Journal of Mathematical Modelling and Algorithms **3:** 263–283, 2004.

A Meta-heuristic with Orthogonal Experiment for the Set Covering Problem

JINGPENG LI[1] and RAYMOND S. K. KWAN[2]
[1]*School of Computer Science & Information Technology, The University of Nottingham, Nottingham NG8 1BB, U.K. e-mail: jpl@cs.nott.ac.uk*
[2]*School of Computing, University of Leeds, Leeds LS2 9JT, U.K. e-mail: rsk@comp.leeds.ac.uk*

Abstract. This paper reports an evolutionary meta-heuristic incorporating fuzzy evaluation for some large-scale set covering problems originating from the public transport industry. First, five factors characterized by fuzzy membership functions are aggregated to evaluate the structure and generally the goodness of a column. This evaluation function is incorporated into a refined greedy algorithm to make column selection in the process of constructing a solution. Secondly, a self-evolving algorithm is designed to guide the constructing heuristic to build an initial solution and then improve it. In each generation an unfit portion of the working solution is removed. Broken solutions are repaired by the constructing heuristic until stopping conditions are reached. Orthogonal experimental design is used to set the system parameters efficiently, by making a small number of trials. Computational results are presented and compared with a mathematical programming method and a GA-based heuristic.

Mathematics Subject Classifications (2000): 90-08 computational methods.

Key words: set covering, fuzzy subset, evolutionary algorithm, scheduling.

1. Introduction

The Set Covering Problem (SCP) has been proven to be NP-complete [7], which means that large problem instances become intractable for exact solution methods. This problem has important applications in many fields such as resource allocation, crew scheduling, pattern recognition and machine learning.

The SCP is to find a subset of the columns of an $m \times n$ zero-one matrix

$$A = \{a_{ij} \in \{0, 1\};\ i = 1, 2, \ldots, m;\ \ j = 1, 2, \ldots, n\}$$

that covers all rows at a minimum cost, based on a set of costs $\{c_j;\ j = 1, 2, \ldots, n\}$. Defining $x_i = 1$ if column j (with an associated cost $c_j > 0$) is in the solution and $x_j = 0$ otherwise, the SCP is to

$$\text{Minimize} \quad \sum_{j=1}^{n} c_j x_j \tag{1}$$

$$\text{Subject to:} \quad \sum_{j=1}^{n} a_{ij} x_j \geqslant 1, \quad i \in I = \{1, 2, \ldots, m\}, \tag{2}$$

$$x_j = 0 \text{ or } 1, \quad j \in J = \{1, 2, \ldots, n\}. \tag{3}$$

Constraint (2) ensures that each row is covered by at least one column, and (3) requires that whole columns are used.

Integer Linear Programming (ILP) is the traditional exact method for the SCP, which often encounters computational difficulty for large problems. Heuristic approaches are therefore popular in recent years, in an effort to solve the SCP near-optimally within reasonable time. A number of attempts have been made by using the Lagrangian-based heuristics [2, 3, 9, 5]. Sen [17] presents an approach based on simulated annealing, and Ohlsson et al. [16] develop a mean field feedback algorithm based on artificial neural network. Genetic algorithms (GAs) have also been extensively studied to solve this problem. For example, Beasley and Chu [4] present a GA-based heuristic with several modifications to the basic genetic procedures; Solar et al. [19] use a parallel GA to interchange information between parallel searches; Aickelin [1] gives an indirect approach where the GA is used to map the solution space, and a separate decoding routine then built solutions to the original problem.

As an extension of our early work [13–15] on driver scheduling, this paper presents a meta-heuristic for large-scale SCPs, which consists of three stages. In the first stage, five factors characterized by fuzzy membership functions are aggregated to evaluate the structure and generally the goodness of a column. This evaluation function is incorporated into a refined greedy algorithm to evaluate all candidate columns, and then to decide which column is going to be selected in the process of constructing a solution.

In the second stage, a self-evolving algorithm is designed to guide the above constructing heuristic to build an initial solution and then improve it further. In each generation an unfit portion of the working solutions is removed. Broken solutions are repaired by the constructing heuristic. By carrying out iterative improvement and constructive perturbation, this algorithm has the ability to escape from local minima.

The above two stages involve a number of evaluation weights and several other parameters, which will influence the system performance significantly and are difficult to determine. Therefore, the third stage is to set these influencing parameters efficiently by using the method of orthogonal experimental design [21]. This method uses orthogonal arrays to study the wide range of parameter space, with a small number of experiments.

This paper is organized as follows: Section 2 describes the method of fuzzy evaluation for the SCP; Section 3 presents a self-evolving algorithm to control the

searching process; Section 4 introduces the method of orthogonal experimental design in detail; Section 5 gives experimental results on real world problems, and finally Section 6 gives some concluding remarks.

2. Fuzzy Evaluation for Set Covering

A building heuristic for a possible solution is inherently sequential. Among the large number of columns to be chosen, it is difficult to judge which one is more effective than others because the criteria bear some uncertainty. Fuzzy set theory [23] is therefore used to assign each column a quantitative value according to its structural state. The fitter the structure of a column, the larger its value is.

2.1. CONSTRUCTION OF THE FACTOR SET

The main factors concerning the column's structure are the number of rows it covers (u_1), its cost (u_2), the ratio of its number of covered rows to its cost (u_3), the average coverage number of all the rows covered by this column (u_4), and the fractional cover of the relaxed LP solution (u_5).

2.1.1. *Factor* u_1

The objective for set covering is to minimize the total cost of the solution. In many real-world problems, the total cost is usually increased with the number of columns in the solution. To reduce the number of columns used as much as possible, the number of rows each column covers should be taken into account as a criterion.

Considering two columns that cover a different number of rows, one column should be more efficient than another because it covers more rows and thus potentially minimises the number of columns in the final solution. If every column in the solution covers as many rows as possible, it can be concluded that, on average in such a solution, the number of columns is fewer thus leading to a lower total cost. Hence we assume that the goodness of a column j ($j \in J$) generally increases with the covering number of rows R_j.

