Algorithm for identifying generalised technical contradictions in experiments

Lei Lin, Ivana Rasovska, Roland de Guio, Sébastien Dubois

Algorithm for identifying generalised technical contradictions in experiments

Lei Lin[1], Ivana Rasovska[2], Roland De Guio[1], Sébastien Dubois[1]

1. LGECO, INSA de Strasbourg
24, bd de la Victoire
67084 Strasbourg Cedex, France
{lei.lin ;roland.deguio ;sebastien.dubois}@insa-strasbourg.fr

2. ICube UMR 7357 - Laboratoire des sciences de l'ingénieur, de l'informatique et de l'imagerie, Université de Strasbourg
300 bd Sébastien Brant - BP 10413 - 67412 Illkirch Cedex, France
ivana.rasovska@insa-strasbourg.fr

ABSTRACT. *This paper proposes an algorithm for identifying and extracting generalised technical contradictions from experiments, Contradictions are used as the starting point for certain approaches to inventive problem solving. These methods are based on dialectical contradictions, which are used for understanding and representing inventive problems. The resolution of such problems is actually the resolution of the identified contradictions, which cannot be solved by optimisation methods and must be solved using inventive principles. The proposed algorithm uses the output data from design of experiments based on statistical theory as input data. The objective of this algorithm is to identify the complete set of generalised technical contradictions.*

RESUME. *Ce papier propose un algorithme d'identification et d'extraction des contradictions techniques généralisées utilisées comme point de départ de certaines démarches de résolution de problème d'invention. Ces approches basées sur la dialectique identifient les contradictions afin de mieux comprendre et représenter les problèmes d'invention. La résolution du problème passe ensuite par la résolution de la contradiction qui empêche a priori de résoudre le problème par le biais des méthodes d'optimisation et nécessite l'utilisation des principes de résolution inventives. L'algorithme proposé utilise comme entrée les données issues d'une méthode statistique connue dans le domaine technique comme le plan d'expérience. L'objectif de l'algorithme est d'identifier l'ensemble complet des contradictions techniques généralisées. Dans le papier nous présentons le problème combinatoire, sa complexité et évaluons ses limites sur des exemples.*

KEYWORDS: *TRIZ, generalised technical contradiction, design of experiments, binary programming.*

MOTS-CLES : *TRIZ, contradiction technique généralisée, plan d'expériences, programmation binaire.*

10.3166/JESA.47.1-26 © 2013 Lavoisier

1. Introduction

Inventive design involves special problem-solving principles for proposing new creative solutions that satisfy design requirements. Among these special problem-solving methods is the theory of inventive problem solving (TRIZ) proposed by Altshuller (1988). According to Savransky (2000), the TRIZ ideology is based on two major ideas: contradiction and ideality. Contradiction is the basic law of materialist dialectics and is linked to technical problem formulation, whereas ideality is the essence of idealism and is related to the final solution. Both of these concepts should be consciously included in any inventive problem-solving process. As a dialectical based theory, the TRIZ attributes the existence of system evolution problems to a set of contradictions between system parameters. The principal motivations for the use of dialectical based approaches are twofold: identify contradictions to better understand and express inventive design problems and to view the procedure of inventive problem solving as a procedure of contradiction resolution. At the beginning of the problem-solving process, the contradictions must be clearly defined and comprehensible. Indeed, a misguided contradiction choice may lead to a decrease in the efficiency or even effectiveness of the problem-solving process. Hence, contradiction extraction and interpretation play a dominant role in the dialectical problem-solving approach.

Despite the fact that different models of contradictions were introduced by Altshuller (1988) based on patent analysis, there are gaps in procedures for contradiction gathering and representation. The concept of representing a design problem as a network of contradictions and using semantic rules to drive design using this network was introduced in (Cavallucci et *al.* , 2005). A formal definition of contradiction and its potential variations was proposed in (Rousselot et *al*., 2012). The concept of a "contradiction cloud" as a three-value graphical representation of a set of elementary contradictions was presented in (Cavallucci et *al.* , 2008). Another algorithm for extracting the most important contradiction in a network of problems (Khomenko et *al*., 2007) was proposed in (Baldussu et *al*., 2011). Almost all of these proposals are based on technical contradictions. A technical contradiction that occurs between two system parameters is a contradiction in which the improvement of one parameter leads to a degradation in the other. Given a set of situations, a TRIZ technical contradiction between two parameters Y1 and Y2 exists if each time parameter Y1 meets the design objectives, the parameter Y2 does not and vice versa.

Searching for this type of contradiction is simple because the goal is to find pairs of parameters that are never satisfied at the same time. In actuality, this classical TRIZ model of technical contradictions does not always exist. Indeed, the general trend that is observed can be summarised as follows: the more experiments and knowledge about a system there are, the lower the chance of there being a technical contradiction is (i.e., an input for inventive problem-solving methods). Moreover, this model takes into account only two evaluation parameters. Let us suppose that there exists a technical contradiction, we are able to solve it, and we cannot say anything about the satisfaction of the other evaluation parameters. Such a case exemplifies the interest in generalised technical contradictions (GTCs). The GTC

model, which was presented in (Rasovska et *al*., 2008), replaces the two evaluation parameters that exhibit a classical technical contradiction with two concepts of evaluation parameters. A concept consists of an evaluation parameter or a logical disjunction of several evaluation parameters. One parameter can only participate in one of the two concepts involved in a GTC. The desired result is the simultaneous satisfaction of the two concepts. In each concept, there is at least one or more evaluation parameters, the solution of each generalised technical contradiction should satisfy all of the evaluation parameters associated with the two concepts. Thus, the result will be better than that in the case of classical technical contradictions.

To extract contradictions, information about the technical system of interest is required. Our method uses the outputs of a design of experiments (DoE), which is a statistical optimisation method that can be used to describe relations between the input and output parameters of a system and/or optimise the output of a system for a given set of input parameters (Montgomery, 2001). A theoretical bridge between the DoE and generalised technical contradictions was presented in (Dubois et *al.*, 2009a).

In an effort to design operational methods using this link, this paper proposes an exact algorithm for identifying the entire set of generalised technical contradictions that occur in the design of experiments (DoE). To the best of our knowledge, no exact algorithm has been developed to address this problem. First, the principals of the DoE and generalised technical contradictions are illustrated using the concrete example of an electrical circuit breaker. The research space of generalised technical contradictions is defined by the number of system parameters and the number of experiments involved. The difficulty of resolving the problem is discussed, and we will show that the problem is NP-hard. The theoretical assumptions and the binary programming model used to extract the generalised technical contradictions and the search algorithm are introduced using time and space complexity analysis. The final section of the paper introduces the results of the real case of an electrical circuit breaker. Moreover, the interpretation of the results for the general population of binary matrices is presented in accordance with the principles of the design of experiments and with the aid of statistical analysis methods. Finally, different experiments in which our algorithm can be implemented are discussed to demonstrate the algorithm's time consumption and its limitations.

2. Problem description

2.1. Design of experiments and generalised system of contradictions

The design of experiments (DoE) (Montgomery, 2001) is based on the analysis of experimental data. For the purposes of our study, DoE results were transformed and summarised into a binary rectangular matrix. Because generalised technical contradictions are not related to action parameters, the binary matrix can be reduced to its evaluated form denoted Z, as shown in figure 1a. The rows of the matrix represent the experiments, and the columns correspond to the evaluation parameters.

