International Journal of Industrial Engineering & Production Research

June 2014, Volume 25, Number 2
pp. 95-102

http://IJIEPR.iust.ac.ir/

pISSN: 2008-4889

A Local Branching Approach for the Set Covering Problem

M. Yaghini*, M.R. Sarmadi & M. Momeni

Masoud Yaghini, *Assistant Professor, School of Railway Engineering, Iran University of Science and Technology, Tehran, Iran*
Mohammadreza Sarmadi, *MSc., School of Railway Engineering, Iran University of Science and Technology, Tehran, Iran*
Mohsen Momeni, *MSc., School of Railway Engineering, Iran University of Science and Technology, Tehran, Iran*

KEYWORDS

Heuristics, set covering problem,
Local branching algorithm,
Design of experiments

ABSTRACT

The set covering problem (SCP) is a well-known combinatorial optimization problem. This paper investigates development of a local branching-based solution approach for the SCP. This solution strategy is exact in nature, though it is designed to improve the heuristic behavior of the mixed integer programming solver. The algorithm parameters are tuned by design of experiments approach. The proposed method is tested on the several standard instances. The results show that the algorithm outperforms the best heuristic approaches found in the literature.

1. Introduction

The set covering problem (SCP) is a classical combinatorial optimization problem that is central in a variety of scheduling, routing, and location applications. The SCP is a main model for locomotive scheduling in rail transportation, where a given set of trains has to be covered by a minimum-cost set of locomotives that each train should be covered by at least one locomotive.

Let $A=(a_{ij})$ be a 0-1 $m \times n$ matrix with M = {1, 2, …, m} and N = {1, 2, …, n} denoting, respectively, the sets of rows and columns of A. Let $c=(c_j)$ be n-vector of costs associated with the columns of A. We say that a column $j \in N$ covers a row $i \in M$ if $a_{ij}=1$. The problem is to find a minimum cost column subset $S \subset N$ such that each row $i \in M$ is covered by at least one column $j \in S$. Let $x=(x_j)$ be the column vector of variables $x_j=1$ if $j \in S$, $x_j=0$ otherwise. The classic mathematical formulation for the SCP is as follows:

$$Minimize\ z(x)=\sum_{j \in N} c_j x_j \qquad (1)$$

Subject to

$$\sum_{j \in N} a_{ij} x_j \geq 1 \qquad \forall i \in M \qquad (2)$$

$$x_j \in \{0,1\} \qquad \forall j \in N \qquad (3)$$

Objective function (1) calculates the cost. Constraint (2) ensures that each row is covered by at least one column. Constraint (3) ensures the binary nature of decision variables [1].

The set covering problem is known to be NP-hard [2]. It has been considered in the literature as a basic formulation for many real-world optimization

* **Corresponding author: Masoud Yaghini**
Email: *yaghini@iust.ac.ir*
Paper first received Nov. 25, 2012, and in accepted form May 29, 2013.

problems, therefore it is well-known for its numerous applications. Many algorithms have been developed to solve this problem. The literature covers exact, heuristic and metaheuristic approaches to solve the SCP.

Exact algorithms are mostly based on brand-and-bound and branch-and-cut [3, 4]. In recent years some works are presented in this issue such as Avella et al. [5]. Björklund et al. [6] presented a column generation method that effectively exploits the structure of the formulations. The method can be used to find optimal or near-optimal schedules for networks with arbitrary topology and realistic size. They formulated the two problems using set covering formulations and they derived the column generation method. Hemazroa et al. [7] solved an assignment problem by an algorithm combining the column generation technique and a branch-and-cut scheme. Galiniera and Hertz [8] proposed three exact algorithms for solving the large set covering problem. Two of them determine minimal covers, while the third one produces minimum covers. Heuristic versions of these algorithms are also proposed and analyzed.

Some heuristic-based methods are used in the literature. Fisher and Rinnooy Kan [9] pointed that greedy methods are an important class of one-pass constructive heuristics for the SCP, used to rapidly generate a feasible solution after a single sweep through the problem data. Chvátal [10] proposed a widespread constructive heuristic for the SCP which is called Chvátal method. At each step, it examines the unselected columns and selects the one that reduces the total cost by the greatest amount in proportion to the number of rows covered by the column, until all rows have been covered.