Furthermore, since in large-scale SCP problems only a small proportion of the columns will be used to build a solution, it is not desirable to have larger variations in the measure of goodness among these elite columns. On the contrary, for columns that cover more rows, their goodness should increase as smoothly as possible, allowing more chances to be selected later. Based on this consideration, this increase should be non-linear. Thus, an S-shape quadratic membership function μ_1 ($j \in J$) is applied to define factor u_1 as

$$\mu_1 = \begin{cases} 2\left(\dfrac{X_1 - d_1}{b_1 - d_1}\right)^2, & d_1 \leqslant X_1 < \dfrac{b_1 + d_1}{2}, \\ 1 - 2\left(\dfrac{X_1 - b_1}{b_1 - d_1}\right)^2, & \dfrac{b_1 + d_1}{2} \leqslant X_1 \leqslant b_1, \end{cases} \tag{4}$$

where $X_1 = R_j = \sum_{i=1}^{m} a_{ij}$, $\forall j \in J$; $b_1 =$ the maximum number of rows contained in a column; $d_1 =$ the minimum number of rows contained in a column.

2.1.2. *Factor* u_2

To achieve the objective of minimum total cost, the cost of an individual column in the solution should be as small as possible. Therefore, factor u_2 is based on the assumption that the goodness of a column j ($j \in J$) generally decreases with its cost c_j. In our experiments, a linear functions μ_2 ($j \in J$) is used to define factor u_2 as

$$\mu_2 = \frac{b_2 - X_2}{b_2 - d_2}, \quad \forall j \in J, \tag{5}$$

where $X_2 = c_j$; $b_2 =$ maximum cost; $d_2 =$ minimum cost.

2.1.3. *Factor* u_3

According to the above analysis for factor u_1 and factor u_2, to achieve a satisfactory solution, the ideal columns to be used would be those covering as many rows as possible with the smallest possible cost. Unfortunately, in practice these two factors are often contradictory because a column covering more rows usually has a larger cost.

To get a balance between the number of covered rows and the cost for a column, besides these absolute values, the relative ratio of the number of covered rows to its cost can be regarded as a third important criterion. Based on the assumption that the goodness of a column j ($j \in J$) generally increases with the ratio of the number of covered rows to its cost, denoted as R_j/c_j, the membership function μ_3 ($j \in J$) for factor u_3 is therefore defined as

$$\mu_3 = \begin{cases} 2\left(\dfrac{X_3 - d_3}{b_3 - d_3}\right)^2, & d_3 \leqslant X_3 < \dfrac{b_3 + d_3}{2}, \\ 1 - 2\left(\dfrac{X_3 - b_3}{b_3 - d_3}\right)^2, & \dfrac{b_3 + d_3}{2} \leqslant X_3 \leqslant b_3, \end{cases} \quad \forall j \in J, \tag{6}$$

where $X_3 = R_j/c_j = \sum_{i=1}^{m} a_{ij}/c_j$; $b_3 =$ maximum ratio; $d_3 =$ minimum ratio.

2.1.4. *Factor* u_4

Considering all the columns with respect to all the rows to be covered, each row i has an associated coverage list containing L_i columns that are able to cover i, where

$$L_i = \sum_{j=1}^{n} a_{ij}, \quad i \in I. \tag{7}$$

Normally, each column covers at least one row, and the number of rows contained in column j is $\sum_{i=1}^{m} a_{ij}$. Therefore, for all the rows covered by column j, the total coverage number is $\sum_{i=1}^{m}(a_{ij} \times \sum_{j=1}^{n} a_{ij})$, and the average coverage number, denoted as α_j, is formulated as

$$\alpha_j = \sum_{i=1}^{m}\left(a_{ij} \times \sum_{j=1}^{n} a_{ij}\right) \Big/ \sum_{i=1}^{m} a_{ij}, \quad j \in J. \tag{8}$$

The average coverage number is an index whether the components in a column are heavily covered by other columns in general. To find an economic cover, it is reasonable to regard the columns whose rows are heavily covered by others to be less important. Based on this assumption, the goodness of column j ($j \in J$) generally decreases with the average coverage number of all the rows covered by column j. Thus, the membership function $\mu_4(j \in J)$ for factors u_4 is defined as

$$\mu_4 = \frac{b_4 - X_4}{b_4 - d_4}, \quad \forall j \in J, \tag{9}$$

where $X_4 = \alpha_j$; $b_4 =$ maximum average coverage number; $d_4 =$ minimum average coverage number.

2.1.5. *Factor* u_5

The traditional method for column selection is ILP, which is NP-hard [7]. Large problems would have to be divided into sub-problems, and in some cases the ILP process may have difficulties in finding an integer solution. In contrast, the relaxed fractional problem in which the solution vector is not required to be integral is much easier: the optimal solution for the relaxed problem can be found in polynomial time [10]. In addition, Srinivasan [20] showed that the approximation guarantee for the Randomised Rounding Algorithm (RRA) on fractional covers is

$$c_{\mathrm{RRA}} \leqslant c_{\mathrm{Min}}\left(\ln\left(\frac{m}{c_{\mathrm{Min}}}\right) + \ln\ln\left(\frac{m}{c_{\mathrm{Min}}}\right) + \mathrm{O}(1)\right). \tag{10}$$

Here c_{RRA} is the number of sets in the subcover output by the RRA, and c_{Min} is the optimum value of the relaxed LP. Although Slavík [18] proved the performance guarantee of the RRA was no better than that of the simple greedy algorithm, it still can be concluded that the relaxed solution provides some useful information about the distribution of the optimal integer solution. With the advantage of being smooth and nonzero at all points, a Gaussian distribution function μ_5 ($j \in J$) is applied to define the fractional cover factor as

$$\mu_5 = \begin{cases} \mathrm{e}^{-\frac{(X_5-\alpha)^2}{\beta}}, & \text{if column } j \text{ is in the fractional cover;} \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

Let $\mu_5 = 1$ when $X_5 = b_5$, and $\mu_5 = 0.01$ when $X_5 = d_5$, where $X_5 =$ fractional value of column j in the relaxed LP solution; $b_5 =$ maximum value in fractional cover; $d_5 =$ minimum value in fractional cover.

Therefore,

$$\begin{cases} \alpha = b_5, \\ \beta = -\dfrac{(b_5 - d_5)^2}{\ln 0.01}. \end{cases} \tag{12}$$

2.2. FUZZY EVALUATION

Considering various factors that influence the structure of a column, fuzzy evaluation uses aggregation operators to evaluate the efficiency of this column. This paper applies a commonly used arithmetic mean operator, by which the structural coefficient $f_1(j)$ for column j can be formulated as

$$f_1(j) = \sum_{k=1}^{5} (w_k \times \mu_k), \quad \forall j \in J, \tag{13}$$

where w_k ($w_k \geqslant 0$) denotes the corresponding weights for factor u_k ($k = 1, 2, 3, 4, 5$), satisfying

$$\sum_{k=1}^{5} w_k = 1. \tag{14}$$

The main task in the above model is to find a suitable weight distribution among the fuzzy membership functions. These five weights, along with two other parameters to be given in Sections 3.3 and 3.4, are determined by the method of orthogonal experimental design, which will be described in Section 4.