The generic term z_{ij} of matrix Z equals 1 when the experiment e_i meets the design requirements for the evaluation parameter y_j and 0 otherwise. The design goal is to satisfy all of the evaluation parameters, i.e., to find any row of ones. When it is not possible to obtain such output using only the action parameters, inventive methods must be implemented.

$$\begin{array}{c} \\ e_1 \\ e_2 \\ \cdots \\ e_m \end{array} \begin{array}{c} \begin{array}{cccc} y_1 & y_2 & \cdots & y_n \end{array} \\ \begin{pmatrix} z_{11} & z_{12} & \cdots & z_{1n} \\ z_{21} & z_{22} & \cdots & z_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ z_{m1} & z_{m2} & \cdots & z_{mn} \end{pmatrix} \end{array}$$

(a)

	y1	y2	y3	y4	y5	y6
e1	1	0	1	1	1	1
e2	0	1	0	0	1	1
e3	1	0	1	0	0	0
e4	1	1	1	1	0	0
e5	1	0	1	0	1	1
e6	0	1	0	1	1	1
e7	1	0	1	0	0	0
e8	1	0	0	1	1	1
e9	0	1	0	1	1	1

(b)

Figure 1. Generic matrix Z (a) and the example of an electrical circuit breaker (b)

The binary matrix used in our case is illustrated using the example of an electrical circuit breaker. The circuit breaker is a simple technical system whose components are shown in figure 2.

Figure 2. Components of electrical circuit breaker

When an overload occurs, the overload creates a force (due to magnetic and electrical fields), which operates a piece called a firing pin. The firing pin opens the circuit by activating the switch located in the circuit breaker. In the case of a high overload, the firing pin—a plastic stem—cannot be reused. The problem has been studied and the main evaluation parameters defined as follows:

y1: circuit breaker disrepair (satisfied – 1, unsatisfied – 0)

y2: circuit breaker reusability (satisfied – 1, unsatisfied – 0)

y3: spring core mounting (satisfied – 1, unsatisfied – 0)

y4: firing pin bobbin mounting (satisfied – 1, unsatisfied – 0)

y5: normal mode release (satisfied – 1, unsatisfied – 0)

y6: firing pin initial position return (satisfied – 1, unsatisfied – 0)

The system's behaviour under the design parameters was modelled by a DoE, the outputs of which were transformed into the binary matrix Z shown in Fig. 1b.

This experiment does not exhibit any classical TRIZ contradiction. Indeed, each pair of evaluation parameters is simultaneously satisfied in at least one experiment (for instance, y1 and y2 in experiment e4, as shown in figure 3a). Nevertheless, there always exists a generalised technical contradiction (Dubois et *al.*, 2009b). An example of a GTC is illustrated in figure 3b: when the first concept of evaluation parameters "y1 and y3 fit requirements" is satisfied, the second concept "y2 and y5 fit requirements" is not satisfied and vice versa.

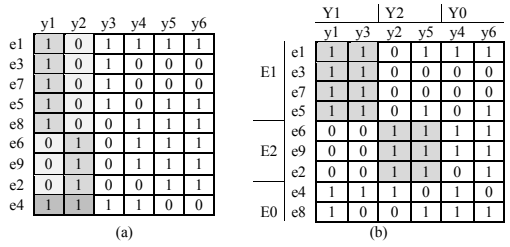


Figure 3. Example of false classical technical contradiction (a) and generalised technical contradiction (b)

Identifying GTCs in the customised DoE table representation requires defining the properties of GTCs. GTCs can be characterised by the set of definitions that allow for the extraction of GTCs from a DoE. Let us suppose that there is no experiment satisfying all of the evaluation parameters at the same time. Identifying a GTC in such a DoE involves searching for:

1. Three sets of evaluation parameters $\left(Y_1, Y_{2,} Y_0\right)$ whose union is the entire set of evaluation parameters and whose intersection is an empty set.

2. Three sets of experiments $\left(E_1, E_{2,} E_0\right)$ whose union is the entire set of experiments and whose intersection is an empty set.

3. The satisfaction of the first set of evaluation parameters Y_1 for the first set of experiments E_1 and the satisfaction of the second set of evaluation parameters Y_2 for the second set of experiments E_2.

4. At least one evaluation parameter from the second set of parameters Y_2 that is not satisfied in each experiment of the first and third sets of experiments E_1 and E_3.

5. At least one evaluation parameter from the first set of parameters Y_1 that is not satisfied in each experiment of the second and third set of experiments E_2 and E_3.

2.2. Formulating generalised technical contradiction

Using the requirements described above, we can formulate the GTC as shown in figure 4.

	Y_1	Y_2	Y_0
E_1	$E_1 \times Y_1 :$ $z_{ij} = 1$	$\forall e_i \in E_1$ $e_i \times Y_2 : \exists j, z_{ij} = 0$	
E_2	$\forall e_i \in E_2$ $e_i \times Y_1 : \exists j, z_{ij} = 0$	$E_2 \times Y_2 :$ $z_{ij} = 1$	
E_0	$\forall e_i \in E_0$ $e_i \times Y_1 : \exists j, z_{ij} = 0$	$\forall e_i \in E_0$ $e_i \times Y_2 : \exists j, z_{ij} = 0$	

Figure 4. Generalised technical contradiction expressed in DoE

We denote the original DoE result by matrix Z (the rows are the experiments; the columns are the responses). Figure 4 shows the matrix of the DoE derived by grouping the columns and the rows. The matrix has been divided into 9 blocks, and in the figure, we have formulated the features into blocks. In the blocks E1×Y1 and E2×Y2, all of the elements are equal to 1. In the remaining 4 blocks associated with Y1 and Y2, there must be at least one element equal to 0 in each row.

2.3. Difficulty of problem resolution - NP-hard problem

To identify the generalised technical contradictions, we are first interested in the size of our search space. In other words, we are interested in answering the following question: how many candidate solutions can we find when searching for generalised technical contradictions? If this problem is viewed as a searching problem, we can use a brute force algorithm to browse the solution space by an enumeration method. Although enumeration is feasible for some small matrices, it can be difficult or even impossible for large matrices. In fact, the search space grows exponentially with the number of rows M and number of columns N of the matrix Z. The following section discusses the analysis of the complexity of a search space.

Quantifying a search space is mainly a combinatorial problem because all possible matrix blocks can be expressed using Stirling numbers of the second kind (Brualdi, 2009). In actuality, the sets Y_0, E_0 may be empty. Therefore, row combinations and column combinations can be addressed using two types of combinatorial models. In the first case, when $Y_0 = \phi$, the problem can be described as a type of Ball-Box Matrix (Brualdi, 2009) , which describes the situation in which N (N>2) different balls are placed in two identical boxes and neither box is empty. In this case, the number of combinations, expressed as a Stirling number of the

second kind, is $s(N,2)$. In the second case, when $Y_0 \neq \phi$, the problem can be described by a Ball-Box Matrix that describes the situation in which N different balls are placed in three identical boxes and no box is empty; however, here, we must select 2 of 3 boxes Y_1 and Y_2. Thus, the result is $s(N,3)\times C_2^3$. The same applies to the column number; thus, the number of combinations of column numbers is $s(N,3)\times C_2^3 + s(N,2)$. In addition, when we permute two column sets and two row sets for $E_1 \times Y_1$ and $E_2 \times Y_2$, only two distinct cases arise. To summarise, we conclude that the search space size is

$$\left(s(M,3)\times C_2^3 + s(M,2)\right)\times\left(s(N,3)\times C_2^3 + s(N,2)\right)\times 2.$$

According to the equation for Stirling numbers of the second kind [11],

$$s(N,k) = \frac{1}{k!}\sum_{t=0}^{k}(-1)^t\binom{k}{t}(k-t)^N \tag{1}$$

$$s(N,2) = 2^{N-1} - 1 \tag{2}$$

Using equations (1) and (2) and supposing $N \geq 3, M \geq 3$, we obtain the size of our search space:

$$\left(\frac{3^N+1}{2} - 2^N\right)\times\left(\frac{3^M+1}{2} - 2^M\right)\times 2 \tag{3}$$

Clearly, the browsing of the search space is an NP-hard problem.