The Chvátal method has been extensively used to produce feasible solutions as a part of more advanced algorithms. Examples of such uses include: the primal-dual approach of Balas and Ho [11], the recursive variant of Avis [12], the approximation algorithms of Baker [13], and the six greedy approaches investigated by Vasko and Wilson [14, 15]. Ablanedo-Rosas and Rego [1] introduced a number of normalization rules and demonstrated the rules superiority to the classical Chvátal rule, especially when solving large scale and real-world instances. To challenge very large-scale SCP instances, arising from crew scheduling in the Italian railway, Caprara et al. [16] designed a Lagrangian based heuristic algorithm, named CFT, which is one of the most effective techniques for the general SCP. Ceria et al. [17] suggested a Lagrangian-based heuristic for solving large-scale set-covering problems arising from crew-scheduling at the Italian Railways.

Umetani and Yagiura [18] compared different relaxation heuristics for the SCP. Yagiura et al. [19] proposed a 3-flip neighborhood local search which has the three characteristics. Naji-Azimi et al. [20] proposed a new heuristic algorithm to solve the SCP problem. The method is based on the electromagnetism metaheuristic approach which, after generating a pool of solutions to create the initial population, applies a fixed number of local search and movement iterations. Caprara et al. [21] compared different exact and heuristic algorithms and provided a complete survey of the existing literature.

The other type of solution method is metaheuristic used for the SCP. The metaheuristics for the SCP includes genetic algorithm [22], simulated annealing algorithm [23], tabu search algorithm [24], and ant colony optimization [25, 26, 27]. Indirect genetic algorithms and parallel genetic algorithms are two variants of the well-known genetic metaheuristic approach, proposed simultaneously by Aickelin [28], Solar et al. [29] for the SCP.

The randomized priority search approach for general and the unicost SCP was proposed by Lan et al. [30] for both the. The unicost set covering problem is to determine the smallest possible subset of columns that also covers sets. If all costs associated with the columns set to 1, the general SCP problem will be converted to the unicost problem. By considering a candidate list, they construct an initial solution with a random selection between the best candidate and a member of the candidate list. A new metaheuristic approach called "randomized gravitational emulation search algorithm" for solving large size set covering problems has been designed by Raja Balachandar and Kannan [31].

In previous researches in the literature, the exact algorithm guarantees to find the optimal solution, but for large-scale problem, limited memory and computing time are two fundamental problems that lead them to become unusable. To cover this problem, most researchers use heuristic and hybrid algorithms to solve the optimization problem. According to the problem characteristics, solving the SCP problem with some algorithms are not efficient enough and the obtained solutions are poor.

In this paper for the local branching algorithm is developed for the SCP. The design of experiments (DOE) approach is used to adjust its parameters. The results are compared with the currently published method in the literature. The experimental results show the efficiency and effectiveness of the proposed algorithm.

The remainder of this paper is organized as follows. Section 2 represents the proposed local branching method. In Section 3, parameter tuning using DOE is described. In Sections 4 the experimental results of the algorithm are discussed. Conclusions are presented in Section 5.

2. The Proposed Local Branching Algorithm for the SCP

The local branching [32] is a heuristic technique that solves mixed-integer programming problems. Though the method is exact in nature, it becomes a

heuristic by redefining some control parameters. It has been designed to provide heuristic solutions of high quality, using an MIP solver. The proposed method is described based on the local branching algorithm for the SCP.

Let us consider a general 0–1 mixed-integer program

$$(p)\ \min\ c^T x$$
$$s.t:\ Ax=b$$
$$x_j \in \{0,1\}\quad \forall j \in \beta \neq \emptyset$$
$$x_j \geq 0\quad \forall j \in \delta$$

where the set of variables is partitioned into (β, δ), being β the set of binary variables. Given a feasible solution $\bar{x}$ of (P) and a positive integer parameter k, the k–OPT neighborhood N($\bar{x}$, k) of $\bar{x}$ is the set of feasible solutions of (P) satisfying the additional local branching constraint (constraint (4)).

$$\Delta(x, \bar{x}) = \sum_{j \in \beta: \bar{x}_j = 1} (1 - x_j) + \sum_{j \in \beta: \bar{x}_j = 0} x_j \leq k \qquad (4)$$

In order to describe the constraint (4), the numerical example is applied; lets us to consider current $\bar{x}$ as $\bar{x} = (1,1,0,0)$ .Then constraint (5), which is the local branching constraint, is constructed as following.

$$\Delta(x, \bar{x}) = (1 - x_1) + (1 - x_2) + x_3 + x_4 \leq k \qquad (5)$$

Given the incumbent solution $\bar{x}$, the solution space can be partitioned by constraint (6).

$$\Delta(x, \bar{x}) \leq k \ \text{(left branch) or } \Delta(x, \bar{x}) \geq k + 1 \ \text{(right branch)} \qquad (6)$$

The idea is that neighborhood N($\bar{x}$, k) of left branch should be sufficiently small to be optimized within a short computing time but still large enough to contain better solution. The value of parameter k should be justified in parameter tuning section. The whole method then alternates strategic phases where the additional local branching constraint are used to define promising solution regions, with tactical phases where these regions are explored through a classical branching scheme on the variables, using an MIP solver to do it.