3. A Self-evolving Algorithm

In this section, a self-evolving algorithm is used to mimic generations of evolution of a single solution. Inspired by the idea of Kling and Banerjee [11] for the placement problem in electronic circuit design, this approach has been tailored to the bus and train driver scheduling problem successfully [13]. It executes a sequence of *Reconstruction*, *Evaluation*, *Selection* and *Mutation* steps in a loop until a stopping condition is reached or no improvement has been achieved for a number of iterations. Throughout the evolution, the currently best solution is retained and finally returned as the final solution. Let us explain these four steps in detail.

3.1. RECONSTRUCTION

The *Reconstruction* step takes a partial solution as the input, and produces a complete solution as the output. While the existing columns in the partial solution

remain unaffected, the *Reconstruction* step is to assign columns to all the uncovered rows to complete a partial solution. Note that in the first iteration of the loop, the partial solution is empty.

Considering all columns with respect to all rows to be covered, each of the remaining unassigned rows i has a coverage list of length L_i, i.e. containing L_i possible columns that can cover it. The greedy-based constructor assumes that the desirability of adding column j ($j \in J$) into the partial solution generally increases with its function value $F'(j)$, formulated as

$$F'(j) = f_1(j) \times \sum_{i \in I'} a_{ij}, \quad \forall j \in J, \tag{15}$$

where $f_1(j)$ is the structural coefficient defined as formula 13), and I' is the set of rows to be covered. However, to introduce diversification, one of the candidates, not necessarily the top candidate, is randomly selected from a Restricted Candidate List (RCL) consisting of columns with r largest function values $F'(j)$. In our experiment, r is set to be 3.

Let $J^* = \{1, 2, \ldots, t\}$ the set of columns in a partial solution, and $S_j = \{i \mid a_{ij} = 1, i \in I\}$ the set of rows covered by column j, the steps to generate a complete solution are:

Step 1 Set $I' = I - \bigcup(S_{j^*} : j^* \in J^*)$.

Step 2 If $I' = \phi$ then stop: J^* is a complete solution and $C(J^*) = \sum(c_{j^*} : j^* \in J^*)$. Otherwise locate a row $t' \in I'$ having $L_{t'} = \min(L_{i'} : i' \in I')$, and then randomly select a column S_k, $k \in \{1, \ldots, r\}$, within RCL from the coverage list of row t'. Proceed to Step 3.

Step 3 Add k to J^*, set $I' = I' - S_k$, and return to Step 2.

Before the *Reconstruction*, some rows may already be over-covered, i.e. covered more than once by the existing columns in the partial solution. The other columns added by the *Reconstruction* step are each chosen to cover at least one currently uncovered row, but they increase the amount of over-cover as well. Thus, some columns might become redundant later, causing all their rows covered by other columns. In the *Selection* step, these redundant columns will be removed automatically because of their zero goodness.

3.2. EVALUATION

The overall evaluation function $F(j^*)$ consists of two parts: structural coefficient $f_1(j^*) \in [0, 1]$ and over-cover penalty $f_2(j^*) \in [0, 1]$, which is formulated as

$$F(j^*) = f_1(j^*) \times f_2(j^*), \quad \forall j^* \in J^*, \tag{16}$$

where is the ratio of the non-overlapped number of rows to total number of rows in j^* ($j^* \in J^*$), which is formulated as

$$f_2(j^*) = \sum_{i=1}^{m} b_{ij^*} \Big/ \sum_{i=1}^{m} a_{ij^*}, \quad \forall j^* \in J^*, \tag{17}$$

where

$$b_{ij^*} = \begin{cases} 0, & \sum_{j^* \in J^*} a_{ij^*} > 1; \\ 1, & \text{otherwise.} \end{cases} \tag{18}$$

If every row in j^* has been covered by one or more other columns in J^* as well, then $f_2(j^*) = 0$; conversely if none of the rows in j^* is over-covered, $f_2(j^*) = 1$.

3.3. SELECTION

This step is to determine whether a column j^* ($j^* \in J^*$) is retained for the next generation, or discarded and placed in a queue for the new allocation. This is done by comparing its goodness $F(j^*)$ to $(p_s - k_s)$, where p_s is a random number generated for each generation in the range [0, 1], and k_s is a value smaller than 1.0. If $F(j^*) > (p_s - k_s)$ then j^* will survive in its present position; otherwise j^* will be removed from the current evolutionary solution. The rows it covers, except those also covered by other columns in the solution, are then released for the next *Reconstruction*. By using this *Selection* process, column j^* with larger goodness $F(j^*)$ have a higher probability of survival in the current solution.

The purpose of subtracting $k_s \in [0, 1]$ from p_s is to improve the SE's convergence capability. Without it, in the case of p_s close to 1, nearly all the columns will be removed from the solution, which is obviously undesirable. A suitable setting of the selection value k_s is important for the algorithm's performance.

3.4. MUTATION

Following the *Selection* step, each retained column j^* ($j^* \in J^*$) still has a chance to be mutated, i.e. randomly discarded from the partial solution at a given rate of p_m, and releases its covered rows, except those also covered by other retained columns, for the next generation. The mutation rate p_m should be much smaller than the selection rate to guarantee convergence. Like the selection valve k_s, p_m is also an influencing parameter.

4. Orthogonal Experimental Design

We have set up the membership functions and heuristic framework, in which seven parameters will influence the system performance significantly. These parameters are weight w_k for criteria u_k ($k = 1, 2, 3, 4, 5$) in the fuzzy evaluation model, selection value k_s in the *Selection* step and mutation rate p_m in the *Mutation* step. Common approaches for parameter design lead either to a long time span for trying out all combinations, or to a premature termination of the design process with results far from optimal in most cases.

Each trial with a specific parameter setting is a process of iterative improvement and thus is quite time-consuming. The time to conduct full combinations of all possible parameter values for the optimal solution is impractical. To reduce the number of experiments and still achieve satisfactory solutions, Taguchi's method of orthogonal experimental design can be applied [21].

4.1. PRELIMINARIES

Orthogonal experimental design for parameter optimization provides a systematic and efficient approach to determine near optimal parameter settings. It applies orthogonal arrays to study a large number of variables with a relatively small number of trials, significantly reducing the number of experimental configurations.