3. Problem formulation

3.1. Transformation of the generalised technical contradiction research problem into a binary integer program problem

Our problem is similar to several problems in different research areas. The problem can be classified as a data mining problem (methods and models used in data mining are reviewed in (Larose, 2006). Data mining is the analysis of observational data sets to find relationships between parameters and to represent the data in understandable and useful ways for the data owner. Our problem is also quite similar to seriation and matrix reordering problems, which seek to reorder objects into a sequence along a one-dimensional continuum (Liiv, 2010) by exploratory combinatorial data analysis techniques. Our problem can also be approached using two-mode clustering methods (Mechelen et *al*., 2004), (Marcotorchino, 1987). Such methods provide a simultaneous clustering of the rows and columns of a rectangular

data matrix but cannot restrict other parts of the matrix. All available methods associated with the above-mentioned approaches can be used to find one or several unique solutions to our problem, but not one can find all of them. In this paper, we use the binary integer programming model to analyse our problem, which results in a specific algorithm providing all of the solutions of this problem.

Identifying GTCs is generally considered a problem of defining how to group the columns and rows of matrix Z, which is a yes-or-no decision problem that can be transformed into a binary integer program, a linear programming problem in which all of the variables are restricted to binary values (Chen et *al*., 2010). Specifically, in BIP, each variable can only take a value of 1 or 0 denoting yes or no. Thus, a BIP can be used to model real-world situations using logical expression. In our problem, a BIP is used to express decision variables that indicate whether the columns or rows pertain to the columns and rows of a matrix block.

Suppose that matrix Z has M rows and N columns. Let us denote the parts of a 3-partition of the columns and rows as $\{Y_1, Y_2, Y_0\}$ and $\{E_1, E_2, E_0\}$, respectively. Our goal is to find all of the 3-partitions of the column set and 3-partitions of the row set that meet the requirements indicated in figure 4.

First, we use two N-dimensional binary vectors $C_{i=1,2}$ to denote the two sets of columns (Y_1, Y_2). If one vector component of C_1 (resp. C_2) equals 1, the corresponding column belongs to the column set Y_1 (resp. Y_2); otherwise, it does not belong to Y_1 (resp. Y_2). Second, we use two M-dimensional binary vectors $R_{i=1,2}$ to denote the two sets of rows (E_1, E_2). If one vector component of R_1 (resp. R_2) equals 1, the corresponding row belongs to the row set E_1 (resp. E_2). The vectors are as follows:

$$C_{i=1,2} = \left(c_1^i, c_2^i, c_3^i, .., c_N^i\right)_N,\ R_{i=1,2} = \left(r_1^i, r_2^i, r_3^i, ..., r_M^i\right)_M;$$

$C_{i=1,2}$ represents $Y_{i=1,2}$, $R_{i=1,2}$ represents $E_{i=1,2}$.

For convenience, we assume that $N_1 = \left(\sum_j c_j^1\right)$, $N_2 = \left(\sum_j c_j^2\right)$, $M_1 = \left(\sum_j r_j^1\right)$, $M_2 = \left(\sum_j r_j^2\right)$.

Our problem can then be transformed into an integer programming problem. The restrictions regarding block $E_1 \times Y_1$ and $E_2 \times Y_2$ indicated in figure 4 can become an objective function, as $R_i Z C_i^T$ is equal to the sum of all elements in the block $E_i \times Y_i$. $N_i M_i$ is the product of the number of rows and columns of the block

(E_i, Y_i) .When the blocks (E_1, Y_1) and (E_2, Y_2) are full of "ones", the quantity $\sum_{i=1,2} R_i Z C_i^T - \sum_{i=1,2} N_i M_i$ is equal to zero and negative otherwise. It is clear that the maximum value of this equation is zero.

The restriction for the block $E_2 \times Y_1$ shown in figure 4 indicates that the cardinality of Y_1 is greater than the sum of the elements in each row of the block $E_2 \times Y_1$. Thus, we have the inequalities for each row of $E_2 \times Y_1$:

$$N_1 - \left(\sum_{j:c_j^1=1, c_j^1 \in C_1} z_{ij} \right) \geq 1.$$

Using the same method to model the block $E_1 \times Y_2$, $E_0 \times Y_2$ and $E_0 \times Y_1$ provide three additional groups of inequalities. It should be noted that the membership of a row to E_0 is determined by the complementary set $E_1 \cup E_2$.

3.2. Definition of integer programming problem

To summarise, the search problem subject to the restrictions shown in figure 4 becomes a problem of calculating all of the solutions of the following integer programming problem:

Decision variables:

$$C_1 = \left(c_1^1, c_2^1, c_3^1, \ldots c_n^1\right), C_2 = \left(c_1^2, c_2^2, c_3^2, \ldots c_n^2\right), R_1 = \left(r_1^1, r_2^1, r_3^1, \ldots r_m^1\right),$$
$$R_2 = \left(r_1^2, r_2^2, r_3^2, \ldots r_m^2\right).$$

Input parameter: matrix Z

Objective function:

$$Maximise \sum_{i=1,2} R_i Z C_i^T - \sum_{i=1,2} N_i M_i \tag{4}$$

subject to

$$N_1 - \left(\sum_{j:c_j^1=1, c_j^1 \in C_1} z_{ij} \right) \geq 1, \forall i \in \left\{ i \mid r_i^2 = 1, r_i^2 \in R_2 \right\} \tag{5}$$

$$N_2 - \left(\sum_{j:c_j^2=1, c_j^2 \in C_2} z_{ij} \right) \geq 1, \forall i \in \left\{ i \mid r_i^1 = 1, r_i^1 \in R_1 \right\} \tag{6}$$

$$N_1 - \left(\sum_{j:c_j^1=1, c_j^1 \in C_1} z_{ij} \right) \geq 1, \forall i \in \left\{ i \mid r_i^2 = 0, r_i^2 \in R_2, r_i^1 = 0, r_i^1 \in R_1 \right\} \tag{7}$$

$$N_2 - \left(\sum_{j:c_j^2=1, c_j^2 \in C_2} z_{ij} \right) \geq 1, \forall i \in \left\{ i \mid r_i^2 = 0, r_i^2 \in R_2, r_i^1 = 0, r_i^1 \in R_1 \right\} \tag{8}$$

$$R_1 + R_2 \leq (1,1,\ldots,1)_M \tag{9}$$

$$C_1 + C_2 \leq (1,1,\ldots,1)_N \tag{10}$$

$$c_i^j = 0,1; r_i^j = 0,1 \tag{11}$$

$$C_1, C_2, R_1, R_2 \neq 0 \tag{12}$$

This BIP is nonlinear because the object function is a polynomial; however, the program can be transformed into a linear one according to the procedure described in reference (Chen et *al*., 2010).

3.3. Relaxation of binary programming problem into sub-problems

The binary programming problem described above cannot be solved by any ordinary method. On the one hand, the search space grows exponentially with the parameters N and M; therefore, the problem cannot be solved by brute force searching. On the other hand, our goal is to obtain all solutions meeting the maximum number of conditions; thus, the basic approach to linear BIPs, such as branch and bound or branch and cut, are invalid, giving rise to the design of a new algorithm for solving this problem.

	Y_1	Y_0
E_1	$\forall (i,j) \in E_1 \times Y_1; z_{ij} = 1;$	
E_0	$\forall i \in E_0, \exists j \in Y_1; z_{ij} = 0;$	

Figure 5. Sub-problem blocks in matrix Z

According to the observations made above for the BIP, the restrictions of variables $(C_1, R_1), (C_2, R_2)$ are symmetrical, and the result must be the same by exchanging the Y_1, Y_2 and E_1, E_2; thus, we want to simplify the binary problem to a set of sub-problems, the merging of which provides the solution to the original problem. The generic search sub-problem is illustrated in figure 5.

We use the vectors $R_1 = (r_1, r_2, \ldots, r_M)_M$, $C_1 = (c_1, c_2, c_3, \ldots c_N)_N$ to denote Y_1 and E_1 in figure 5. If the vector component is equal to 1, the corresponding column or row belongs to Y_1 or E_1.

