \\ \{3\} & \{1,2,5\} & \{3,5\} \end{bmatrix},$$

$$K_4\left[X_1X_2X_4\right] = \begin{bmatrix} \{1,2\} & * & * \\ \{3\} & \{3,4,5\} & * \\ \{3\} & \{1,2\} & \{3\} \end{bmatrix},$$

$$K_5\left[X_3X_4\right] = \begin{bmatrix} \{1,\} & * \\ \{2,3\} & \{4\} \end{bmatrix},$$

$$K_6\left[X_2X_3X_4\right] = \begin{bmatrix} \{2,3\} & * & * \\ \{1\} & \{1,2,3\} & * \\ \{1\} & \{4,5\} & \{1\} \end{bmatrix},$$

$$K_7\left[X_1X_3X_4\right] = \begin{bmatrix} \{2\} & * & * \\ \{1,3\} & \{3,5\} & * \\ \{1,3\} & \{2,4,5\} & \{3,5\} \end{bmatrix},$$

$$K_8\left[X_1X_3X_4\right] = \begin{bmatrix} \{1,2,3\} & \{2,3,4\} & * \\ \{1,2,3\} & \{1,5\} & \{1,3,4\} \end{bmatrix}.$$

In the smart tables K_2-K_7, the first columns were corrected. In the smart table K_8 the first row was removed. Further steps of the search process are carried out by analogy.

4 CONCLUSIONS

The paper proposes to reduce the search of the solutions of non-numerical CSPs, presented by smart-tables of the *D*-type, to determination of the order of joining several smart tables of the *C*-type. A heuristic is proposed for choosing a pair of joined smart tables of the *C*-type at each step of search. Two variants of using this heuristic are studied. A conclusion is made that it is reasonable to use the number of smart tuples as the cardinality of a smart table, rather than the number of elementary tuples of the relation. For the first time, reduction rules are proposed for the case of smart tables of the *C*-type. The developed method makes it possible to efficiently find all the solutions of non-numerical CSPs and it is easily adapted to solve optimization problems. The time complexity of backtracking depth-first search is estimated as a product of domains of all the variables. The time complexity of the proposed method is estimated as a product in which each multiplier is equal to the number of non-empty components of the corresponding row of the initial smart table of the *D*-type. The proposed method gives a significant gain in the case of a large number of variables, a large size of their domains and a relatively small number of rows of a smart table. The method is focused on application in intelligent systems dealing with poorly formalized subject domains. It is planned to study the influence of the order of orthogonalization of smart tables columns on the speed of computations when joining a pair of the *C*-type smart tables.

ACKNOWLEDGMENTS

The reported study was funded by RFBR, project numbers 19-07-00359-a, 20-07-00708-a.

REFERENCES

[1] S Russel and P Norvig (2010). Artificial Intelligence: A Modern Approach. 3rd edition. Prentice Hall, 1132 p.
[2] A Mackworth (1977). Consistency in networks of relations. Artificial Intelligence, 8(1), 99-118. DOI: 10.1016/0004-3702(77)90007-8.
[3] R Bartak (1999). Constraint Programming: In Pursuit of the Holy Grail. Proceedings of the Week of Doctoral Students (WDS99), 555-564.
[4] B Charlier, M Khong, C Lecoutre and Y Deville (2017). Automatic Synthesis of Smart Table Constraints by Abstraction of Table Constraints. Proceedings of IJCAI 2017, 681-687. DOI: https://doi.org/10.24963/ijcai.2017/95.
[5] G Audemard, C Lecoutre and M Maamar (2020). Segmented Tables: An Efficient Modeling Tool for Constraint Reasoning. ECAI 2020, 315-322.
[6] R Yap, and W Wang (2020). Generalized Arc Consistency Algorithms for Table Constraints: A Summary of Algorithmic Ideas. AAAI 2020, 13590-13597. DOI: https://doi.org/10.1609/aaai.v34i09.7086.
[7] G Perez and J C Regin (2014). Improving GAC-4 for table and MDD constraints. CP 2014. LNCS, 8656, 606-621. DOI: http://dx.doi.org/10.1007/978-3-319-10428-7_44.
[8] H Verhaeghe, C Lecoutre and P Schaus (2017). Extending compact-table to negative and short tables. Proceedings of AAAI 17, 3951-3957. DOI: https://dl.acm.org/doi/abs/10.5555/3298023.3298142.
[9] L Ingmar and C Schulte (2018) Making Compact-Table Compact. CP 2018, Lecture Notes in Computer Science, 11008, 210-218. DOI: https://doi.org/10.1007/978-3-319-98334-9_14.
[10] K Cheng and R Yap (2010). An MDD-based generalized arc consistency algorithm for positive and negative table constraints and some global constraints. Constraints, 15(2), 265-304. DOI: http://dx.doi.org/10.1007/s10601-009-9087-y.
[11] C Jefferson and P Nightingale (2013). Extending simple tabular reduction with short supports. Proceedings of IJCAI 2013, 573-579.
[12] J Mairy, Y Deville and C Lecoutre (2015). The Smart Table Constraint. Integration of AI and OR Techniques in Constraint Programming. CPAIOR 2015. Lecture Notes in Computer Science, 9075, 271-287. DOI: http://dx.doi.org/10.1007/978-3-319-18008-3_19.
[13] H Verhaeghe, C Lecoutre, Y Deville and P Schaus (2017). Extending Compact-Table to Basic Smart Tables. Principles and Practice of Constraint Programming. CP 2017, Lecture Notes in Computer Science, 10416, 297-307. DOI: http://dx.doi.org/10.1007/978-3-319-66158-2_19.
[14] A Schneider and B Choueiry (2018) PW-CT: Extending Compact-Table to Enforce Pairwise Consistency on Table Constraints. CP 2018, Lecture Notes in Computer Science, 11008, 345-361. DOI: https://doi.org/10.1007/978-3-319-98334-9_23.
[15] A Zuenko and A Fridman (2009). Development of n-tuple algebra for logical analysis of databases with the use of two-place predicates. Journal of Computer and Systems Sciences International, 48(2), 254-261. DOI: http://dx.doi.org/10.1134/S1064230709020099.
[16] A Zuenko (2018). 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), 45. DOI: 10.1145/3207677.3277959
[17] A Zuenko (2020). Representation and Processing of Qualitative Constraints Using a New Type of Smart Tables. Proceedings of the 4th International Conference on Computer Science and Application Engineering (CSAE '20), 45, 1–7, DOI: https://doi.org/10.1145/3424978.3425023.
[18] G Møller (1995). On the Technology of Array-Based Logic. Ph. D. thesis, http://www.arraytechnology.com/documents/lic.pdf.