4.1.1. *Orthogonal Array*

Orthogonal arrays are a special set of Latin squares [6], constructed by Taguchi to lay out the experimental design. In this array, the columns are mutually orthogonal or balanced. That is, for any pair of columns, all combinations of factor levels occur an equal number of times. Consider an example shown in Table I.

The array is designated by the symbol $L_9(3^4)$, involving four factors A, B, C, and D, each at three levels one (1), two (2), and three (3). The array has a size of nine rows and four columns. The numbers (1/2/3) in the rows indicate the factor levels and each row represents specific test characteristics of each experiment. The vertical columns represent the experimental factors to be studied using that array. Each of the columns contains three assignments at each levels (1, 2, or 3) for the corresponding factors. These conditions combine in nine possible ways (i.e. (1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2), (3, 3)) for two factors, with 3^4 possible combinations of levels for all the four factors.

Table I. Orthogonal array $L_9(3^4)$

	A	B	C	D
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2	1	2	3
5	2	2	3	1
6	2	3	1	2
7	3	1	3	2
8	3	2	1	3
9	3	3	2	1

The orthogonal array facilitates the experimental design process by assigning factors to the appropriate columns. In this case, referring to Table I, factors A, B, C, and D are arbitrarily assigned to columns 1, 2, 3, and 4 respectively. From the table, nine trials of experiments are needed, with the level of each factor for each trial-run as indicated in the array. The experimental descriptions are reflected through the condition level. The experimenter may use different designators for the columns, but the nine trial-runs will cover all combinations, independent of column definition. In this way, the orthogonal array assures consistency of the design carried out by different experimenters.

4.1.2. *Comparison to the Traditional Method of Factorial Design*

The traditional method of factorial design is to investigate all possible combinations and conditions in an experiment that involves multiple factors. For example, if the factorial design is implemented for the four 3-level factors in Table I, the total number of trials needed would be a full combination of 81 (3^4) trials, rather than 9 trials by the orthogonal array $L_9(3^4)$.

The reason why using orthogonal arrays could find superior parameter configurations with a small number of experiments is its mutual balance. For example, in Table I, each column contains three ones, three twos, and three threes; and any pair of columns contain all combinations of levels (i.e. (1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2), (3, 3)) exactly once. It is the characteristic of mutual balance that guarantees the choice of combinations producing elite solutions.

In summary, in case that the parameter-interaction space is relatively smooth, compared with the traditional method of factorial design, Taguchi's orthogonal array is superior since:

- It is efficient in handling larger numbers of factor variables;
- It can produce similar and consistent results, even though the experiments may be carried out by different experimenters;
- It can determine the contribution of each quality-influencing factor.

4.2. THE METHOD OF ORTHOGONAL EXPERIMENTAL DESIGN

Due to of the characteristic of mutual balance in orthogonal arrays, Taguchi's approach can explore the solution space as extensively as possible. Moreover, due to the ANOVA process to be introduced later, Taguchi's approach can also exploit the solution space as much as possible.

To carry out orthogonal experimental designs, the choice of the factors is crucial. Since the seven parameters in the proposed algorithm will influence the system performance greatly, they are regarded as the control factors currently investigated in this paper. Basically, this approach consists of the three steps of defining the parameter space, determining the factor levels, and analysis of variance.

4.2.1. *Definition of the Parameter Ranges*

The first step is to define the ranges of the seven control factors in the proposed algorithm. Without any pre-knowledge about the influence of the weights w_k ($k = 1, 2, 3, 4, 5$) on the algorithm, the levels of w_k are evenly set over the full applicable range of $[0, 1]$. However, the range for the selection value k_s and the mutation rate p_s should be much narrower, since according to our past experience, the self-evolving algorithm usually yields better solutions with $k_s \in [0.20, 0.30]$ and $p_s \in [0.05, 0.06]$ respectively, and these two parameters are relatively independent of w_k.

4.2.2. *Determination of the Factor Levels*

This step is to define the initial levels of the control factors, and sequentially choose the most suitable orthogonal array. In order to facilitate the description of this step, three definitions are given as follows:

DEFINITION 1. A factor-level table is a tableau, where each row represents a control factor and each column represents an individual level.

DEFINITION 2. In a factor-level table, degrees of freedom for a factor are the number of levels of this factor minus one, and degrees of freedom for the table are the sum of degrees of freedom for all factors.

DEFINITION 3. Degrees of freedom for an orthogonal array are the number of trials minus one. For example, the orthogonal array $L_9(3^4)$ in Table I has eight degrees of freedom.

The initial levels of the individual control factors can be set arbitrarily, and the associated orthogonal array can be chosen flexibly. However, in practice the two principles below are normally complied with [21]:

(1) The degrees of freedom for the factor-level table should be no larger than the degrees of freedom for the orthogonal array to be used;
(2) The number of factors in the factor-level table should be no larger than the number of columns in the orthogonal array.

In theory, there should be as many levels for each factor as possible. However, in this situation, the number of trials needed would be too large. The most suitable orthogonal array is the one that maintains the best balance. Since the number of factors investigated has been fixed at seven, the task for the orthogonal experimental design is to determine the largest practically feasible number of levels for each factor, and the number of rows (trials) in the associated orthogonal array.

Let A be the number of levels for each factor, and L be the number of rows (trials) in the orthogonal array. Normally L is the square of an integer, denoted as

$$L = k^2, \quad k \in \{2, 3, \ldots, n\}. \tag{19}$$

According to the principle above,

$$7 \times (A-1) \leqslant L-1 \Leftrightarrow A \leqslant \frac{L-1}{7} + 1 \Leftrightarrow A \leqslant \frac{k^2-1}{7} + 1$$
$$\Rightarrow \text{if } k = 6, \text{ then } A \leqslant 6, \text{ where } k \text{ is the smallest integer to be satisfied}$$
$$\Leftrightarrow \text{if } L = 36, \text{ then } A \leqslant 6. \tag{20}$$

Therefore the largest number of levels for each factor is 6, the smallest number of trials is 36, and the most ideal orthogonal is array $L_{36}(6^7)$. Unfortunately, the authors had difficulty in finding a readily available $L_{36}(6^7)$ from related literature. Therefore, the second largest array $L_{25}(5^6)$ from [21] is applied, which is the optimal design to handle six 5-level factors.

By studying the proposed algorithm, we believe that the seventh factor p_s is relatively less important than the others due to the relatively minor role of the *mutation* step. To maintain a balance between necessary precision and number of experiments, factors w_k ($k = 1, 2, 3, 4, 5$) and k_s are finally defined to be 5 levels, and p_s to be 2 levels respectively (shown in Table II). These seven factors are assigned to the $L_{50}(2^1 \times 5^6)$ orthogonal array shown in Table III. This is an economical and efficient design for dealing with these seven factors using only 50 trials, rather than 31,250 ($2^1 \times 5^6$) experimental trials.

