# Effective Approaches to Solve P-Center Problem via Set Covering and SAT

Xiaolu Liu, Yuan Fang, Jiaming Chen, Zhouxing Su, Chumin Li, Zhipeng Lu

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 4, 2020, accepted August 16, 2020, date of publication August 31, 2020, date of current version September 15, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.3018618*

# Effective Approaches to Solve *P*-Center Problem via Set Covering and SAT

**XIAOLU LIU[1], YUAN FANG[2], JIAMING CHEN[1], ZHOUXING SU[2], CHUMIN LI[3], AND ZHIPENG LÜ[2]**

[1]College of System Engineering, National University of Defense Technology, Changsha 410000, China
[2]SMART, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
[3]MIS, University of Picardie Jules Verne, 80090 Amiens, France

Corresponding author: Jiaming Chen (chen.jiaming568@foxmail.com)

**ABSTRACT** The classic $p$-center problem consists of choosing a set of $p$ vertices in an undirected graph as facilities in order to minimize the maximum distance between each client vertex and its closest facility. The problem is equivalent to covering all vertices by no more than $p$ circles with the smallest possible radius, which can be tackled by solving a series of the decision version of set covering subproblems with the same cardinality constraint ($\leq p$) and gradually decreasing the covering radius. In this paper, we solve the $p$-center problem via set covering and SAT. We first transform the $p$-center problem into a series of set covering subproblems and simplify them by some reduction rules. Then, we present two kinds of encoding methods to convert them into CNF format and solve them with several state-of-the-art SAT solvers. Tested on three sets of totally 70 benchmark instances, our proposed approach can improve the previous best known results for 3 instances using the heuristic SAT solvers while proving the optimality for 59 instances using the exact SAT solvers. The computational results demonstrate the effectiveness of the proposed approach in terms of both solution quality and computational efficiency. In addition, the main advantage of our approach is twofold: The independence of the subproblems allows the problem to be solved in parallel; The approach to transform the original problem into SAT is flexible such that various state-of-the-art SAT solvers can be used.

**INDEX TERMS** $p$-center problem, set covering, SAT, encoding method, reduction rule.

## I. INTRODUCTION

The $p$-center problem is a classical facility location problem which has important applications in the fields of telecommunication industry, transportation science, public services and so on. It consists of seeking the location of $p$ facilities and assigning clients to them in order to minimize the maximum distance between a client and its nearest facility.

Hakimi [1] first introduced the absolute center problem to find a center $C$ in a graph $G$ such that the maximum distance from $C$ is minimized. The absolute center problem can be considered as the $p$-center problem when $p = 1$. However, there may be multiple centers that need to locate in real applications. Therefore, Hakimi [2] extended the absolute center problem to the $p$-center problem for more general applications. The $p$-center problem has been proven to be NP-hard [3] and many algorithms including exact and heuristic methods have been proposed for solving this problem.

The associate editor coordinating the review of this manuscript and approving it for publication was Juan Liu.

Most exact methods are based on solving a finite series of related subproblems, for example, the set covering problem. Minieka [4] proposed the first set covering based approach to solve the $p$-center problem by decreasing the covering radius step by step until the optimal value of the set covering subproblem within the current radius is greater than $p$. Garfinkel *et al.* [5] improved the approach proposed by Minieka [4] by using a binary search technique for the selection of the covering radius to reduce the search space. Elloumi *et al.* [6] proposed a new integer programming formulation (ELP) which is also based on the set covering subproblem and is better than the classical formulation proposed by Daskin [7]. Calik and Tansel [8] developed new integer programming formulations and an exact algorithm based on decomposition to their models (IP) for solving the $p$-center problem, which provides a tighter lower bound than the approach proposed by Elloumi *et al.* [6]. Daskin [9] presented an algorithm where the covering radius is also searched by a binary search, but the subproblem is replaced by the maximal set covering problem which consists of maximizing the number of covered clients by no more than $p$ facilities within

the given radius. Ilhan and Pinar [10] proposed a two-phase method based on a feasibility subproblem where it checks whether it is feasible to cover all the clients by no more than $p$ facilities within a given radius. In this method, it first solves the linear programming (LP) feasibility problems to obtain a suitable lower bound and then solves a series of IP feasibility problems starting from the lower bound. Al-Khedhairi and Salhi [11] proposed modifications (IP* and Daskin*) to the previous algorithms: They enhanced the algorithm proposed by Ilhan and Pinar [10] by reducing the number of ILP iterations needed to find the optimal solution, and modified the approach proposed by Daskin [7] with tighter initial lower and upper bounds and a more appropriate binary search method which can reduce the number of subproblems to be solved.