The methodology is converted into a heuristic by adding several parameters. Two parameters are used to put a time limit to the total solving computation time and also to each left branch node solving computation time, respectively. The algorithm starts with a feasible solution $\bar{x}_1$ of (P).

The left branching constraint $\Delta(x, \bar{x}_1) \leq k$ is added to the model and creating a left branch sub-problem that is solved with an MIP solver. If a better solution $\bar{x}_2$ is found, then it becomes the new incumbent. The process backtracks to the father node, the constraint $\Delta(x, \bar{x}_1) \leq k$ is replaced by $\Delta(x, \bar{x}_1) \geq k + 1$, and a new left branch node is created by adding the cut $\Delta(x, \bar{x}_2) \leq k$ to the model.

If the solution $\bar{x}_1$ is not improved within the node time limit, the size of the neighborhood $N(\bar{x}_1, k)$ (i.e., the right hand side of constraint (4)) is reduced. This can be considered an intensification step. A diversification mechanism acts when the MIP solver reports infeasibility or when it is unable to find a feasible solution.

The diversification consists of enlarging the neighborhood of the reference solution $\bar{x}$, by increasing the right hand side of constraint (4).

```
// The pseudocode for the local branching algorithm
Read data
 // Variables initialization
Initialize k, maxDiv, nodeTimeLimit, totalTimeLimit;
Let bestSoFar = UB = TL = +∞;
Let elapsedTime =nodeNumber =divCounter = nodeObjective= 0;
Let diversify = false;
Let firstFeasible = true;
Let rhs = k;
Create model;
Solve zero node (TL, UB, firstFeasible);
If (nodeStatus != Optimal)  {
      Calculate Δ(x, x̄);
      While (divCounter <= maxDiv & elapsedTime <= totalTimeLimit  {
        Add the local branching constraint Δ(x, x̄) ≤ k ;
        TL = min(TL, totalTimeLimit - elapsedTime);
        Solve model (TL, UB, firstFeasible);
        nodeNumber++;
        elapsedTime = currentTime - startTime;
        TL = nodeTimeLimit;
       Check node status;
      }
      TL = totalTimeLimit - elapsedTime;
      firstFeasible = false;
      Solve model (TL, UB, firstFeasible);
}
Output bestSoFar;
End.
```

Fig. 1. The pseudocode for local branching algorithm

The local branching algorithm pseudocode is shown in Figure 1. In this pseudocode, *k*, *maxDiv*, *nodeTimeLimit*, *totalTimeLimit*, *bestSoFar*, and *nodeObjective* are neighborhood size, the maximum number of diversifications, time limit for each tactical branching exploration, overall time limit, best solution, and node objective value, respectively.

In the first part of this pseudocode, variables are initialized. The method consists of a main while loop which is iterated until either the total time limit or the maximum number of diversifications is exceeded. At each iteration, a MIP problem is solved that receives on input three parameters: the local time limit *TL*, the upper bound *UB* used to interrupt the optimization as soon the best lower bound becomes greater or equal to *UB*, and the binary parameter *firstFeasible* to be set to true for aborting the computation when the first

feasible solution is found. MIP solver returns on output the optimal/best solution along the final optimization. After that node status is checked by using *Check node status* method. *diversify* indicating whether the next required diversification or not.
Four different states may occur after each call to MIP solver:

1. *Optimal*: the current MIP has been solved to proven optimality. In this state, the last local branching constraint is reversed into $\Delta(x, \bar{x}) \geq rhs + 1$, the reference solution $\bar{x}$ of value *UB* is updated, the *rhs* set to value of *k* and the algorithm is iterated.
2. *Infeasible*: the current MIP is proven to have no feasible solution of cost strictly less than *UB*, so the last local branching constraint is reversed into $\Delta(x, \bar{x}) \geq rhs + 1$, The *rhs* and *diversify* set to *rhs+k/2* and *true*, respectively. A diversification is implemented depending on the current value of *diversify*. If *diversify* equals to *true*, *TL* and *UB* set to $+\infty$ and the first feasible solution will be returned.
3. *Feasible*: a solution of cost strictly less than the upper bound *UB* has been found, but the MIP solver was not capable of proving its optimality for the current problem (due to the imposed time limit or to the requirement of aborting the execution after the first feasible solution is found).
In order to cut off the current reference solution $\bar{x}$, the last local branching constraint $\Delta(x, \bar{x}) \leq rhs$ is replaced by the constraint $\Delta(x, \bar{x}) \geq 1$ (unless this constraint has been already introduced at step 4, in which case the last local branching constraint is simply deleted). The reference solution $\bar{x}$ of value *UB* is updated. The *diversify* variable set to *false* and value of *k* put into *rhs* variable.
4. *Unknown*: no feasible solution of cost strictly less than *UB* has been found within the node time limit, but there is no guarantee that such a solution does not exist. In this state if *diversify* equals to *true* the last local branching constraint $\Delta(x, \bar{x}) \leq rhs$ is replaced by $\Delta(x, \bar{x}) \geq 1$ in order to escape from the current solution, and the upper bound *UB* and *TL* set to $+\infty$ and *rhs=rhs+k/2* and the first feasible solution will be returned, else if *diversify* equals to *false* the constraint $\Delta(x, \bar{x}) \leq rhs$ is deleted and *rhs=rhs-k/2* [32].

3. Parameter Tuning using DOE Approach

In this section, selecting problems and parameters tuning of the local branching algorithm are discussed. The parameters of the proposed algorithm are tuned using Design of Experiments (DOE) approach and Design Expert software.

An experiment can be described as series of tests in which purposeful changes are made to the input variables of a system so that we may observe and identify the reasons for changes in the output response. DOE refers to the process of planning the experiment so that appropriate data that can be analyzed by statistical methods will be collected, resulting in valid and objective conclusions.
The three basic principles of DOE are replication, randomization, and blocking. By replication, we mean a repetition of the basic experiment. Two important properties of replication are it allows the experimenter to obtain an estimate of the experimental error and if the sample mean is used to estimate the effect of a factor in the experiment, permits the experimenter to obtain a more precise estimate of this effect. Randomization is that both the allocation of the experimental material and the order in which the individual runs or trials of the experiment are to be performed are randomly determined and makes this assumption valid.
Blocking is a design technique used to improve the precision with which comparisons among the factors of interest are made. Often blocking is used to reduce or eliminate the variability transmitted from nuisance factors [33].
The important parameters in DOE approach are response variable, factor, level, treatment and effect. The response variable is the measured variable of interest. In the analysis of metaheuristics, the typically measures are the objective value quality and CPU time [34].
A factor is an independent variable manipulated in an experiment because it is thought to affect one or more of the response variables. The various values at which the factor is set are known as its levels. In metaheuristic performance analysis, the factors include both the metaheuristic tuning parameters and the most important problem characteristics.
A treatment is a specific combination of factor levels. The particular treatments will depend on the particular experiment design and on the ranges over which factors are varied. An effect is a change in the response variable due to a change in one or more factors. Design of experiments is a tool can be used to determine important parameters and interactions between them. Four stages of DOE consist of screening and diagnosis of important factors, modeling, optimization and assessment. This methodology is called sequential experimentation which is used to set the parameters in the DOE approach and is used in this paper for local branching algorithm [35].
Experiments are conducted on eight problems with different sizes. In the local branching algorithm, solution quality and CPU time are considered as the response variables.
Factors, levels, and the final parameters for solving problems are shown in Table 1. These parameters are fixed to solve the test instances.

Tab. 1. Level factorial design for local branching algorithm

Factor	Level		Final parameter
	Low	High	
nodeTimeLimit	50	150	100
totalTimeLimit	400	1000	800
maxDiv	3	15	5
k	15	70	20

4. Experimental Results

The local branching algorithm is tested on a set of 87 set covering problems available from the OR Library. There are fourteen sets of benchmark instances called sets 4, 5, 6, A, B, C, D, E, NRE, NRF, NRG, NRH, CLR, CYC and RAIL. Each of sets 4 and 5 has 10 instances, each of sets 6, A to E, and NRE to NRH has five instances, set CLR has four instances, set CYC has six instances and RAIL has seven instances. The characteristics of these instances such as name, number of rows, number of columns, density (the percentage of nonzero entries in the SCP matrix), and cost range are given in Table 2.