Table II. Control factors and their levels

Control factors	Levels				
	1	2	3	4	5
1. Weight w_1	0.1	0.3	0.5	0.7	0.9
2. Weight w_2	0.1	0.3	0.5	0.7	0.9
3. Weight w_3	0.1	0.3	0.5	0.7	0.9
4. Weight w_4	0.1	0.3	0.5	0.7	0.9
5. Weight w_5	0.1	0.3	0.5	0.7	0.9
6. *Selection* valve k_s	0.22	0.24	0.26	0.28	0.30
7. *Mutation* rate p_m	0.05	0.06	–	–	–

Table III. $L_{50}(2^1 \times 5^6)$ orthogonal array (the values in parenthesis represent the factor levels)

Trial No.	Control factors						
	1. w_1	2. w_2	3. w_3	4. w_4	5. w_5	6. k_s	7. p_m
1	0.1(1)	0.1(1)	0.1(1)	0.1(1)	0.1(1)	0.22(1)	0.05(1)
2	0.1(1)	0.3(2)	0.3(2)	0.3(2)	0.3(2)	0.24(2)	0.05(1)
3	0.1(1)	0.5(3)	0.5(3)	0.5(3)	0.5(3)	0.26(3)	0.05(1)
4	0.1(1)	0.7(4)	0.7(4)	0.7(4)	0.7(4)	0.28(4)	0.05(1)
5	0.1(1)	0.9(5)	0.9(5)	0.9(5)	0.9(5)	0.30(5)	0.05(1)
6	0.3(2)	0.1(1)	0.3(2)	0.5(3)	0.7(4)	0.30(5)	0.05(1)
7	0.3(2)	0.3(2)	0.5(3)	0.7(4)	0.9(5)	0.22(1)	0.05(1)

Table III. (Continued)

Trial No.	Control factors						
	1. w_1	2. w_2	3. w_3	4. w_4	5. w_5	6. k_s	7. p_m
8	0.3(2)	0.5(3)	0.7(4)	0.9(5)	0.1(1)	0.24(2)	0.05(1)
9	0.3(2)	0.7(4)	0.9(5)	0.1(1)	0.3(2)	0.26(3)	0.05(1)
10	0.3(2)	0.9(5)	0.1(1)	0.3(2)	0.5(3)	0.28(4)	0.05(1)
11	0.5(3)	0.1(1)	0.5(3)	0.9(5)	0.3(2)	0.28(4)	0.05(1)
12	0.5(3)	0.3(2)	0.7(4)	0.1(1)	0.5(3)	0.30(5)	0.05(1)
13	0.5(3)	0.5(3)	0.9(5)	0.3(2)	0.7(4)	0.22(1)	0.05(1)
14	0.5(3)	0.7(4)	0.1(1)	0.5(3)	0.9(5)	0.24(2)	0.05(1)
15	0.5(3)	0.9(5)	0.3(2)	0.7(4)	0.1(1)	0.26(3)	0.05(1)
16	0.7(4)	0.1(1)	0.7(4)	0.3(2)	0.9(5)	0.26(3)	0.05(1)
17	0.7(4)	0.3(2)	0.9(5)	0.5(3)	0.1(1)	0.28(4)	0.05(1)
18	0.7(4)	0.5(3)	0.1(1)	0.7(4)	0.3(2)	0.30(5)	0.05(1)
19	0.7(4)	0.7(4)	0.3(2)	0.9(5)	0.5(3)	0.22(1)	0.05(1)
20	0.7(4)	0.9(5)	0.5(3)	0.1(1)	0.7(4)	0.24(2)	0.05(1)
21	0.9(5)	0.1(1)	0.9(5)	0.7(4)	0.5(3)	0.24(2)	0.05(1)
22	0.9(5)	0.3(2)	0.1(1)	0.9(5)	0.7(4)	0.26(3)	0.05(1)
23	0.9(5)	0.5(3)	0.3(2)	0.1(1)	0.9(5)	0.28(4)	0.05(1)
24	0.9(5)	0.7(4)	0.5(3)	0.3(2)	0.1(1)	0.30(5)	0.05(1)
25	0.9(5)	0.9(5)	0.7(4)	0.5(3)	0.3(2)	0.22(1)	0.05(1)
26	0.1(1)	0.1(1)	0.1(1)	0.1(1)	0.1(1)	0.22(1)	0.06(2)
27	0.1(1)	0.3(2)	0.3(2)	0.3(2)	0.3(2)	0.24(2)	0.06(2)
28	0.1(1)	0.5(3)	0.5(3)	0.5(3)	0.5(3)	0.26(3)	0.06(2)
29	0.1(1)	0.7(4)	0.7(4)	0.7(4)	0.7(4)	0.28(4)	0.06(2)
30	0.1(1)	0.9(5)	0.9(5)	0.9(5)	0.9(5)	0.30(5)	0.06(2)
31	0.3(2)	0.1(1)	0.3(2)	0.5(3)	0.7(4)	0.30(5)	0.06(2)
32	0.3(2)	0.3(2)	0.5(3)	0.7(4)	0.9(5)	0.22(1)	0.06(2)
33	0.3(2)	0.5(3)	0.7(4)	0.9(5)	0.1(1)	0.24(2)	0.06(2)
34	0.3(2)	0.7(4)	0.9(5)	0.1(1)	0.3(2)	0.26(3)	0.06(2)
35	0.3(2)	0.9(5)	0.1(1)	0.3(2)	0.5(3)	0.28(4)	0.06(2)
36	0.5(3)	0.1(1)	0.5(3)	0.9(5)	0.3(2)	0.28(4)	0.06(2)
37	0.5(3)	0.3(2)	0.7(4)	0.1(1)	0.5(3)	0.30(5)	0.06(2)
38	0.5(3)	0.5(3)	0.9(5)	0.3(2)	0.7(4)	0.22(1)	0.06(2)
39	0.5(3)	0.7(4)	0.1(1)	0.5(3)	0.9(5)	0.24(2)	0.06(2)
40	0.5(3)	0.9(5)	0.3(2)	0.7(4)	0.1(1)	0.26(3)	0.06(2)
41	0.7(4)	0.1(1)	0.7(4)	0.3(2)	0.9(5)	0.26(3)	0.06(2)
42	0.7(4)	0.3(2)	0.9(5)	0.5(3)	0.1(1)	0.28(4)	0.06(2)
43	0.7(4)	0.5(3)	0.1(1)	0.7(4)	0.3(2)	0.30(5)	0.06(2)
44	0.7(4)	0.7(4)	0.3(2)	0.9(5)	0.5(3)	0.22(1)	0.06(2)
45	0.7(4)	0.9(5)	0.5(3)	0.1(1)	0.7(4)	0.24(2)	0.06(2)
46	0.9(5)	0.1(1)	0.9(5)	0.7(4)	0.5(3)	0.24(2)	0.06(2)
47	0.9(5)	0.3(2)	0.1(1)	0.9(5)	0.7(4)	0.26(3)	0.06(2)
48	0.9(5)	0.5(3)	0.3(2)	0.1(1)	0.9(5)	0.28(4)	0.06(2)
49	0.9(5)	0.7(4)	0.5(3)	0.3(2)	0.1(1)	0.30(5)	0.06(2)
50	0.9(5)	0.9(5)	0.7(4)	0.5(3)	0.3(2)	0.22(1)	0.06(2)