There are also various effective heuristic and metaheuristic algorithms for the $p$-center problem. Mladenovic *et al.* [12] presented a basic variable neighborhood search (VNS) and two tabu search (TS) heuristics for this problem. Their experiments showed that the proposed VNS and two TS algorithms are superior to previous classical heuristics, and VNS performs the best on average in terms of both the solution quality and computational time. However, TS performs slightly better for the instances with smaller $p$ values. Pullan [13] proposed a memetic genetic algorithm (PBS), which is a population-based meta-heuristic that uses phenotype crossover and directed mutation operators to generate new starting points for a local search. For larger $p$-center instances, PBS is able to effectively utilize a number of computer processors. It can get high quality solutions for the classical benchmarks. Yin *et al.* [14] proposed a greedy randomized adaptive search procedure with path-relinking (GRASP/PR) algorithm, which combines both GRASP and path-relinking. In their algorithm, each iteration of GRASP/PR consists of the construction of a randomized greedy solution, followed by a tabu search procedure. GRASP/PR was considered to be the best heuristic for it can obtain the optimal solutions for most of the public benchmarks.

Satisfiability (SAT) is the problem of determining if there exists an assignment of boolean variables that satisfies a given Boolean formula and is the first problem proven to be NP-complete [15]. There are many literatures about solving combinatorial optimization problems based on SAT. Van [16] proposed a method for solving graph coloring problem by encoding it into SAT. They improved the previous lower bounds for the classical benchmarks such as DSJC125.5. Soh *et al.* [17] proposed a SAT based exact approach for solving the two-dimensional strip packing problem (2SPP). They showed that their method is competitive with the previous state-of-the-art 2SPP methods. In particular, they found better solution for instance HT08 than the best solution found by the previous exact and heuristic methods.

In this paper, we present a method for solving the $p$-center problem via set covering and SAT, since the SAT problem has been intensively studied in the academic society over the last decades and a lot of progress has been made. Thus, various state-of-the-art SAT solvers can be used, which is an advantage of our method. To our best knowledge, our paper establishes a bridge between the $p$-center problem and SAT for the first time. We first transform the problem to a finite series of the decision version of set covering subproblems with the same cardinality constraint ($\leq p$) and gradually decreasing covering radius. Then, we simplify these set covering subproblems by some data reduction rules which were proposed by Alber *et al.* [18] and encode them into CNF format with two different encoding modes. We then solve the $p$-center problem by solving these encoded subproblems using the state-of-the-art SAT solvers. Note that these encoded subproblems are independent and can be solved in parallel. Our method is tested on three sets of totally 70 classical benchmarks of $p$-center problem and it can obtain the optimal or best known solutions for all the 70 instances, where the previous best known results are improved for 3 instances. Furthermore, we can prove the optimality for 59 out of 70 instances. To the best of our knowledge, there are some instances which are proven to optimality for the first time.

The rest of this paper is organized as follows. Section II describes the formal definition of the $p$-center problem and its decision version problem. Section III presents the method for solving the $p$-center problem via set covering and SAT. Section IV reports the computational results and comparison with the state-of-the-art algorithms in the literature. Section V analyzes and discusses the importance of data reduction, how to prove the optimality for the $p$-center problem by SAT and the performance differences of the two encoding modes used in our algorithm. Section VI summarizes the main contribution of this work and concludes the paper.

## II. PROBLEM DEFINITION

Given a complete undirected graph $G = (N, E)$, where $N$ is the set of nodes and $E$ is the set of edges. The distance between any two nodes $u$ and $v$ is denoted by $d(u, v)$. The $p$-center problem is to find a set of facilities $S \subseteq N$, such that $|S| = p$ and the objective function:

$$f = max_{v \in N} min_{u \in S} d(u, v)$$

is minimized.

