View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Elsevier - Publisher Connector

Scientia Iranica E (2012) 19 (3), 895–901

Sharif University of Technology

Scientia Iranica

Transactions E: Industrial Engineering

www.sciencedirect.com

Research note

Heuristic algorithm for solving the integer programming of the lottery problem

A. Mohammadi [a,*], **I. Nakhaei Kamal Abadi** [b]

[a] *Zarghan Branch, Islamic Azad University, Zarghan, Iran*

[b] *Department of Industrial Engineering, Tarbiat Modares University, Tehran, Iran*

Received 21 June 2010; revised 2 January 2012; accepted 27 April 2012

KEYWORDS

Integer programming;
Set covering problem;
Lottery problem;
Lotto–Meta heuristic;
Neighborhood search.

Abstract In this paper, we propose a heuristic algorithm, named the Lotto–Meta heuristic, to solve small instances of the lottery problem, using its set covering formulation. The algorithm uses a randomized method, allotting a priority to each column to be included in the solution. A neighborhood search strategy is fused with the algorithm to enhance the search and to balance the exploration and exploitation procedures. Computational results show that our method outperforms the best known solutions for a number of hard instances.

© 2012 Sharif University of Technology. Production and hosting by Elsevier B.V.

Open access under CC BY-NC-ND license.

1. Introduction

In a lottery problem, p numbers are randomly drawn from a n-set. These p numbers are lottery winning numbers of which we are not aware. We write k numbers out of the n numbers on each ticket. We wish to know how many tickets have to be bought, and what numbers should be written on each ticket, so that we will have at least one ticket which has, at least, t numbers in common with the p numbers of the lottery winning numbers. The objective is minimizing the number of tickets that have to be purchased. We call this problem a (n, k, p, t) lottery problem, with $n \geq k, p$ and $k, p \geq t$. Since the first research into the lottery problem proposed by Sterboul [1], this problem has been an interesting subject, such that many companies and researchers have dealt with it. An overview of the lottery problem can be found in [2]. They list the known lower and upper bounds of the problem. There are many theorems and results regarding lower and upper bounds concerning the lottery problem in [3–5]. A number of algorithms are proposed for solving the lottery problem by Li and van Rees [2,5]. In spite of the simplicity of the lottery model, its solution is very hard. Therefore, these algorithms can only deal with problems with small instances of n, p and k [2,5]. Jans and Degraeve formulated the lottery problem as a set covering problem for a special case ($p = k$) and solved it by CPLEX [6]. They did not consider the case $p \neq k$ [6]; this case is discussed in this paper. To the best of our knowledge, only small instances of lottery problems can be solved. Currently, nobody in the literature has solved realistically sized problems [2,6].

The set covering problem is an applied classic problem in combinatorial optimization [7–17]. The current work encourages people to solve entertaining puzzles through integer programming. To the best of our knowledge, the lottery problem with $p \neq k$ has not been discussed in set covering literature. Furthermore, to the best of our knowledge, no heuristic approach is proposed for the lottery problem as the set covering framework. Based on the limited available research into the lottery problem, we are motivated to develop the Lotto–Meta heuristic algorithm to solve the lottery problem as a set covering formulation.

Each implementation of the Lotto–Meta heuristic is made up of two phases; constructive and improvement phases. During the constructive phase, the algorithm uses a randomized greedy heuristic to obtain an initial solution to the problem. Then, the improvement phase permits exploration of the

* Corresponding author.
E-mail addresses: aboozar.mohammadi@yahoo.com (A. Mohammadi), nakhai_isa@yahoo.com (I. Nakhaei Kamal Abadi).

Peer review under responsibility of Sharif University of Technology.

Production and hosting by Elsevier

1026-3098 © 2012 Sharif University of Technology. Production and hosting by Elsevier B.V. Open access under CC BY-NC-ND license.
doi:10.1016/j.scient.2012.04.015

generated solution neighborhood in an attempt to find a better solution. After the improvement phase, the best solution found during this phase is compared to the best-known solution, and substitutes it if the objective value is better than the one previously known. These steps are continued once a stopping criterion is met; the best solution obtained during this procedure is returned.

This paper is organized as follows: In Section 2, the lottery problem, as a set covering problem, will be described. The proposed algorithm is presented in Section 3. In Section 4, experimental results and comparisons are reported, and conclusions are presented in Section 5.