For convenience, we assume that $M_1 = \left(\sum_j r_j \right)$ and that $N_1 = \left(\sum_j c_j \right)$. Our goal is to seek the block expressed by (E_1, Y_1) in which all of the elements in the block are equal to 1.

We can use the same approach used previously to define the object function: $R_1 Z C_1^T - M_1 N_1$, the maximum value of which is zero. Moreover, its accompanying block $(\overline{E}, Y)$ underneath has at least one element equal to 0 for each row, which can be formulated by the inequalities: $N_1 - \left(\sum_{j:c_j=1, c_j \in C_1} z_{ij} \right) \geq 1, i \in \{ i \mid r_i = 1, r_i \in R_1 \}$.

Decision variables:

$$R_1 = (r_1, r_2, \ldots, r_M)_M, C_1 = (c_1, c_2, c_3, \ldots c_N)_N$$

Input parameter: matrix Z;

$$\text{Maximise: } R_1 Z C_1^T - M_1 N_1 \tag{13}$$

Subject to:

$$N_1 - \left(\sum_{j:c_j=1, c_j \in C_1} z_{ij} \right) \geq 1, i \in \{ i \mid r_i = 0, r_i \in R_1 \} \tag{14}$$

$$r_i = 0,1; c_i = 0,1 \tag{15}$$

$$R_1 \neq \phi ; C_1 \neq \phi$$

After all of the results of the sub-problems are gathered, we need to combine the blocks to solve the original problem. The combinatorial strategy is to examine all of the combinations of the two results of the sub-problem with the following rule: If the intersection of their row sets and column sets are empty ($E_1 \cap E_2 = \phi$, $Y_1 \cap Y_2 = \phi$), the blocks (E_1, Y_1) and (E_2, Y_2) constitute a solution of the original problem. Correctness of the strategy: if the solution sets of two problems include each other, they are equal.

4. Problem resolution

4.1. Search algorithm for the sub-problem

Now, we describe the method for addressing the sub-problem. We can generally search the pattern along two directions: along columns and rows. Because our problem has a restriction regarding $\overline{E}$, we search the blocks row by row.

The searching procedure consists of five steps:

1. Let i =1.
2. Select the columns' set, the elements of which equal 1 in row i.
3. Select one subset of the columns' set defined in step 2. If the column set was previously selected in another row, reselect the subset. If not, mark the column set as having been checked.
4. Store this column set as the block column set, and add row number i to the row set of the block. Look up the row k, k>i, if it has the same column set in which elements equal 1, add the row number in the block row set.
5. Output block as one solution: if the column set in row i does not finish traversing, return to 3. Else, if i=M, the algorithm terminates; otherwise increment i by 1, and return to 2.

In step 3, another problem that must be addressed is traversing the subsets of subsets. The problem is essentially a Generating All n-Tuples Problem. Equivalently, we want to visit all n-Tuples $(y_1, y_2, \ldots y_n)$ where each $y_i = 0 \text{ or } 1$. One can find more details in reference (Knuth, 2011).

In the following, we explain why the search algorithm can obtain all of the solutions of the sub-problem. First, we prove that the blocks are solutions of the sub-problem. It is explicit that the elements in block are equal to 1; thus, they meet the requirements of the objective function. Equivalently, it is clear that each row in block $(\overline{E}, Y)$ has at least one element equal to 1; if not, according to the algorithm, the row whose elements are equal 1 will be added to the block row set. Second, we prove that the algorithm can gather all of the solutions. Suppose that there exists a block that is the solution of the sub-problem. Thus, all of its elements are equal to 1

to meet the requirements of the objective function, and each row in the block $\left(\overline{E}, Y\right)$ has at least one element equal 1 to satisfy the restriction. The block's row number set must have its smallest element ranked above the others; when the algorithm operates on this row, the block column set pertains to this row column set, as long as it has not been previously checked. Thus, this algorithm yields all of the solutions.

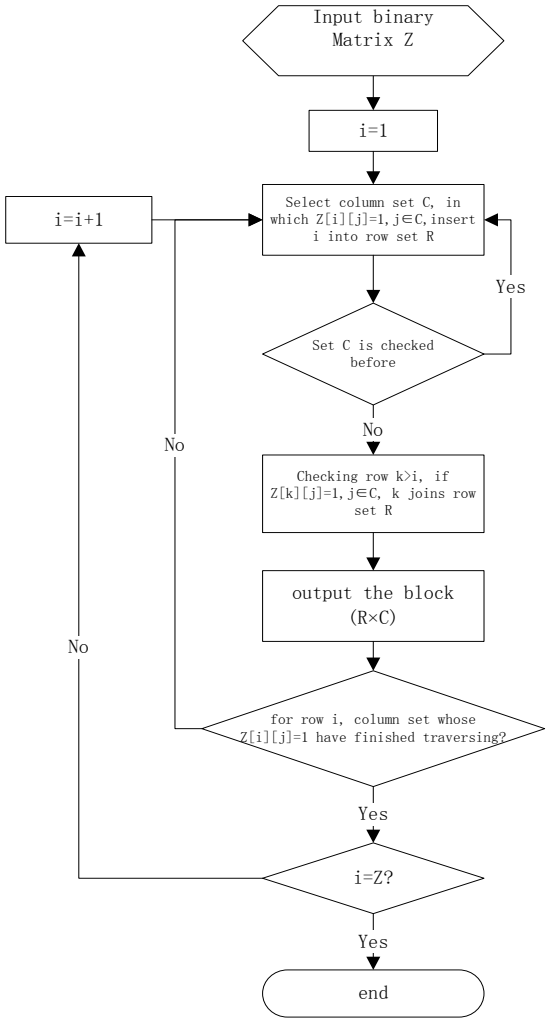


Figure 6. Flow chart of algorithm

4.2. Time and space complexity analysis

We presume that the maximum number of elements equal to 1 in each row is n and the row number is M ; thus, the time consumed by visiting all of the possible combinations of elements equal to 1 is $O\left(M \times 2^n\right)$. Moreover, we must scan the remaining rows (fewer than M rows remain). Therefore, the time complexity in the searching phase is $O\left(M \times M \times 2^n\right)$. Suppose that the number of blocks in the searching phase is S ; thus, time complexity in the comparing phase is $O\left(\frac{S(S-1)}{2}\right) = O\left(S^2\right)$. In conclusion, the time complexity is $O\left(M^2 \times 2^n + S^2\right)$.The principal space cost is related to the storage of the blocks in the searching step. Because the number of blocks is lower than 2^N, the space complexity is $O\left(2^N\right)$

5. Illustration of results in the case of electrical circuit breaker

The purpose of this section is twofold: in the first part, we will use a circuit breaker example to illustrate the various concepts of technical contradictions presented in the paper. The algorithm used on the concrete case helps us to measure the number of technical contradictions as well as generalised technical contradictions in the real case. In the second part, we will examine a sample of matrices with the same number of rows and columns as the matrix considered in the electrical circuit breaker example but with different numbers of evaluation parameters equal to 1. This study is done to confirm the real case results and to measure and compare different possible but quite similar cases. In our future research we want to compare the results of our automatic approach with the results of human expert's analysis. For the moment our study points out the complexity of such a search and relatively small number of found technical contradictions versus a big quantity of found generalised technical contradictions. Furthermore, the effect of the density of ones (the number of satisfied evaluation parameters) on the number of technical and generalised technical contradictions will be studied.

5.1. Generalised technical contradictions

The first set of experiments concerned the search for generalised technical contradictions in the case of an electrical circuit breaker (figure 7). We identified 117 generalised technical contradictions within the 341 possible pairs of concepts. As an example, we discuss four different GTCs, which can be interpreted as follows.