Tab. 2. Characteristics of the test instances

Instance	No. of Rows	No. of Columns	Density (%)	Cost Range
Set 4	200	1000	2	[1, 100]
Set 5	200	2000	2	[1, 100]
Set 6	200	1000	5	[1, 100]
Set A	300	3000	2	[1, 100]
Set B	300	3000	5	[1, 100]
Set C	400	4000	2	[1, 100]
Set D	400	4000	5	[1, 100]
E.1	560	500	20	[1, 1]
E.2	430	500	20	[1, 1]
E.3	50	500	20	[1, 1]
E.4	50	500	20	[1, 1]
E.5	514	500	20	[1, 1]
Set NRE	500	5000	10	[1, 100]
Set NRF	500	5000	20	[1, 100]
Set NRG	1000	10000	2	[1, 100]
Set NRH	1000	10000	5	[1, 100]
CLR.10	511	210	12	[1, 1]
CLR.11	1023	330	12	[1, 1]
CLR.12	2047	495	12	[1, 1]
CLR.13	4095	715	12	[1, 1]
CYC.06	240	192	2	[1, 1]
CYC.07	672	448	0.89	[1, 1]
CYC.08	1792	1024	0.39	[1, 1]
CYC.09	4608	2304	0.17	[1, 1]
CYC.10	11520	5120	0.07	[1, 1]
CYC.11	28160	11264	0.03	[1, 1]
Rail 507	507	63009	1.3	[1, 2]
Rail 516	516	47311	1.3	[1, 2]
Rail 582	582	55515	1.2	[1, 2]
Rail 2536	2536	1081841	0.4	[1, 2]
Rail 2586	2586	920683	0.34	[1, 2]
Rail 4284	4284	1092610	0.24	[1, 2]
Rail 4872	4872	968672	0.2	[1, 2]

To evaluate the performance of the hybrid algorithm, the proposed algorithm is compared with the best solution found in the literature in Table 3. The surrogate constraint normalization rules [1], 3-flip neighborhood local search [19], and Lagrangian-based heuristic [17] are selected to compare with the proposed local branching algorithm. The Java programming language and CPLEX 11 software as an MIP solver are used to implement the proposed algorithm. The program was run on a personal computer with core 2 CPU at 2.66 GHz, 4 GBs of RAM, and operating under Microsoft Windows Vista. The PROB columns indicate the name of the instances. Column SCNR stands for surrogate constraint normalization rules, 3-Flip NB column refers to the 3-flip neighborhood local search and L-Heuristic column corresponds to the Lagrangian-based heuristic method. Obj. Value in LOCB columns refer to the solution found by our proposed local branching algorithm and the CPU Time column indicates the execution time for finding the solution in seconds. The best solution that achieved by the selected methods is bolded. IMPROVE column displays the improvement percentage of the proposed algorithm relative to the best solution. As a solution quality, for each test problem the percentage of improvement from the best solution is calculated by Equation (7).

$$IMPROVE = \frac{(Obtained\ solution - Best\ solution)}{Best\ solution\ found\ by\ other\ methods} \times 100 \qquad (7)$$

Tab. 3. Comparing results

PROB	SCNR	3-Flip NB	L-Heuristic	Obj. Value	CPU Time	IMPROVE (%)
4.1	461	**429**	N.A.	429	<1	0.00%
4.2	564	**512**	N.A.	512	<1	0.00%
4.3	559	**516**	N.A.	516	<1	0.00%
4.4	541	**494**	N.A.	494	<1	0.00%
4.5	573	**512**	N.A.	512	<1	0.00%
4.6	586	**560**	N.A.	560	<1	0.00%
4.7	461	**430**	N.A.	430	<1	0.00%
4.8	538	**492**	N.A.	492	1	0.00%
4.9	731	**641**	N.A.	641	<1	0.00%
4.1	545	**514**	N.A.	514	<1	0.00%
5.1	288	**253**	N.A.	253	<1	0.00%
5.2	340	**302**	N.A.	302	<1	0.00%
5.3	245	**226**	N.A.	226	<1	0.00%
5.4	264	**242**	N.A.	242	1	0.00%
5.5	232	**211**	N.A.	211	<1	0.00%
5.6	244	**213**	N.A.	213	<1	0.00%
5.7	311	**293**	N.A.	293	<1	0.00%
5.8	313	**288**	N.A.	288	<1	0.00%
5.9	307	**279**	N.A.	279	<1	0.00%
5.1	286	**265**	N.A.	265	<1	0.00%
6.1	159	**138**	N.A.	138	<1	0.00%
6.2	171	**146**	N.A.	146	<1	0.00%
6.3	158	**145**	N.A.	145	<1	0.00%
6.4	148	**131**	N.A.	131	<1	0.00%
6.5	191	**161**	N.A.	161	<1	0.00%
A.1	279	**253**	N.A.	253	2	0.00%
A.2	278	**252**	N.A.	252	1	0.00%
A.3	262	**232**	N.A.	232	1	0.00%
A.4	234	**234**	N.A.	234	<1	0.00%
A.5	236	**236**	N.A.	236	<1	0.00%
B.1	75	**69**	N.A.	69	<1	0.00%
B.2	84	**76**	N.A.	76	2	0.00%
B.3	85	**80**	N.A.	80	<1	0.00%
B.4	89	**79**	N.A.	79	4	0.00%
B.5	79	**72**	N.A.	72	1	0.00%
C.1	253	**227**	N.A.	227	1	0.00%
C.2	250	**219**	N.A.	219	2	0.00%
C.3	271	**243**	N.A.	243	2	0.00%
C.4	255	**219**	N.A.	219	1	0.00%
C.5	231	**215**	N.A.	215	1	0.00%