2. Lottery problem as set covering problem

For a $(n,\ k,\ p,\ t)$ lottery problem, a specific ticket is characterized by the choice of the k numbers that are filled out of n *numbers*. A specific draw is characterized by the set of p numbers out of n *numbers* that are lottery winning numbers. Suppose S as the set of all possible tickets, and D as the set of all possible draws. Then, a binary variable for each possible ticket is defined as follows:

$$\forall j \in S : x_j = \begin{cases} 1 & \text{if ticket } j \text{ is filled out} \\ 0 & \text{otherwise.} \end{cases}$$

Define S_i as the set of all tickets which have at least t numbers in common with draw i, and define D_j as the set of all draws which have at least t numbers in common with ticket j. Finally, we define the parameters of the coefficient matrix as follows:

$$\forall i \in D,\ \forall j \in S : a_{ij} = \begin{cases} 1 & \text{if ticket } j \in S_i \\ 0 & \text{otherwise.} \end{cases}$$

The objective function is to minimize the number of tickets that have to be filled out. There is only one set of constraints imposed, so that, for each possible draw, there must be at least one ticket filled out which has at least t numbers in common with that draw. The variables for the tickets must be binary. The resulted formulation is a set covering problem:

$$\text{Min} \sum_{j \in S} x_j, \tag{1}$$

subject to

$$\sum_{j \in S} a_{ij} x_j \geq 1 \quad \forall i \in D, \tag{2}$$

$$x_j \in \{0, 1\} \quad \forall j \in S. \tag{3}$$

The number of tickets is equal to the number of combinations of k elements out of n:

$$|S| = \binom{n}{k}.$$

The number of constraints of the problem is equal to:

$$|D| = \binom{n}{p}.$$

Suppose that a specific outcome is drawn with p numbers. We wish to know how many tickets have at least t numbers in common with this particular draw. This will give us the number of variables with a non-zero coefficient per constraint. First, we find how many tickets have exactly t numbers in common. This is the number of combinations of t out of the p selected numbers, multiplied by the number of combinations of the $k-t$ numbers on a ticket out of the pool of the $n-p$ numbers that were not part of the outcome: $\binom{p}{t}\binom{n-p}{k-t}$. The number of tickets with exactly $t+1$ numbers in common is $\binom{p}{t+1}\binom{n-p}{k-t-1}$. By the same reasoning, it can be found that the number of variables per constraint is:

$$\forall i \in D \quad |S_i| = \sum_{m=t}^{\text{Min}\{k,p\}} \binom{p}{m}\binom{n-p}{k-m}.$$

Similarly, the number of draws out of D which have at least t numbers in common with a specific ticket is equal to the number of non zeros in each column of the coefficient matrix, which is equal to:

$$\forall j \in S \quad |D_j| = \sum_{m=t}^{\text{Min}\{k,p\}} \binom{k}{m}\binom{n-k}{p-m}.$$

2.1. Characteristics of the SCP formulation of the lottery problem

- The obtained set covering problem is unicost.
- The number of non zeroes is the same for each column. Similarly, the number of non zeroes is the same in each row. These properties increase the hardness of the problem.
- There is equivalence between a draw and a ticket in the problem, when $p = k$.
- When $p = k$, the set of all possible tickets and the set of all possible draws are equal, and the coefficient matrix is symmetric.

2.2. The LP solution of the problem

Theorem. *If $p = k$, then in the optimal LP solution, each variable, x_j, has the value $\frac{1}{|S_i|} = \frac{1}{\sum_{m=t}^{k}\binom{k}{m}\binom{n-k}{k-m}}$ and the optimal LP objective value is $\frac{|S|}{|S_i|} = \frac{\binom{n}{k}}{\sum_{m=t}^{k}\binom{k}{m}\binom{n-k}{k-m}}$.*

Nurmela and Östergård [18] acquired this bound founded on covering designs, and Li and van Rees [4] generalized it for the lottery problem. Burger et al. [19] verified this bound also using the graph theory. Jans and Degraeve [6] proved the bound using a Linear Programming set covering approach. However, the knowledge of the structure of an optimal LP solution is not very helpful in finding the optimal IP solution, and it can be used as a lower bound.