It is interesting to know how good the results derived from the above 50 trials are, when compared to all other possible combinations. Because of its mutual balance of orthogonal arrays, this performance ratio can be guaranteed by the following theorem in non-parametric statistics [22].

THEOREM 1. *Suppose random variable X is subject to a probabilistically continuous distribution $F(X)$, and $x_1, x_2, \ldots, x_n$ are simple samples (or random observation values) of X. If $x_1, x_2, \ldots, x_n$ are sorted in ascending order, denoted as $x_1 \leqslant x_2 \leqslant \cdots \leqslant x_n$, then the performance ratio for x_i $(i = 1, 2, \ldots, n)$ is*

$$E[F(x_i)] = \frac{i}{n+1}. \tag{21}$$

In particular,

$$E[F(x_n)] = \frac{n}{n+1}. \tag{22}$$

Formula (22) means that the best experimental result in these simple samples is probabilistically better than $\frac{n}{n+1}\%$ of all possible results defined in the whole discrete solution space. In this case, the best result by $L_{50}(2^1 \times 5^6)$ is better than 98.04% (= 50/51) results of all 31,250 trials.

4.2.3. *Analysis of Variance*

The contribution of individual quality-influencing factors is crucial to the control enforced on the experimental design. A statistical method, Analysis of Variance (ANOVA), is commonly used to analyse the results of the orthogonal experimental design, and to determine how much variation each factor has contributed. By studying the main effects of each factor, the general tendencies of the influencing factors can be characterized. The characteristics can be controlled, such that a lower, or a higher, value in a particular factor produces the preferred result. Thus, the levels of influencing factors to produce the best results can be predicted [21].

Since the main purpose of this paper is to test the suitability of the proposed approach for the set covering problem, the authors only perform an initial investigation about the wide range of parameter settings, and uses orthogonal experimental design to find a suitable, but coarse, range of the control factors. Therefore, the current research simply chooses the parameter configuration from Table III that leads to the best results, and skips the follow-on process of ANOVA and further experiments.

5. Computational Results

The proposed approach was coded in Borland C++, and runs on a Pentium II 400 MHz machine using the Windows 98 operating system. Eight large size problems originating from the public transport industry are used as test problems [12].

Table IV. Details of the test problems and related best known solutions

Data	Rows	Columns	Density (%)	Best known solutions		
				Cover size	Cost	Elapsed time (seconds)
T1*	340	29380	1.90	62	509.25	955
T2	437	25099	1.26	116	1003.55	69
T3	546	43743	1.80	64	562.22	>40000
T4*	613	22568	1.58	75	851.09	452
T5	707	144339	0.51	242	2247.52	>80000
T6	1164	29465	0.36	276	2083.15	>80000
T7	1495	28639	0.30	349	2661.12	>80000
T8	1873	50000	0.27	395	3137.20	>80000

Details of these problems, including the number of rows, number of columns, and density (percentage of ones in the a_{ij} matrix), are given in Table IV.

The best results are mostly obtained by a commercial ILP solver called TRACS II [8]. In those cases (marked by asterisk) that TRACS II has difficulty in finding solutions, results achieved by a hybrid GA incorporating strong domain knowledge [12] are cited.

To give fair comparisons, each test problem was run using the same pseudo random number seed at the beginning of the program. The best parameters are those producing the smallest solution cost among the 50 trials of orthogonal experimental design.

To speed up the process of orthogonal experimental design, the iteration number of the self-evolving algorithm is set to be a moderate constant of 200 for each experiment of all problems. The parameter sets that produce the smallest cost are used to carry out further searches and obtain the final solutions. If no improvement has been achieved for 1000 iterations, the program will be terminated.

5.1. EXPERIMENT 1

The indices of Maximum, Minimum, Mean, and Standard Deviation in statistics are applied to study the distribution of the experimental results in terms of solution cost. The Mean is a measure to evaluate the average performance of the proposed algorithm, while the Standard Deviation is a summary measure of the differences of each result from the mean. Let t_i be the variable of solution cost, then

$$\text{Minimum} = \min\{t_1, \ldots, t_{50}\}, \tag{23}$$

$$\text{Maximum} = \max\{t_1, \ldots, t_{50}\}, \tag{24}$$

Table V. Summary results of 50 trials using $L_{50}(2^1 \times 5^6)$

Data	Minimum		Maximum		Mean		Standard Deviation
	Cost	RPD (%)	Cost	RPD (%)	Cost	RPD (%)	Cost
T1	529.28	3.93	571.52	12.23	549.13	7.83	10.97
T2	1004.55	0.10	1074.17	7.04	1026.12	2.25	15.23
T3	586.68	4.35	663.72	18.05	628.42	11.77	17.34
T4	850.60	−0.06	924.45	8.62	889.45	4.51	18.26
T5	2269.07	0.96	2490.70	10.82	2329.33	3.64	56.83
T6	2158.53	3.62	2322.77	11.50	2222.58	6.69	37.46
T7	2764.55	3.89	2989.70	12.35	2855.26	7.30	46.13
T8	3321.70	5.88	3636.07	15.90	3514.04	12.01	64.76
Avg.		**2.83%**		**12.06%**		**7.00%**	**33.37**