GTC #1 represents the contradiction between the circuit breaker reusability y2 and both the spring core mounting y3 and normal mode release y5. GTC #2 represents the contradiction between the following pairs of parameters: the circuit

breaker reusability y2 and normal mode release y5 and the couple circuit breaker disrepair y1 and spring core mounting y3. GTC #3 represents the contradiction between the couple circuit breaker reusability y2 and firing pin initial position return y6 and the couple spring core mounting y3 and normal mode release y5. Finally, GTC #4 represents the contradiction between the couple circuit breaker reusability y2 and firing pin initial position return y6 and the rest of the evaluation parameters.

	y2	y3	y5	y1	y4	y6
e2	1	0	1	0	0	1
e4	1	1	0	1	1	0
e6	1	0	1	0	1	1
e9	1	0	1	0	1	1
e1	0	1	1	1	1	1
e5	0	1	1	1	0	1
e3	0	1	0	1	0	0
e7	0	1	0	1	0	0
e8	0	0	1	1	1	1

GTC #1

	y2	y5	y1	y3	y4	y6
e2	1	1	0	0	0	1
e6	1	1	0	0	1	1
e9	1	1	0	0	1	1
e1	0	1	1	1	1	1
e3	0	0	1	1	0	0
e4	1	0	1	1	1	0
e5	0	1	1	1	0	1
e7	0	0	1	1	0	0
e8	0	1	1	0	1	1

GTC #2

	y2	y6	y3	y5	y1	y4
e2	1	1	0	1	0	0
e6	1	1	0	1	0	1
e9	1	1	0	1	0	1
e1	0	1	1	1	1	1
e5	0	1	1	1	1	0
e3	0	0	1	0	1	0
e4	1	0	1	0	1	1
e7	0	0	1	0	1	0
e8	0	1	0	1	1	1

GTC #3

	y2	y6	y1	y3	y4	y5
e2	1	1	0	0	0	1
e6	1	1	0	0	1	1
e9	1	1	0	0	1	1
e1	0	1	1	1	1	1
e3	0	0	1	1	0	0
e4	1	0	1	1	1	0
e5	0	1	1	1	0	1
e7	0	0	1	1	0	0
e8	0	1	1	0	1	1

GTC #4

Figure 7. Four examples of GTCs in the case of an electrical circuit breaker

5.2. Relationship between TC /GTC and density of ones

We randomly created 377 matrices that were similar to the circuit breaker matrix except with respect to their density of ones: 9 rows and 6 columns with no column full of zeros and no row full of ones. Figure 8 shows the evolution of the number of TCs/GTCs with the density of ones. Some trends can be observed: the number of TCs decreases with the density and tends to zero at a density of up to 50%. The average number of GTCs increases up to a density of 50% and then decreases. The mean values of each population are correlated to observe the general trend. The points indicated by a star point in figure 8 represent the real characteristics of the electrical circuit breaker matrix; the results confirm that this example is not an exception.

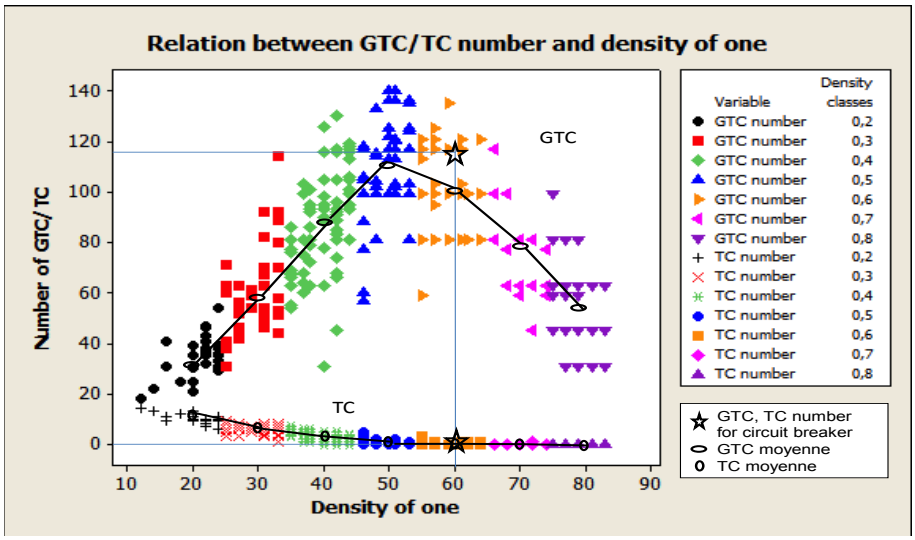


Figure 8. TC/ GTC number versus density for 9 x 6 matrices

To study the dispersion of the GTC and TC number in greater detail, the two box plots shown in figure 9 were created. The plots show the interval extent, the interquartile box, the linked mean values, and the abnormal points for each population.

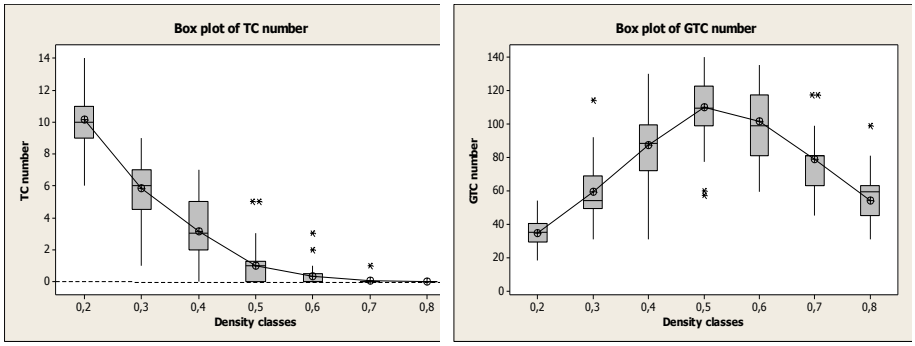


Figure 9. Relation between GTC/TC number and density of ones in general population

6. Factors affecting GTC and TC number: generalisation

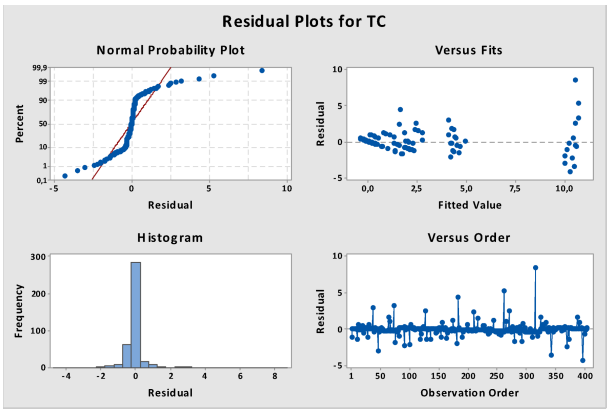
As mentioned before the results of our algorithm shows the big complexity in the search of technical and generalised technical contradictions. In some conditions we can be faced with the situation that there is no technical contradiction and so we cannot solve the inventive problem overcoming no existing technical contradiction. On the contrary, at the same time there is a lot of generalised technical contradictions and the human expert is not able to deal with so many possibilities. Then we should choose in our future work the "best" set of generalised technical contradictions to solve the inventive problem. For this we will identify the important factors influencing the number of technical and generalised technical contradictions in order to reduce this complexity.

This section reveals how the number of columns (parameters) and rows (experiments) and the density of ones (number of satisfied parameters) affect the number of found TCs and GTCs. We generated several populations of random matrices according to the three previously mentioned influencing factors. Each factor has three levels (i.e., 6, 9, and 13 columns—50, 100, and 200 rows—20%, 50%, and 80% density). Finally, for each combination, 15 matrices with no line full of ones and no column full of zeros were created randomly. Overall, the sample contained a total of 405 different matrices.