Tab. 3 (continued)

PROB	SCNR	3-Flip NB	L-Heuristic	LOCB		IMPROVE (%)
				Obj. Value	CPU Time	
D.1	69	**60**	N.A.	60	2	0.00%
D.2	71	**66**	N.A.	66	8	0.00%
D.3	81	**72**	N.A.	72	3	0.00%
D.4	67	**62**	N.A.	62	9	0.00%
D.5	70	**61**	N.A.	61	1	0.00%
E.1	**5**	N.A.	N.A.	5	1	0.00%
E.2	**5**	N.A.	N.A.	5	<1	0.00%
E.3	**5**	N.A.	N.A.	5	15	0.00%
E.4	**5**	N.A.	N.A.	5	<1	0.00%
E.5	**5**	N.A.	N.A.	5	<1	0.00%
NRE.1	30	**29**	N.A.	29	40	0.00%
NRE.2	34	**30**	N.A.	30	214	0.00%
NRE.3	30	**27**	N.A.	27	55	0.00%
NRE.4	33	**28**	N.A.	28	72	0.00%
NRE.5	32	**28**	N.A.	28	34	0.00%
NRF.1	16	**14**	N.A.	14	58	0.00%
NRF.2	16	**15**	N.A.	15	51	0.00%
NRF.3	16	**14**	N.A.	14	10	0.00%
NRF.4	16	**14**	N.A.	14	50	0.00%
NRF.5	15	**13**	N.A.	13	73	0.00%
NRG.1	199	**176**	176	176	33	0.00%
NRG.2	171	**154**	155	154	800	0.00%
NRG.3	185	**166**	167	166	800	0.00%
NRG.4	186	**168**	170	168	800	0.00%
NRG.5	192	**168**	169	168	800	0.00%
NRH.1	72	**63**	64	63	800	0.00%
NRH.2	73	**63**	64	63	800	0.00%
NRH.3	66	**59**	60	59	800	0.00%
NRH.4	67	**58**	59	58	800	0.00%
NRH.5	61	**55**	55	55	800	0.00%
CLR.10	**29**	N.A.	N.A.	25	22	-13.79%
CLR.11	**28**	N.A.	N.A.	23	102	-17.86%
CLR.12	**28**	N.A.	N.A.	26	800	-7.14%
CLR.13	**32**	N.A.	N.A.	26	800	-18.75%
CYC.06	N.A.	N.A.	N.A.	60	100	0.00%
CYC.07	N.A.	N.A.	N.A.	144	690	0.00%
CYC.08	N.A.	N.A.	N.A.	352	650	0.00%
CYC.09	N.A.	N.A.	N.A.	816	800	0.00%
CYC.10	N.A.	N.A.	N.A.	1916	800	0.00%
CYC.11	N.A.	N.A.	N.A.	4268	800	0.00%
RAIL 507	197	**174**	**174**	174	800	0.00%
RAIL 516	194	**182**	**182**	182	800	0.00%
RAIL 582	236	**211**	**211**	211	800	0.00%
RAIL 2536	770	**691**	692	691	800	0.00%
RAIL 2586	1064	945	**936.1**	936	800	0.00%
RAIL 4284	1215	**1064**	1070	1066	800	0.19%
RAIL 4872	1698	**1528**	1534	1530	800	0.13%

In Lagrangian-based heuristic the standard problem are categorized and the fixed time limit is set for each category. For the small problems, time limit is set in 3000 seconds and for the medium and large problems, time limit is set in 10,000 seconds. In 3-flip neighborhood local search, Time limits of the algorithms set to 180 seconds for types E–H, 600 seconds for RAIL 507, 516 and 582, and 18,000 s for RAIL 2536–4872.