2.3. Symmetry

Due to symmetry, it can be concluded that there exist many alternative optimal solutions. Sherali and Smith [20] show that the existence of alternative optimal solutions has a bad effect on solution times for solving IP problems. The branch-and-bound method cannot cut a node that leads to any of the alternative optimal solutions and must evaluate them all. In this case, to exclude some alternative solutions, one specific variable can be fixed as one. This fixing partly breaks the symmetry and decreases solution time significantly. Jans and Degraeve [6] show that fixing one variable still allows an optimal solution to be found, because an optimal solution can be obtained with a specific variable fixed to one. Problems, (n, k, p, t) and (n, p, k, t), are symmetric and their optimal

solutions are not necessarily the same. For example, the optimal solution of (9, 6, 5, 4) is 3, and the optimal solution of (9, 5, 6, 4) is 4.

3. Heuristic approach

Because of the rare characteristics of the problem described in Section 2, exact algorithms are not able to solve large instances of the problem. As will be seen in computational results, a powerful optimizer, such as CPLEX, is not able to solve some problems. A brief review of the set covering problem literature shows that the proposed procedures do not produce good solutions for unicost problems, and heuristics are not excluded. The reason is that columns can no longer be compared with each other based on column cost and, thus, the problem is harder to solve.

In this paper, a two-phase heuristic is proposed for the lottery problem, which consists of a neighborhood search, randomness and a priority of selection for each column. The following notations are adopted by the authors:

$I = \{i | i \in D,$ row i is uncovered$\}$;

$X = \{j | j \in S,\ x_j = 1\}$;

d_j: The number of currently uncovered rows that could be covered by column j; $d_j = \text{card}\,(I \cap D_j)$;

$J = S - X$;

v_i: The number of selected columns that can cover row i; $v_i = \text{card}\,(X \cap S_i)$

3.1. Constructive phase: creating initial solution

Initially, d_j is the same for all columns, and $I = D$. Therefore, in the beginning, no row is covered. A column, w, is randomly chosen and is placed in X. All rows covered by w are removed from I, and d_j is changed for each column. Let $f = \max\{d_j | j \in J\}$. A column is randomly selected from the set $\{j | d_j = f,\ j \in J\}$; it is placed in X and the set of all rows that are covered by this column are removed from I. This is the greedy heuristic for set covering problems [13]. Again, d_j is updated for all columns in J. By performing these steps, all rows are covered and redundant columns are deleted at the end. The pseudo codes for this phase are summarized in Figures 1 and 2.

3.2. Improvement phase: improving the initial solution

The number of columns which is dependent on the number of selected columns, (approximately $\left\lfloor \frac{\text{card}(X)}{2} \right\rfloor + 1$), are randomly removed from the given feasible solution and is called X', which is now an infeasible solution because there are some uncovered rows. Let $D^* = D - \cup_{j \in X'} D_j$ be the set of all uncovered rows after deletion and let $S^* = \cup_{i \in D^*} S_i$ be the columns that could cover these rows. Consider a reduced SCP of covering the set of rows, D^*, using the set of columns, S^*, as follows:

$$\text{Min} \sum_{j \in S^*} x_j, \tag{4}$$

s. t.

$$\sum_{j \in S^*} a_{ij} x_j \geq 1 \quad \forall i \in D^*, \tag{5}$$

$$x_j \in \{0, 1\} \quad \forall j \in S^*. \tag{6}$$

```
Constructive (D, S):
    Let the solution set be empty: X = ∅
    for each j ∈ S : d_j = |D_j|
    Let I = D, J = S
    w= RND(S)
    add element w to the solution; X = X ∪ {w}
    I = I − D_w, J = J − {w}
    For each i ∈ D_w: For each j ∈ S_i: d_j = d_j − 1
    while I ≠ ∅
        f = max{d_j | j ∈ J}
        w = RND({j | d_j = f, j ∈ J})
        add element w to the solution: X = X ∪ {w}
        I = I − D_w, J = J − {w}
        For each i ∈ D_w: For each j ∈ S_i: d_j = d_j − 1
    End while
    Remove redundant columns from X
    Return X
```