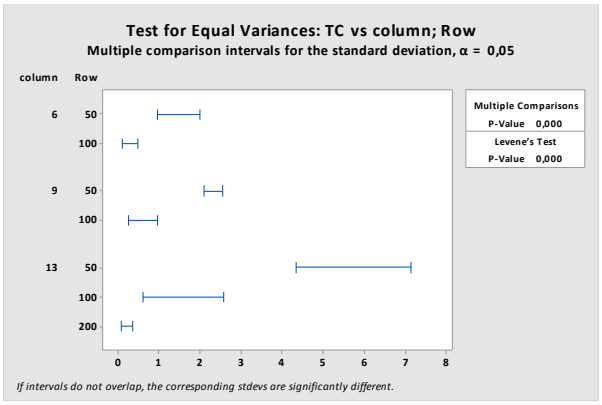
6.1. Factors affecting the number of TCs in the general population

To identify the main factors affecting the number of TCs, we used the general complete factorial design of experiments and variance analysis. Three factors were studied, i.e., the column number, the row number, and the density of ones in the matrix. The results obtained for the design of experiments revealed three different populations and thus three Henry's curves (figure 10a). Each population should be treated independently because the populations do not show similar behaviour (their variances are not the same, as shown in figure 10b). There is a large variance in the number of TCs for the matrices with a small number of rows, and the variance in the number of TCs also increases with the number of columns. The variance in the density was not studied because technical contradictions were identified only for the case of 20% density and the number of TC varied from 0 to 19. The matrices with 200 rows and the matrices with densities greater than 50% almost never featured any TCs.

With the aid of an interaction diagram, a graphical representation of the main effects, and the ANOVA table, different hypotheses were tested for each factor. As shown in figure 11a, the column number has a positive factual effect on the number of possible technical contradictions, but it is correlated with the negative effect of the number of rows and the density, especially for the matrices with fewer than 100 rows and a density less than 50%. At densities greater than 50%, the number of technical contradictions is always zero; thus, other effects are counteracted against. These results confirm intuition and seem to generalise the average limiting zone for the existence of TCs.

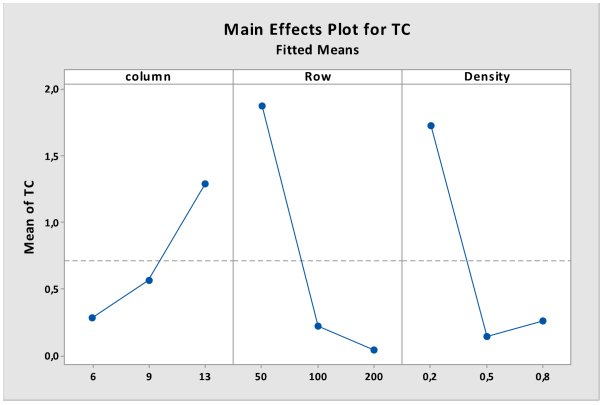


(a)

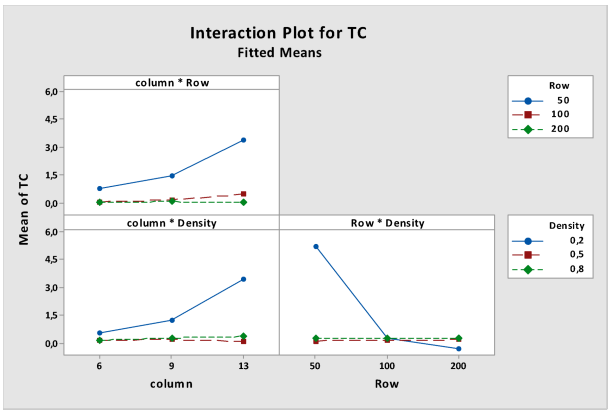


(b)

Figure 10. Residual values (a) and test of the variance equality (b) for TC number



(a)

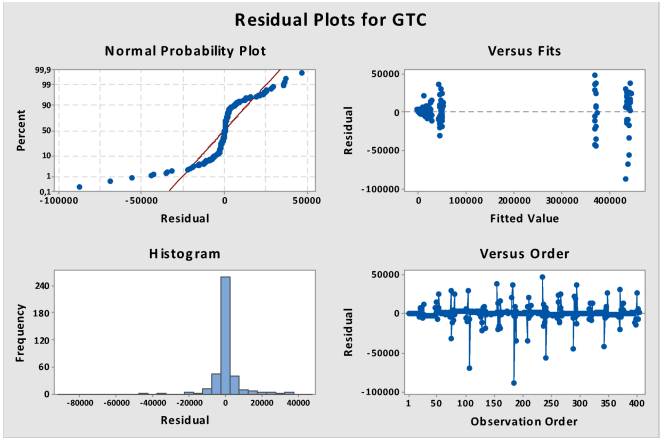


(b)

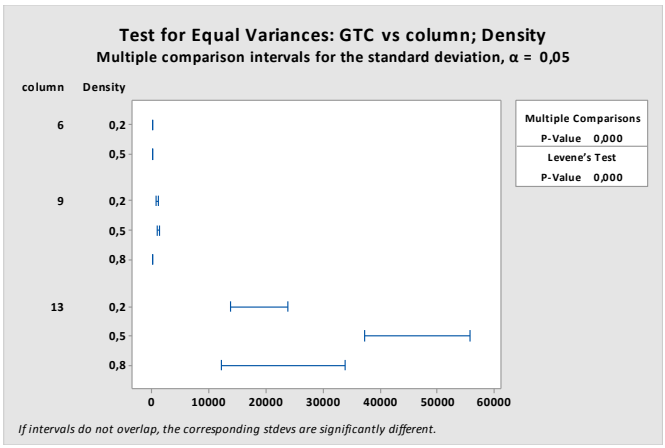
Figure 11. Principal effects of factors (a) and interactions between the factors (b) for TCs

6.2. Factors affecting the number of GTCs in the general population

We identified the main influencing factors for the number of GTCs using the same procedure followed for TC case. The general complete factorial design of experiments and variance analysis were used to study three factors: the column number, the row number, and the density of ones in the matrices. As in the case of the TCs, three different populations and thus three Henry's curves (figure 12a) were detected. Their variances were compared by the test of variance equality, as shown in figure 12b. The variance in the number of GTCs increases with the number of columns, whereas the variance does not change with row number. The variance is smaller at densities of 20% and 80% but large at a density of 50%.

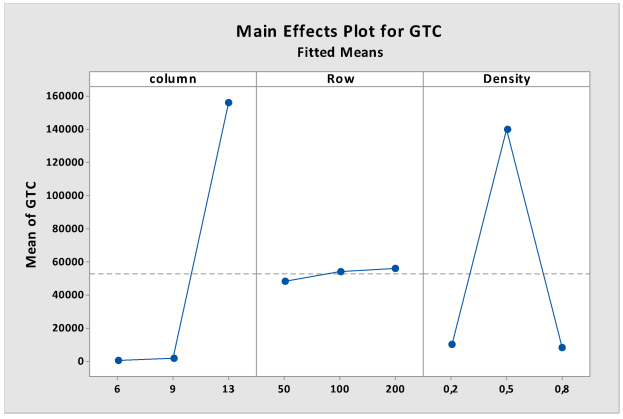


(a)

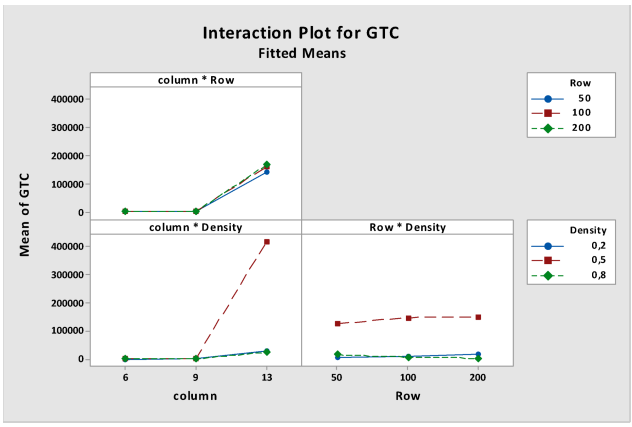


(b)