Table VI. Summary results of 50 trials using randomised parameter sets

Data	Minimum		Maximum		Mean		Standard Deviation
	Cost	RPD (%)	Cost	RPD (%)	Cost	RPD (%)	Cost
T1	535.73	5.20	570.72	12.07	547.45	7.50	10.19
T2	1017.68	1.41	1070.13	6.63	1024.04	2.04	12.72
T3	594.35	5.71	660.48	17.48	627.38	11.59	16.02
T4	869.33	2.14	912.33	7.20	883.68	3.83	16.96
T5	2291.73	1.97	2461.62	9.53	2325.59	3.47	49.34
T6	2160.42	3.71	2292.80	10.06	2219.40	6.54	30.42
T7	2798.55	5.16	2952.35	10.94	2854.21	7.26	42.15
T8	3460.3	9.98	3606.45	14.95	3502.40	11.64	63.60
Avg.		**4.41%**		**11.10%**		**6.73%**	**30.18**

$$\text{Mean } \bar{t} = \sum_{i=1}^{50} t_i/50, \tag{25}$$

$$\text{Standard Deviation } = \sqrt{\sum_{i=1}^{50} (t_i - \bar{t})^2/49}. \tag{26}$$

Table V shows the summary results of 50 trials by parameters associated with the orthogonal array $L_{50}(2^1 \times 5^6)$, and Table VI shows the summary results of 50 trials by parameter sets produced randomly. According to the Relative Percentage

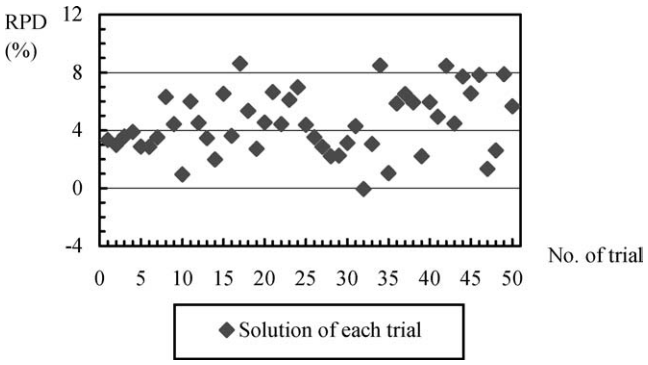

Figure 1. RPD of solution cost versus trial number.

Table VII. Comparative results

Data	Final solution				
	Cover size	RPD (%)	Cost (hours)	RPD (%)	Elapsed time (seconds)
T1	65	4.84	507.53	−0.34	520
T2	118	1.72	994.90	−0.86	161
T3	66	3.13	565.38	0.56	730
T4	75	0.00	819.68	−3.69	168
T5	243	0.41	2246.32	−0.05	1398
T6	273	−1.09	2082.77	0.68	286
T7	345	−1.15	2674.18	0.49	316
T8	393	−0.51	3243.11	3.38	2482
Ave.		**0.92%**		**0.02%**	

Deviation (RPD) results in Table V and Table VI, the average RPD of Minimum, Maximum, and Standard Deviation of $L_{50}(2^1 \times 5^6)$ are 2.83%, 12.06%, and 33.37 respectively, while those produced with random parameters are 4.41%, 11.10%, and 30.18 respectively.

Compared with results achieved with randomised parameters, on average results using orthogonal arrays have better Minimum, worse Maximum, and larger Standard Deviation. This demonstrates that the results achieved by the orthogonal array are more evenly scattered throughout the solution space, some of which are thus inevitably elite. Therefore, it is not necessary to study the full parameter solution space for a near-optimal solution. Figure 1 gives a full demonstration of the variation of solutions for the 50 trial-runs using $L_{50}(2^1 \times 5^6)$ for the T4 instance.

The final results of the test problems are compiled in Table VII. Computational results show that the solutions derived by the proposed approach are very close or even better than to those of the previous best known solutions. The negative

percentage deviation indicates the percentage improvement over the previous best known solution. In terms of solution cost, our results are only 0.02% larger on average: in 4 out of the 8 test problems, our results are better. In term of the executing time, it is obvious that in general our results are obtained at much faster speed, particularly for larger instances.

5.2. EXPERIMENT 2

In addition to finding the best solutions, another task for the first orthogonal experimental design is to explore whether there exists a generally good pattern of parameter settings. According to the experiments, using the orthogonal array $L_{50}(2^1 \times 5^6)$, we find that out of 50 different parameter sets, one set (0.3, 0.9, 0.1, 0.3, 0.5, 0.28, 0.06) gives the best results for all data instances. Naturally, one might be interested to know whether those results in Table VII could be improved further by simply implementing another orthogonal experimental design, which uses the same orthogonal array, but is carried out in narrower ranges of values for the factors.

Table VIII shows the definition of the control factors and their levels in the narrower ranges, which are centred respectively on the parameter configuration of (0.3, 0.9, 0.1, 0.3, 0.5, 0.28, 0.06) found above. These seven factors are assigned to the same $L_{50}(2^1 \times 5^6)$ orthogonal array as shown in Table III.

The final results of the test problems are compiled in Table IX. It shows that compared with results by the parameters defined in the larger ranges, the solutions derived by the refined parameter settings have been slightly improved. On average, the solution cost of the proposed approach is 0.33% lower. At this stage, there is no 'general' parameter set that works for all instances.

With respect to the variation of solution costs for individual cases, Figure 2 depicts the improvement of the cost versus the number of iterations for instance T1. Although the actual values may differ among various cases, the characteristic shapes of the curves are similar.

Table VIII. Control factors and their levels in the narrower ranges

Control factors	Levels				
	1	2	3	4	5
1. Weight w_1	0.20	0.25	0.30	0.35	0.40
2. Weight w_2	0.80	0.85	0.90	0.95	1.00
3. Weight w_3	0.00	0.05	0.10	0.15	0.20
4. Weight w_4	0.20	0.25	0.30	0.35	0.40
5. Weight w_5	0.40	0.45	0.50	0.55	0.60
6. *Selection* valve k_s	0.270	0.275	0.280	0.285	0.290
7. *Mutation* rate p_m	0.055	0.060	–	–	–

Table IX. Comparative results by the refined parameter settings

Data	Final solution				
	Cover size	RPD (%)	Cost (hours)	RPD (%)	Elapsed time (seconds)
T1	64	3.23	504.13	−1.01	384
T2	118	1.72	993.35	−1.01	161
T3	66	3.13	565.38	0.56	730
T4	74	−1.33	814.53	−4.30	668
T5	243	0.41	2246.32	−0.05	1398
T6	275	−0.36	2073.36	−0.47	468
T7	348	−0.29	2670.57	0.36	420
T8	391	−1.01	3239.23	3.25	962
Ave.		**0.69%**		**−0.33%**	

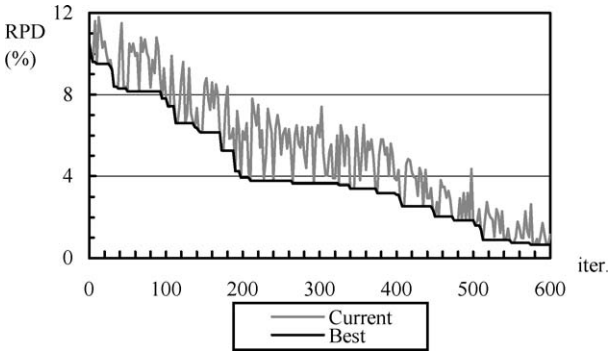

Figure 2. RPD of solution cost (in current and best solution respectively) versus iteration number.