Figure 1: Constructive procedure.

```
Redundant(X):
  For each i ∈ D: v_i = 0
  For each j ∈ X: For each i ∈ D_j: Update the number of
                  selected columns for each row i: v_i = v_i + 1
  For each j ∈ X
        If (min_{i∈D_j}{v_i}) > 1
        Remove j from X
        For each i ∈ D_j: v_i = v_i − 1
        End if
  End for
```

Figure 2: Pseudo-code for removing the redundant columns.

```
Improvement(D, S, X, Remove-col):
    z= objective function value of X
    X'= Randomly remove Remove-col columns from X
    Formulate a reduced-size SCP:
                        D* = D − ∪_{j∈X'} D_j
                        S* = ∪_{(i ∈ D*)} S_i
    X*= construction(D*, S*)
    Construct the neighboring solution: X' ∪ X*
    Remove redundant columns from X' ∪ X*
    If (objective function value of X' ∪ X*) < z then
        X = X' ∪ X*
        z= objective function value of X
    End if
    Return X
```

Figure 3: Improvement procedure.

By applying the constructive phase for the reduced SCP problem, a solution, X^*, is created. Then, combining the solutions of the reduced SCP (X^*) and the partial solution (X'), a new solution is generated called a neighboring solution. The redundant columns from $X^* \cup X'$ are removed. If this neighboring solution ($X^* \cup X'$) improves, solution X will be replaced by it. The pseudo code for the improvement phase is presented in Figure 3.

Table 1: Computational results for the lottery problems with $k = 3$ and $t = 2$.

Lottery	$\|S\|$	$\|S_i\|$	LP	Jans and Degraeve [6]			Lotto–Meta heuristic		Li and van Rees [2]	
				IP	Time I (s)	Time II (s)	IP	Time (s)	LB	UP
(5, 3, 3, 2)	10	7	1.43	2	0.00	0.00	2	0.00	2	2
(6, 3, 3, 2)	20	10	2.00	2	0.00	0.00	2	0.00	2	2
(7, 3, 3, 2)	35	13	2.69	4	0.01	0.00	4	0.00	4	4
(8, 3, 3, 2)	56	16	3.5	5	0.30	0.01	5	0.00	5	5
(9, 3, 3, 2)	84	19	4.42	7	4.56	0.47	7	0.04	7	7
(10, 3, 3, 2)	120	22	5.45	8	11.81	1.56	8	0.05	8	8
(11, 3, 3, 2)	165	25	6.6	10	635	33.70	10	0.20	10	10
(12, 3, 3, 2)	220	28	7.86	11	626	45.69	11	0.21	11	11
(13, 3, 3, 2)	286	31	9.23	13	18 523	1004	13	0.61	13	13
(14, 3, 3, 2)	364	34	10.71	14	4170	623	14	0.16	14	14
(15, 3, 3, 2)	455	37	12.30	n.a.	43 200[a]	43 200[a]	18	0.41	18	18
(16, 3, 3, 2)	560	40	14.00	n.a.	43 200[b]	43 200[c]	19	0.56	19	19

[a] Discontinued before optimality; upper bound: 18, lower bound: 15.
[b] Discontinued before optimality; upper bound: 19, lower bound: 16.
[c] Discontinued before optimality; upper bound: 19, lower bound: 17.

Table 2: Computational results for the lottery problems with $k = 4$ and $t = 2$.

Lottery	$\|S\|$	$\|S_i\|$	LP	Jans and Degraeve [6]			Lotto–Meta heuristic		Li and van Ree [2]	
				IP	Time I (s)	Time II (s)	IP	Time (s)	LB	UP
(5, 4, 4, 2)	5	5	1.00	1	0.00	0.00	1	0.00	1	1
(6, 4, 4, 2)	15	15	1.00	1	0.00	0.02	1	0.00	1	1
(7, 4, 4, 2)	35	31	1.13	2	0.02	0.00	2	0.00	2	2
(8, 4, 4, 2)	70	53	1.32	2	0.03	0.02	2	0.00	2	2
(9, 4, 4, 2)	126	81	1.56	2	0.11	0.02	2	0.00	2	2
(10, 4, 4, 2)	210	115	1.83	3	0.55	0.05	3	0.00	3	3
(11, 4, 4, 2)	330	155	2.13	3	7.00	0.09	3	0.01	3	3
(12, 4, 4, 2)	495	201	2.46	3	15.22	0.14	3	0.03	3	3
(13, 4, 4, 2)	715	253	2.83	5	43 200[a]	170	5	0.06	5	5
(14, 4, 4, 2)	1001	311	3.22	5	18 575	14.00	5	0.11	5	5
(15, 4, 4, 2)	1365	375	3.64	n.a.	43 200[b]	43 200[c]	7	0.17	7	7
(16, 4, 4, 2)	1820	445	4.09	n.a.	43 200[d]	43 200[e]	7	3.60	7	7