Figure 12. Residual values (a) and test of the variance equality (b) for GTC



(a)



(b)

Figure 13. Principal effects of factors
(a) and interactions between the factors (b) for GTCs

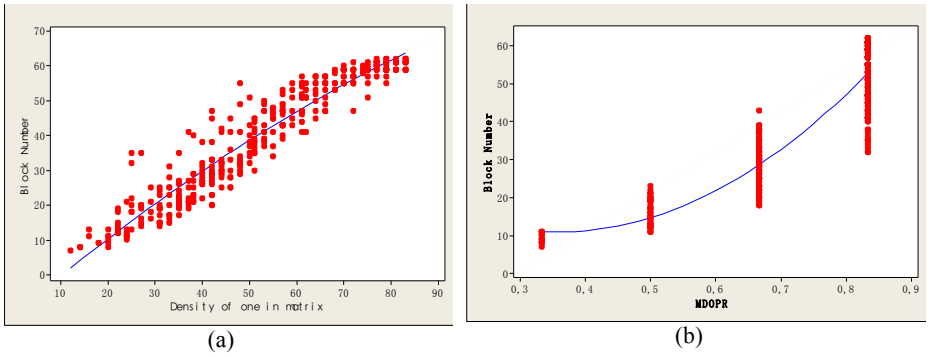
The variance tests confirm the results of the main effect analysis and interaction diagram shown in figure 13. As shown, the number of rows does not have a significant effect on the number of GTCs. The most positive significant effect is the number of columns. The second most significant effects are the density and the interaction between the density and the number of columns. It appears that the shape of the number GTC versus density curve does not change with the number of columns and rows: it increases up to a density of 50% and decreases thereafter. This behaviour will have an impact on our future strategy for filtering for GTCs.

7. Time consumption of the algorithm

To estimate the algorithm's time consumption for a given matrix, we consider that the block number mainly affects the searching time. In the following, we analyse the interaction between the block number, the density of ones, and the maximum density of ones per row for the general population of matrices similar to the matrix for the electrical circuit breaker.

7.1. Relationship between block number and density of ones in the case of the electrical circuit breaker

In the case of the electrical circuit breaker, we studied the relationship between the number of blocks, which indicates the time consumption of the algorithm and the density of ones in the matrix. Figure 14a shows that the number of blocks and thus the time consumption increase almost linearly with the density of ones, by monitoring the mean values. In figure 14b, the number of blocks is plotted as a function of the maximum density of ones per row in the matrix. For convenience, we use MDOPR to denote the maximum density of ones per row. In summary, for a given matrix similar to that of the electrical circuit breaker, the most influential factor affecting the algorithm's time consumption is block number. Moreover, the results of the experiments indicate that the time consumption is approximately proportional to the density of ones in the matrix.

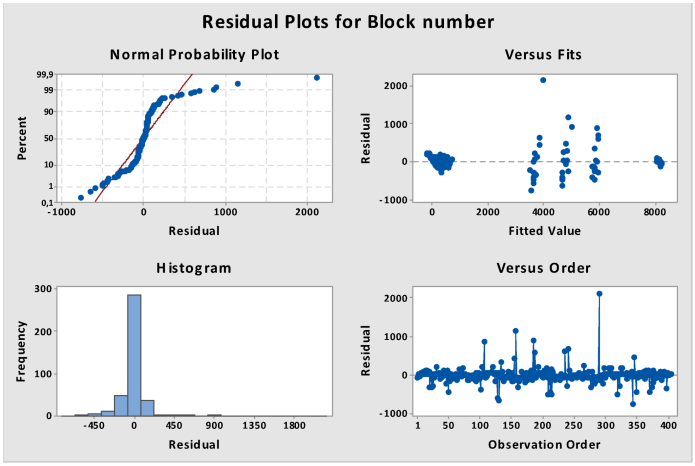


(a) (b)

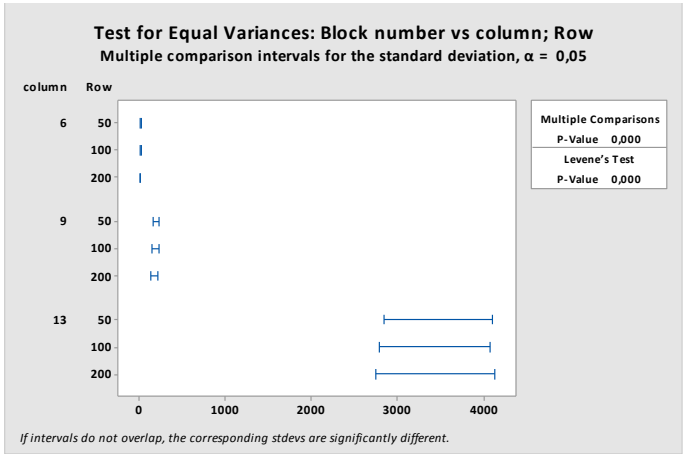
Figure 14. Scatter plots of block number versus density of ones (a) and versus maximum density of ones per row (b)

7.2. Relationship between block number and NC/NR/density in the general population

We identified the main influencing factors for the number of blocks in the same manner as that for TCs and GTCs. The general complete factorial design of experiments and variance analysis were used to study three factors: the column number, the row number, and the density of ones. Three different populations and thus three Henry's curves (figure 12a) were detected; their variances were compared by the test of variance equality, as shown in figure 12b. The variance in the number of block increases with the number of columns. The variance does not change with different row numbers and increases only slightly with the density of ones.



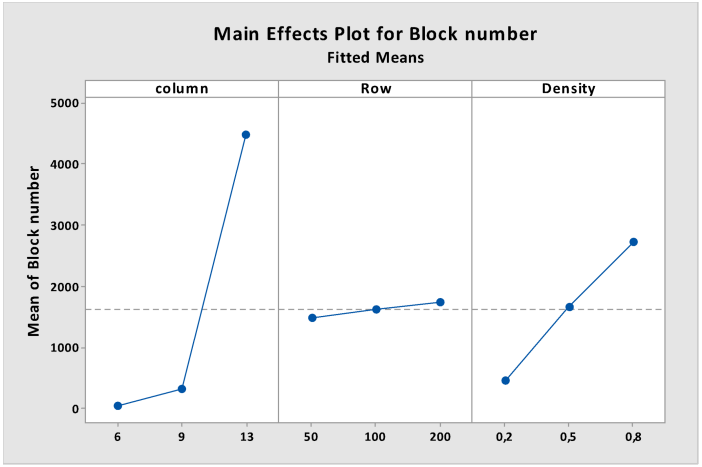
(a)



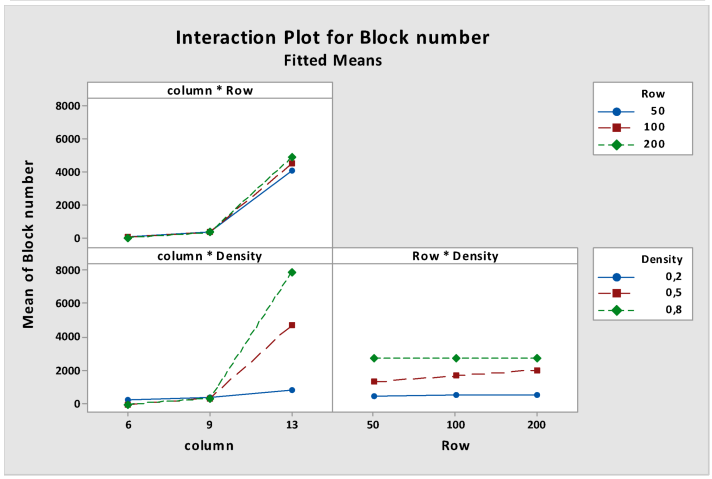
(b)

Figure 15. Residual values (a) and test of the variance equality (b) for block number

We cannot conclude that the number of rows has a significant effect on the block number. The most positive significant effects are the number of columns and the density. The interaction between the density and the number of columns also has a significant effect.