The average of IMPROVE column for the proposed local branching algorithm is -0.66 percent. The results show the efficiency and effectiveness of the proposed algorithm. Figure 2 shows the best objective value of the local branching algorithm in each branching node for CYC.09 problem.

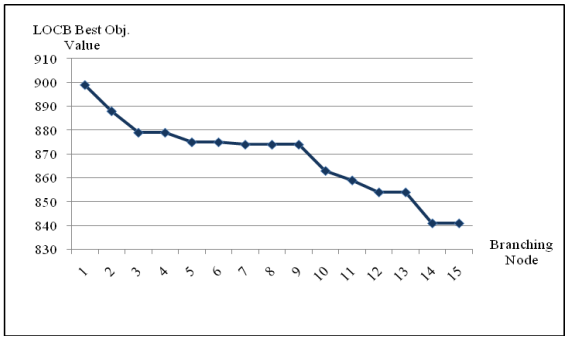

Fig. 2. Convergence of LOCB best objective value for CYC.09

5. Conclusions

This paper presented the local branching algorithm for solving the set covering problem. The validity and efficiency of the proposed method are put into test over a series of computational experiments on fourteen sets of standard test problems. To adjust the best parameter values in the proposed algorithm, design of experiments method is used to find the most appropriate parameters. The experimental results show the efficiency and effectiveness of the proposed algorithm. The average percentage of improvement for the proposed algorithm in compare with the best solution in the literature is -0.66 percent. The outcome is the local branching method clearly outperforms other heuristics in the literature, finding the best solution until now for most of the instances with a reasonable computational effort. These results are very encouraging, and suggest that combining mathematical programming and metaheuristic techniques is a worth pursuing research direction. The application of this formulation and solution method in real problems as a case study is suggested for future researches.

References

[1] Ablanedo-Rosas, J.H., Rego, C., "*Surrogate Constraint Normalization for the Set Covering Problem*", European Journal of Operational Research, Vol. 205, 2010, pp. 540–551.

[2] Garey, M.R., Johnson, D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, San Francisco, 1979.

[3] Balas, E., Carrera, M.C., "*A Dynamic Subgradient-Based Branch-and-Bound Procedure for Set Covering*", Operations Research, Vol. 44, 1996, pp. 875–890.

[4] Fisher, M., Kedia, P., "*Optimal Solution of Set Covering/Partitioning Problems Using Dual Heuristics*", Management Science, Vol. 36, 1990, pp. 674–688.

[5] Avella, P., Boccia, M., Vasilyev, I., "*Computational*

Experience with General Cutting Planes for the Set Covering Problem", Operations Research Letters, Vol. 37, 2009, pp. 16–20.

[6] Bjö rklund, P., Värbrand, P., Yuan, D., *"A Column Generation Method for Spatial TDMA Scheduling in ad Hoc Networks"* Ad Hoc Networks, Vol. 2, 2004, pp. 405–418.

[7] Hemazroa, T.D., Jaumardb, B., Marcotte, O., "*A Column Generation and Branch-and-Cut Algorithm for the Channel Assignment Problem*", Computers & Operations Research, Vol. 35, 2008, pp. 1204–1226.

[8] Galinier, P., Hertz, A., *"Solution Techniques for the Large Set Covering Problem*", Discrete Applied Mathematics, Vol. 155, 2007, pp. 312–326.

[9] Fisher, M.L., Rinnooy Kan, A.H.G., "*The Design, Analysis and Implementation of Heuristics*", Management Science, Vol. 34, 1988, pp. 263–265.

[10] Chvátal, V., "*A greedy Heuristic for the Set Covering Problem*", Mathematics of Operations Research, Vol. 4, 1979, pp. 233–235.

[11] Balas, E., Ho, A., "*Set covering algorithms using cutting planes, heuristics, and subgradient optimization: a computational study*", Mathematical Programming, Vol. 12, 1980, pp. 37–60.

[12] Avis, D., "*A Note on Some Computationally Difficult Set Covering Problems*", Mathematical Programming, Vol. 18, 1980, pp. 138–145.

[13] Baker, E.K., "*Efficient Heuristic Algorithms for the Weighted Set Covering Problem*", Computers & Operations Research, Vol. 8, 1981, pp. 303–310.

[14] Vasko, F.J., Wilson, G.R., "*Using a Facility Location Algorithm to Solve Large Set Covering Problems*", Operations Research Letters, Vol. 3, 1984, pp. 85–90.