[a] Discontinued before optimality; upper bound: 5, lower bound: 4.
[b] Discontinued before optimality; upper bound: 7, lower bound: 4.
[c] Discontinued before optimality; upper bound: 7, lower bound: 5.
[d] Discontinued before optimality; upper bound: 8, lower bound: 5.
[e] Discontinued before optimality; upper bound: 7, lower bound: 6.

The following parameters are used:

- *Max-constructive*: the number of times that the constructive phase is executed.
- *Max-improvement*: the number of times that the improvement phase is executed.
- *Remove-col*: the number of columns that are removed in the improvement phase.
- *Max-total*: the number of times that total steps are executed (both constructive and improvement phases).

The constructive phase is performed *Max-constructive* times, and a solution is generated each time. The best solution is considered as input for the improvement phase. The improvement phase is applied *Max-improvement* times and, for each time, the best improved solution is considered as the input to the next improvement iteration. This process is performed *Max-total* times, and the best solution found during these *Max-total* times is the solution to the lottery problem.

To choose the parameters that are used in the Lotto–Meta heuristic, a trial and error procedure was used. In this research, the *Max-constructive* parameter is chosen from {1, 3, 10}, *Max-improvement* is chosen from {0, 1, 3, 10, 30, 80, 100, 200, 1000} and, in most replications, *Max-total* is assumed to be one. Thus, very different combinations of parameter setting are tested.

4. Computational result

Jans and Degraeve solved the lottery problem for a special case, ($p = k$), on a Pentium 4, 2.8 GHz computer with 512 MB RAM, under Windows XP [6]. They used CPLEX 9.1.3 to solve the IP problems. They performed two experiments and reported on the CPU times, which are given in seconds and hundreds of seconds. In the first setting (Time I), they solved the problem as given by Relations (1)–(3). In the second setting (Time II), they fix the first variable as one [6]. Their computational results are presented in Tables 1–6 [6]. It is indicated, for some instances, that the IP for Jans and Deraeve is 'n.a.', in case they could not prove optimality within 12 h [6]. In Tables 1–6, we list the lower and upper bounds found by Jans and Degraeve for which optimality could not be proven [6]. Furthermore, the best known lower and upper bounds for the problems obtained by Li and van Rees appear in all tables [2]. The Lotto–Meta heuristic was coded by the authors in MATLAB 2007 and run on a Pentium 4, 2.8 GHz computer, with 512 MB RAM, under Windows XP. In case the time is greater than one hundred seconds, we do not report the hundreds of seconds. A time limit of 12 h was constrained. As seen in Table 7, the bounds obtained by Li and van Rees are outperformed for problems (12, 7, 5, 4), (14, 7, 5, 4), (14, 8, 5, 4), (15, 9, 5, 4), (14, 7, 6, 4), (12, 5, 7, 4), (14, 6, 7, 4).

Table 3: Computational results for the lottery problems with $k = 4$ and $t = 3$.

Lottery	$\|S\|$	$\|S_i\|$	LP	Jans and Degraeve [6]			Lotto–Meta heuristic		Li and van Rees [2]	
				IP	Time I (s)	Time II (s)	IP	Time (s)	LB	UP
(5, 4, 4, 3)	5	5	1.00	1	0.00	0.00	1	0.00	1	1
(6, 4, 4, 3)	15	9	1.67	3	0.02	0.00	3	0.00	3	3
(7, 4, 4, 3)	35	13	2.69	4	0.01	0.00	4	0.00	4	4
(8, 4, 4, 3)	70	17	4.12	6	1.48	0.09	6	0.02	6	6
(9, 4, 4, 3)	126	21	6.00	9	53.67	3.81	9	0.07	9	9
(10, 4, 4, 3)	210	25	8.40	n.a.	43 200[a]	43 200[b]	14	19.01	12	14
(11, 4, 4, 3)	330	29	11.38	n.a.	43 200[c]	43 200[d]	19	24.07	16	19

[a] Discontinued before optimality; upper bound: 14; lower bound: 12.
[b] Discontinued before optimality; upper bound: 14; lower bound: 13.
[c] Discontinued before optimality; upper bound: 19; lower bound: 14.
[d] Discontinued before optimality; upper bound: 19; lower bound: 15.