6. Conclusions and Future Work

This paper presents a meta-heuristic to solve the SCP. First, a function is designed to evaluate the structure of each column using fuzzy factors. This function is embedded into the proposed self-evolving algorithm, which mimics generations of evolution on a single solution. In each generation an unfit portion of the working solution is removed. Any infeasible solution is repaired by a greedy building process.

There are seven investigated control factors in the proposed approach. Using the "change-one-factor-at-a-time" method of experimentation, a prohibitively large number of 31,250 experiments needs to be carried out. This paper applies orthogonal experimental design to reduce the number of experiments to 50.

Taken as a whole, our approach has a number of advantages. First, it is simple to carry out because it uses greedy algorithms and local heuristics. Secondly, due to its characteristic of maintaining only a single solution at each generation and discarding inferior columns from this solution, this approach converges fast compared with other meta-heuristics. Thirdly, combined with iterative improvement and constructive perturbation, the approach has the ability to achieve superior solutions by effective exploration of the solution space. Computational results have demonstrated its potential.

However, it needs to be mentioned that at this stage we only test our approach based on our data instances originating from real-world driver scheduling problems, which is a specialised set covering problem. In terms of ILP, the columns correspond to shifts and the rows correspond to pieces of work that are used to compose individual shifts. Therefore, in this kind of set covering problems, each shift must satisfy conditions in the Labour Agreement between management and unions, not just any possible combination of pieces of work. Our next step in the future work is to test the performance of the proposed approach on randomly generated problems obtained from the OR-Library [2, 3], and compare our approach with others based on the same test problems.

Another future research direction is to study the interaction of individual parameters and to improve the solution quality further. According to the experimental results there is no significant improvement in general even if the refined parameter settings have been used. The reason might be that it is only the relative levels of two or three parameters that are important, or that some weights are much more sensitive than others. An advanced ANOVA process therefore needs to be carried out to analyse the results of the orthogonal experimental design, and to determine how much variation each factor has contributed.

Acknowledgements

We would like to express our gratitude to Dr. Uwe Aickelin for his helpful comments and suggestions to improve this paper.

References

1. Aickelin, U.: An indirect genetic algorithm for set covering problems, *J. Oper. Res. Soc.* **53** (2002), 1118–1126.
2. Beasley, J. E.: A Lagrangian heuristic for set covering problems, *Naval Res. Logist.* **37** (1990), 151–164.
3. Beasley, J. E.: OR-Library: Distribution test problems by electronic mail, *J. Oper. Res. Soc.* **41** (1990), 1069–1072.
4. Beasley, J. E. and Chu, P. C.: A genetic algorithm for the set covering problem, *European J. Oper. Res.* **94** (1996), 392–404.
5. Caprara, A., Fischetti, M. and Toth, P.: A heuristic method for the set covering problem, *Oper. Res.* **47** (1999), 730–743.

6. Dénes, J. and Keedwell, A. D.: *Latin Square and Their Applications*, English Universities Press Limited, Budapest, 1974.
7. Garey, M. R. and Johnson, D. S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.
8. Fores, S., Proll, L. and Wren, A.: TRACS II: A hybrid IP/heuristic driver scheduling system for public transport, *J. Oper. Res. Soc.* **53** (2002), 1093–1100.
9. Haddadi, S.: Simple Lagrangian heuristic for the set covering problem, *European J. Oper. Res.* **97** (1997), 200–204.
10. Karmarkar, N.: A new polynomial-time algorithm for linear programming, *Combinatorica* **4** (1984), 373–395.
11. Kling, R. M. and Banerjee, P.: ESP: A new standard cell placement package using simulated evolution, in *Proceedings of the 24th ACM/IEEE Design Automation Conference*, 1987, pp. 60–66.
12. Kwan, R. S. K, Kwan, A. S. K. and Wren, A.: Evolutionary driver scheduling with relief chains, *Evolutionary Computation* **9** (2001), 445–460.
13. Li, J. and Kwan, R. S. K.: A fuzzy simulated evolution algorithm for the driver scheduling problem, in *Proceedings of the 2001 IEEE Congress on Evolutionary Computation*, IEEE Press, 2001, pp. 1115–1122.
14. Li, J. and Kwan, R. S. K.: A fuzzy theory based evolutionary approach for driver scheduling, in L. Spector et al. (eds), *Proceedings of the Genetic and Evolutionary Computation Conference*, Morgan Kaufman, 2001, pp. 1152–1158.
15. Li, J. and Kwan, R. S. K.: A fuzzy genetic algorithm for driver scheduling, *European J. Oper. Res.* **147** (2003), 334–344.
16. Ohlsson, M., Peterson, C. and Söberderg, B.: An efficient mean field approach to the set covering problem, *European J. Oper. Res.* **133** (2001), 583–599.
17. Sen, S.: Minimal cost set covering using probabilistic methods, in *Proceedings of the 1993 ACM Symposium on Applied Computing: States of the Art and Practice*, 1993, pp. 157–164.
18. Slavík, P.: A tight analysis of the greedy algorithm for set cover, in *ACM Symposium on Theory of Computing*, 1996, pp. 435–441.
19. Solar, M., Parada, V. and Urrutia, R.: A parallel genetic algorithm to solve the set-covering problem, *Comput. Oper. Res.* **29** (2002), 1221–1235.
20. Srinivasan, A.: Improved approximations of packing and covering problems, in *ACM Symposium on Theory of Computing*, 1995, pp. 268–276.
21. Taguchi, G.: *System of Experimental Design*, Vols. 1 and 2, UNIPUB/Kraus International Publications, New York, 1987.
22. The triplex design group of Chinese Association of Statistics: *Orthogonal Method and Triplex Design*, Science Press, Beijing, 1987.
23. Zadeh, L. A.: Fuzzy sets, *Inform. Control* **8** (1965), 338–353.