[15] Vasko, F.J., Wilson, G.R., "*An Efficient Heuristic for Large Set Covering Problems*", Naval Research Logistics Quarterly, Vol. 31, 1984, pp. 163–171.

[16] Caprara, A., Fischetti, M., Toth, P., "*A Heuristic Method for the Set Covering Problem*", Operations Research, Vol. 47, 1999, pp. 730–743.

[17] Ceria, S., Nobili, P., Sassano, A., "*Lagrangian-Based Heuristic for Large-Scale Set Covering Problems*", Mathematical Programming, Vol. 81, 1998, pp. 215–228.

[18] Umetani, S., Yagiura, M., "*Relaxation Heuristics for the Set Covering Problem*", Journal of the Operations Research Society of Japan, Vol. 50, 2007, pp. 350–375.

[19] Yagiura, M., Kishida, M., Ibaraki, T., "*A 3-Flip Neighborhood Local Search for the Set Covering Problem*", European Journal of Operational Research, Vol. 172, 2006, pp. 472–499.

[20] Naji-Azimi, Z., Toth, P., Galli, L., "*An Electromagnetism Metaheuristic for the Unicost Set Covering Problem*", European Journal of Operational Research, Vol. 205, 2010, pp. 290–300.

[21] Caprara, A., Fischetti, M., Toth, P., "*Algorithms for the Set Covering Problem*", Annals of Operations Research, Vol. 98, 2000, pp. 353–371.

[22] Beasley, J.E., Chu, P.C., "*A Genetic Algorithm for the Set Covering Problem*", European Journal of Operational Research, Vol. 94, 1996, pp. 392–404.

[23] Brusco, M.J., Jacobs, L.W., Thompson, G.M., "*A Morphing Procedure to Supplement a Simulated Annealing Heuristic for Cost- and Coverage-Correlated Set-Covering Problems*", Annals of Operations Research, Vol. 86, 1999, pp. 611–627.

[24] Caserta, M., Tabu search-based metaheuristic algorithm for large-scale set covering problems. In: Doerner, K.F., Gendreau, M., Greistorfer, P., Gutjahr, W.J., Hartl, R.F., Reimann, M. (eds.), *Metaheuristics: Progress in Complex Systems Optimization*. New York, Springer, 2007, pp. 43–63.

[25] Lessing, L., Dumitrescu, I., Stützle, T. "*A Comparison Between ACO Algorithms for the Set Covering Problem*", Lecture Notes in Computer Science, Vol. 3172, 2004, pp.1–12.

[26] Ren, Z., Feng, Z., Zhang, Z., "*New Ideas for Applying Ant Colony Optimization to the Set Covering Problem*", Computers & Industrial Engineering, Vol. 58, 2010, pp. 774–784.

[27] Gouwanda, D., Ponnambalam, S.G., *"Evolutionary Search Techniques to Solve Set Covering Problems",* Proceedings of World Academy of Science, Engineering and Technology, Vol. 29, 2008, pp. 20–25.

[28] Aickelin, U., "*An Indirect Genetic Algorithm for Set Covering Problems*", Journal of the Operational Research Society, Vol. 53, 2002, pp. 1118–1126.

[29] Solar, M., Parada, V., Urrutia, R., "*A Parallel Genetic Algorithm to Solve the Set Covering Problem*", Computers & Operations Research, Vol. 29, 2002, pp. 1221–1235.

[30] Lan, G., DePuy, G.W., Whitehouse, G.E., "*An Effective and Simple Heuristic for the Set Covering Problem*", European Journal of Operational Research, Vol. 176, 2007, pp. 1387–1403.

[31] Raja Balachandar, S., Kannan, K., "*A Meta-Heuristic Algorithm for Set Covering Problem Based on Gravity*", International Journal of Computational and

Mathematical Sciences, Vol. 4, 2010, pp. 223–228.

[32] Fischetti, M., Lodi, A., "*Local Branching*", Mathematical Programming, Vol. 98, 2003, pp. 23–47.

[33] Montgomery, D.C., *Design and Analysis of Experiments*, John Wiley & Sons, 2009.

[34] Adenso-Díaz, B., Laguna, M., "*Fine-Tuning of Algorithms using Fractional Experimental Designs and Local Search*", Operations Research, Vol. 54, 2006, pp. 99–114.

[35] Ridge, E., *Design of Experiments for the Tuning of Optimization Algorithms*. PhD thesis, Department of Computer Science, University of York, United Kingdom, 2007.