Table 4: Computational results for the lottery problems with $k = 5$ and $t = 2$.

Lottery	$\|S\|$	$\|S_i\|$	LP	Jans and Degraeve [6]			Lotto–Meta heuristic		Li and van Rees [2]	
				IP	Time I (s)	Time II (s)	IP	Time (s)	LB	UP
(6, 5, 5, 2)	6	6	1.00	1	0.00	0.00	1	0.00	1	1
(7, 5, 5, 2)	21	21	1.00	1	0.02	0.00	1	0.00	1	1
(8, 5, 5, 2)	56	56	1.00	1	0.00	0.00	1	0.00	1	1
(9, 5, 5, 2)	126	121	1.04	2	0.20	0.02	2	0.00	2	2
(10, 5, 5, 2)	252	226	1.12	2	0.97	0.02	2	0.01	2	2
(11, 5, 5, 2)	462	381	1.21	2	7.42	0.09	2	0.04	2	2
(12, 5, 5, 2)	792	596	1.33	2	32.50	0.36	2	0.09	2	2
(13, 5, 5, 2)	1287	881	1.46	3	43 200[a]	1.81	3	0.21	3	3
(14, 5, 5, 2)	2002	1264	1.61	3	43 200[b]	7.38	3	0.45	3	3
(15, 5, 5, 2)	3003	1701	1.77	3	43 200[c]	17.75	3	0.94	3	3

[a] Discontinued before optimality; upper bound: 3, lower bound: 2.
[b] Discontinued before optimality; upper bound: 3, lower bound: 2.
[c] Discontinued before optimality; upper bound: 4, lower bound: 2.

Table 5: Computational results for the lottery problems with $k = 5$ and $t = 3$.

Lottery	$\|S\|$	$\|S_i\|$	LP	Jans and Degraeve [6]			Lotto–Meta heuristic		Li and van Rees [2]	
				IP	Time I (s)	Time II (s)	IP	Time (s)	LB	UP
(6, 5, 5, 3)	6	6	1.00	1	0.00	0.00	1	0.00	1	1
(7, 5, 5, 3)	21	21	1.00	1	0.00	0.00	1	0.00	1	1
(8, 5, 5, 3)	56	46	1.22	2	0.02	0.00	2	0.00	2	2
(9, 5, 5, 3)	126	81	1.56	2	0.09	0.02	2	0.00	2	2
(10, 5, 5, 3)	252	126	2.00	2	0.05	0.01	2	0.01	2	2
(11, 5, 5, 3)	462	181	2.55	5	43 200[a]	222	5	0.03	5	5
(12, 5, 5, 3)	792	246	3.22	6	43 200[b]	1757	6	0.08	6	6
(13, 5, 5, 3)	1287	321	4.01	n.a.	43 200[c]	43 200[d]	8	9.50	8	8
(14, 5, 5, 3)	2002	406	4.93	n.a.	43 200[e]	43 200[f]	10	497.41	8	10

[a] Discontinued before optimality; upper bound: 5, lower bound: 4.
[b] Discontinued before optimality; upper bound: 6, lower bound: 5.
[c] Discontinued before optimality; upper bound: 8, lower bound: 5.
[d] Discontinued before optimality; upper bound: 8, lower bound: 6.
[e] Discontinued before optimality; upper bound: 12, lower bound: 5.
[f] Discontinued before optimality; upper bound: 11, lower bound: 6.

When we increase the value of t for fixed values of n, k and p, we acquire a set covering problem with the same number of variables and constraints, but with fewer variables per constraint. The results show that the problems become more difficult to solve with an increasing value of t. This is revealed from the Jans and Degraeve results [6]. Also, this is revealed from the bounds found by Li and van Rees, as the gap between the lower and upper bounds gets wider [2].