(a)



(b)

Figure 16. Principal effects of factors
(a) and interactions between the factors (b) for block number

7.3. Large-scale matrices and time consumption of our algorithm

This part of the study concerns large-scale matrices and the relation between the number of satisfied evaluation parameters (equal to 1) and the time cost of the algorithm. We used matrices in which each element is equal to 1 owing to the

dominating effect of the number of evaluation parameters whose values equal 1 on the time complexity. We performed the test for five large-scale matrices and present the time consumption —the matching time and searching time— in Table 1. The test was performed on five matrices containing only ones as the elements and with an increasing number of evaluation parameters (columns). The number of matrix blocks is proportional to the maximum of satisfied parameters in each row.

Table 1. Experimental Results

	Num of Columns	Matching time(s)	Num of Blocks	Searching time(s)
Data1	5	0.007	31	0.190
Data2	8	0.240	255	0.197
Data3	10	3.632	1023	0.234
Data4	12	50.159	4095	0.401
Data5	15	3666.167	32767	2.283

Data 1 has 5 evaluation parameters (columns) and 1500 experiments (rows). Data 2 has 8 evaluation parameters and 1500 experiments. Data 3 has 10 evaluation parameters and 1500 experiments. Data 4 has 12 evaluation parameters and 1500 experiments. Data 5 has 15 evaluation parameters and 1500 experiments. The computer used to run the tests was 64-bit machine with 8G of memory and a dual-core CPU operating at 3.4 GHz.

From Table 1, we draw the following conclusion: the time cost grows exponentially with the number of evaluation parameters equal to 1; the time consumed in the searching block is less than that consumed in the matching block. Thus, in these experiments, the time cost is determined by the matching phase. Because we need to store all of the blocks in memory during the matching phase, the space consumption is beyond the capability of the experimental computer (8G) when the number of evaluation parameters is greater than 15 and the number of experiments is equal to 1500.

8. Conclusions

In the design of a technical system, two types of optimisation problems are confronted: one can be solved by improving the engineering parameters of existing systems, and the other concerns the reconstruction of the technical system. When the design objective cannot be attained by improving the stated parameters, we must use inventive problem-solving principles. In this study, we used the principle based on the dialectical approach, which identifies contradictions to understand and represent design problems better. The design problem-solving procedure involves the resolution of the best contradiction.

Our goal was to identify the complete set of generalised technical contradictions. The results of the experiments indicate that the more technical parameters and experiments describing the technical system there are, the greater the number of

possible combinations there is. Thus, a human expert is not able to identify and consider all of the combinations, which is why we tried to automate the search for generalised technical contradictions. The method is reliable when the maximum number of evaluation parameters is less than 15. The matching time is the main factor affecting the algorithm's speed. Thus, to improve the algorithm, we should find some restriction for discarding blocks before the matching phase. This problem can also be classified as a data mining problem and is quite similar to seriation and matrix reordering problems, such as the two-mode clustering problem; however, these methods do not propose to find all possible solutions.

Thanks to the algorithm it becomes possible to evaluate quantitatively the volume of contradictions and check the hypothesis about rarity of TC existence. In the experimental section of this paper, first is provided the result of our algorithm for the case of an electrical circuit breaker. It confirms the large number of GTCs. Second an analysis of the number of GTCs and TCs versus density of ones for the population of binary matrices with the same number of rows and columns than the circuit breaker was performed. The result shows that the average number of GTCs increases with density until 50% and then decreases, while the number of TCs decreases with density and is generally equal to zero after 60%. The values of TC and GTCs of the circuit breaker example are consistent with the results of this general population of matrices. Third, in order to generalise previous results, we evaluated the impact of the number of columns, density and rows on the number of GTCs and TCs. The analysis of the results shows that, for a given number of rows and columns, the average value of GTCs and TCs evolve in the same way than for the circuit breaker (i.e. the average number of GTCs increases with density until 50% and then decreases, while the number of TCs decreases with density and is generally equal to zero after 60%). Nevertheless, the number of columns acts as a shape factor. For instance, the maximum value of GTCs, which is obtained for density equal to 50%, increases exponentially with the number of columns. The main factors affecting the number of technical contradictions and generalised technical contradictions was identified with the aid of the principles of the design of experiments and statistical analysis tests. These results will have an impact on our strategy for solving inventive design problems and selecting the most appropriate technical or generalised technical contradictions to solve such problems.

Once all generalised technical contradictions are identified, several elimination strategies are subsequently needed to choose only the GTCs leading to resolution of the design problem. Two different criteria were proposed in [18] to select the most meaningful generalised technical contradictions. Nevertheless, generalised technical contradictions provide only a partial solution in TRIZ problem solving. To resolve the contradictions, we need more details about the system parameters and the relationships thereof. The next step in our research will be the identification of physical contradictions according to the identified GTCs.

References

Altshuller G.S. (1988*). Creativity as an Exact Science*. Gordon and Breach, New York.

Baldussu A., Becattini N., Cascini G. (2011). Network of contradictions analysis and structured identification of critical control parameters. *Procedia Engineering*, vol. 9, p. 3-17.

Brualdi R.A. (2009). Chapter 8 Special Counting Sequences*., Introductory combinatorial*, The 5th edition, New Jersey, Pearson, p. 274-291.

Cavallucci D., Khomenko N., Morel C. (2005). Towards inventive design through management of contradictions. *CIRP International Design Seminar*, Shanghai, China.

Cavallucci D., Rousselot F., Zanni C. (2008). On contradiction clouds. *Proceeding of the TRIZ Future conference*, Enschede, Netherlands.

Chen D., Batson R.G., Dang Y. (2010). *Applied Integer Programming - Modeling and Solution*. John Wiley & Sons, New York.

Dubois S., Rasovska I., De Guio R. (2009a). Interpretation of a General Model for Inventive problems, the Generalized System of Contradiction. *CIRP Design Conference 2009*, Cranfield, England.

Dubois S., Eltzer T., De Guio R. (2009b). A dialectical based model coherent with inventive and optimization problems. *Computer in Industry*, vol. 60, n° 8, p. 575-583.

Khomenko N., De Guio R., Lelait L., Kaikov I. (2007). A framework for OTSM-TRIZ Based Computer Support to be used in Complex Problem Management*. International Journal of Computer Applications in Technology,* vol. 30, n° 1-2, p. 88-104.

Knuth D. E. (2011). Chapter 7 Combinatorial Searching. *The art of computer programming. volume 4A combinatorial algorithm.* Addison-Wesley Professional, Boston, p.281-319.

Larose D. T. (2006*). Data mining methods and models*. John Wiley & Sons, New York.

Liiv I. (2010). Seriation and matrix reordering methods: An historical overview. *Statistical Analysis and Data Mining*, vol. 3, n° 2, p. 70-91.

Marcotorchino F. (1987). A unified approach of the Block-Seriation Problems. *Journal of Applied Stochastic Models and Data Analysis*, vol 3, n° 2.

Mechelen I.V., Bock H.H., De Boeck P. (2004). Two-mode clustering methods: a structured overview. *Statistical Methods in Medical Research*, vol.13, n° 2, p. 363-394.

Montgomery D.C. (2001). *Design and Analysis of Experiments*, 5th edition. John Wiley & Sons, New York.

Rasovska I., Dubois S., De Guio R. (2010). Study of different principles for automatic identification of generalized system of contradictions out of design of experiments. *The 8th international conference of modelling and simulation*, Hammamet, Tunisia.

Rousselot F., Zanni-Merk C., Cavallucci D. (2012). Towards a formal definition of contradiction in inventive design. *Computers in Industry*, vol. 63, n° 3, p. 231-242.

Savransky S.D. (2000*). Engineering of Creativity Introduction to TRIZ Methodology of Inventive Problem Solving*. Boca Raton, Florida.