5. Conclusion

The inherent symmetry of the lottery problem leads to many alternative optimal solutions for this problem. For this reason, exact algorithms are unable to solve some lottery problems, but some heuristic approaches are successful. In this paper, the SCP model of the lottery problem is solved by the Lotto–Meta heuristic, incorporating randomness and a neighborhood search, and a priority for each column included in the solution. In the Lotto–Meta heuristic algorithm, constructive heuristics are used once again to generate neighboring solutions in the improvement phase.

This algorithm has proved to be very fast, and capable of producing good solutions. It can solve some problems that CPLEX cannot solve in 12 h, and has outperformed Li and van Rees bounds in some problems [2,6]. Furthermore, another important attractiveness of this method is

Table 6: Computational results for the lottery problems with $k = 5$ and $t = 4$.

Lottery	$\|S\|$	$\|S_i\|$	LP	Jans and Degraeve [6]			Lotto–Meta heuristic		Li and van Rees [2]	
				IP	Time I (s)	Time II (s)	IP	Time (s)	LB	UP
(6, 5, 5, 4)	6	6	1.00	1	0.00	0.00	1	0.00	1	1
(7, 5, 5, 4)	21	11	1.91	3	0.00	0.00	3	0.00	3	3
(8, 5, 5, 4)	56	16	3.50	5	0.33	0.02	5	0.00	5	5
(9, 5, 5, 4)	126	21	6.00	9	54.45	3.88	9	0.05	9	9
(10, 5, 5, 4)	252	26	9.69	14	43 200[a]	5965	14	8.74	10	14
(11, 5, 5, 4)	492	31	14.90	n.a.	43 200[b]	43 200[c]	22	36.22	17	22
(12, 5, 5, 4)	792	36	22.00	n.a.	43 200[d]	43 200[e]	35	478.29	25	35

[a] Discontinued before optimality; upper bound: 14, lower bound: 13.
[b] Discontinued before optimality; upper bound: 24, lower bound: 17.
[c] Discontinued before optimality; upper bound: 23, lower bound: 18.
[d] Discontinued before optimality; upper bound: 35, lower bound: 23.
[e] Discontinued before optimality; upper bound: 36, lower bound: 24.

Table 7: Computational results for the lottery problems.

Lottery	$\|S\|$	$\|D\|$	$\|S_i\|$	$\|D_j\|$	Lotto–Meta heuristic		Li and van Rees [2]	
					IP	Time (s)	LB	UP
(17, 5, 4, 2)	6188	2380	2041	785	5	2.29	5	5
(17, 4, 5, 2)	2380	6188	785	2041	6	1.56	6	6
(14, 7, 6, 2)	3432	3003	3256	2849	2	1.61	2	2
(11, 5, 4, 3)	462	330	91	65	9	3.17	9	9
(14, 5, 4, 3)	1287	1001	153	85	16	129	11	16
(13, 6, 4, 3)	1716	715	372	155	9	5.45	8	9
(13, 7, 4, 3)	1716	715	588	245	6	0.14	6	6
(13, 4, 5, 3)	715	1287	85	153	13	1.44	13	13
(13, 4, 6, 3)	715	1716	155	372	10	3.72	10	10
(13, 4, 7, 3)	715	1716	245	588	6	0.14	6	6
(9, 6, 5, 4)	84	126	34	51	3	0.00	3	3
(12, 7, 5, 4)	792	792	196	196	8	0.55	6	9
(12, 8, 5, 4)	495	792	210	336	3	0.05	3	3
(13, 8, 5, 4)	1287	1287	406	406	6	1.31	6	6
(14, 7, 5, 4)	3432	2002	456	266	16	699.37	8	18
(14, 8, 5, 4)	3003	2002	714	476	7	155.12	6	9
(15, 9, 5, 4)	5005	3003	1470	882	7	43.10	6	8
(9, 5, 6, 4)	126	84	51	34	4	0.00	4	4
(14, 7, 6, 4)	3432	3003	1016	889	7	3.26	6	8
(12, 5, 7, 4)	792	792	196	196	8	0.87	6	9
(14, 6, 7, 4)	3003	3432	889	1016	6	26.96	5	8
(14, 8, 7, 4)	3003	3432	2114	2416	2	1.29	2	2
(15, 5, 9, 4)	3003	5005	882	1470	8	160.23	5	8

its outstanding performance in solving unicost set covering problems.

References

[1] Sterboul, F. "Le Problème du Loto", *Cahiers du centre d'Etudes de Recherche Operationnelle (CERO)*, 20, pp. 443–449 (1978).

[2] Li, P.C. and van Rees, G.H.J. "Lotto design tables", *Journal of Combinatorial Designs*, 10, pp. 335–359 (2002).

[3] Li, P.C. "Some results on lotto designs", Ph.D. Thesis, University of Manitoba (1999).

[4] Li, P.C. and van Rees, G.H.J. "Lower bounds on lotto designs", *Congressus Numerantium*, 141, pp. 5–30 (1999).

[5] Li, P.C. and van Rees, G.H.J. "New constructions of lotto designs", *Utilitas Mathematica*, 58, pp. 45–64 (2000).

[6] Jans, R. and Degraeve, Z. "A note on a symmetrical set covering problem: the lottery problem", *European Journal of Operational Research*, 186, pp. 104–110 (2008).

[7] Aickelin, U. "An indirect genetic algorithm for set covering problems", *Journal of the Operational Research Society*, 53, pp. 1118–1126 (2002).

[8] Bautista, J. and Pereira, J. "A GRASP algorithm to solve the unicost set covering problem", *Computers & Operations Research*, 34, pp. 3162–3173 (2007).

[9] Beasley, J.E. "An algorithm for the set covering problem", *European Journal of Operational Research*, 31, pp. 85–93 (1987).

[10] Beasley, J.E. "A Lagrangian heuristic for set covering problems", *Naval Research Logistics*, 37, pp. 151–164 (1990).

[11] Beasley, J.E. and Chu, P.C. "A genetic algorithm for the set covering problem", *European Journal of Operational Research*, 94, pp. 392–402 (1996).

[12] Caprara, A., Fischetti, M. and Toth, P. "A heuristic method for the set covering problem", *Operations Research*, 47(5), pp. 730–743 (1999).

[13] Caprara, A., Fischetti, M. and Toth, P. "Algorithms for the set covering problem", *Annals of Operations Research*, 98, pp. 353–371 (2000).

[14] Chvatal, V. "A greedy heuristic for the set covering problem", *Mathematics of Operations Research*, 4, pp. 233–235 (1979).

[15] Fisher, M. and Kedia, P. "Optimal solution of set covering partitioning problems using dual heuristic", *Management Science*, 36, pp. 674–688 (1990).

[16] Jacobs, L. and Brusco, M. "Note: a local-search heuristic for large set-covering problems", *Naval Research Logistics*, 42, pp. 1129–1140 (1995).

[17] Lan, G., DePuy, G.W. and Whitehouse, G.E. "An effective and simple heuristic for the set covering problem", *European Journal of Operational Research*, 176, pp. 1387–1403 (2007).

[18] Nurmela, K.J. and Östergård, P.R.J. "Constructing covering designs by simulated annealing. Upper bounds of covering designs by simulated annealing", *Congressus Numerantium*, 96, pp. 93–111 (1993).

[19] Burger, A.P., Grundlingh, W.R. and Vuuren, J.H. "Two combinatorial problems involving lottery schemes: algorithmic determination of solution set", *Utilitas Mathematica*, 70, pp. 33–70 (2006).

[20] Sherali, H.D. and Smith, J.C. "Improving discrete model representations via symmetry considerations", *Operations Research*, 47(10), pp. 1396–1407 (2001).

Aboozar Mohammadi received the B.S. degree in Applied Mathematics in 2006 from Shahrood University of Technology (SUT), and M.S. degree in Applied Mathematics (Operation Research) in 2009 from University Of Kurdistan (UOK), Sanandaj. His research interests include optimization both in discrete and continues space and topics in optimization and combinatorial optimization, scheduling, theory and algorithms, Logistics and inventory management, supply chain management, International manufacturing, heuristics and Meta heuristics, and queuing theory.

Isa Nakhaei Kamal Abadi received the B.S. degree in System Analysis and Computer Appl. and M.Sc. and Ph.D. degrees in Industrial Engineering. He is an Associate Professor of Tarbiat Modares University. His research interests include optimization, scheduling, logistics and inventory management, and supply chain.