The $p$-center problem can be transformed from an optimization problem to a series of decision subproblems. Assuming that all distinguishing lengths of edges in $G$ are sorted in a decreasing order, i.e., $d_1 > d_2 > \ldots > d_K$. Then, we choose $d_i$ $(i = 1, 2, \ldots, K)$ in decreasing order as the coverage radius and remove all the edges whose lengths are larger than $d_i$. For every $d_i$, we judge if there is a feasible solution for the $p$-center problem. We repeatedly choose next $d_i$ until $d_i$ is invalid, i.e., there does not exist a set $S \subseteq N$ which satisfies both $|S| = p$ and $max_{v \in N} min_{u \in S}$ $d(u, v) \leq d_i$. Then, $d_{i-1}$ is the best solution for the original $p$-center problem.

The methodology of solving an optimization problem by transforming it into a series of decision problems is a popular

technique for solving challenging combinatorial optimization problems [19]–[21]. The reason lies in the fact that some optimization problems are much more difficult to solve than their corresponding decision problems. By transforming an optimization problem to a decision one, the search space of the problem becomes greatly narrowed and restricted. In addition, it also becomes easy to identify the reason to cause the infeasibility of a decision problem, which can guide the search effectively and efficiently. For these reasons, we employ this methodology to solve the $p$-center problem.

## III. SOLVING P-CENTER PROBLEM VIA SET COVERING AND SAT

The subproblem in the decision version of the $p$-center problem can be transformed to the decision version of set covering problem, which can be encoded into CNF format and then solved with SAT solvers. Based on this idea, the $p$-center problem can be solved via set covering and SAT.

### A. TRANSFORMING THE SUBPROBLEM TO SET COVERING

Ilhan and Pinar [10] first transformed the subproblem of $p$-center problem to the decision version of the set covering problem, which was formulated as an IP model. Based on the model, we make a small modification such that it can be encoded into CNF format. Given a complete graph $G = (N, E)$ and a coverage radius $d$, let $E' = \{(u, v)|(u, v) \in E \text{ and } d(u, v) \leq d\}$ such that the induced graph $G' = (N, E')$ used for the set covering subproblem can be obtained by removing the edges that cannot cover a node, i.e., whose length is greater than $d$. For each node $u \in N$, we can get a set $S_u \subseteq N$ such that for each node $v \in S_u$ the distance between $u$ and $v$ is less than or equal to $d$. Then, we can obtain the set $F = \{S_1, S_2, \ldots, S_u, \ldots, S_n\}$, where $S_u = \{u\} \cup \{v \in N|(u, v) \in E'\}$. Thus, judging if there exists a set $S \subseteq N$, such that $|S| = p$ and $max_{v \in N} min_{u \in S} d(u, v) \leq d$ is equivalent to judging if there exists $p$ sets in $F$ which can cover all the nodes, i.e., if there exists a subset $F' \subseteq F$ which satisfies $N = \cup_{S_u \in F'} S_u$ and $|F'| = p$. From another perspective, for each node $u \in N$, $F'$ must have at least one set that contains $u$. Furthermore, if $F'$ can cover all the nodes by less than $p$ sets, we can add arbitrarily $p - |F'|$ sets to $F'$ and $F'$ can definitely cover all the nodes. The cardinality constraint $|F'| = p$ can be relaxed to $|F'| \leq p$. Let $x_c$ be a binary variable which means that $S_c$ belongs to $F'$ if $x_c = 1$, we can get the following model.

$$\sum_{c \in \{c|u \in S_c\}} x_c \geq 1, \quad u = 1, 2, \ldots, n \tag{1}$$

$$\sum_{c=1}^{n} x_c \leq p \tag{2}$$

In this model, constraint (1) denotes that for each node $u \in N$, there exists at least one set $S_c$ which contains $u$ belonging to $F'$, and constraint (2) restricts that we can choose at most $p$ sets from $F$. Figure 1 shows an example of transforming a $p$-center problem into set covering problems with different radii.

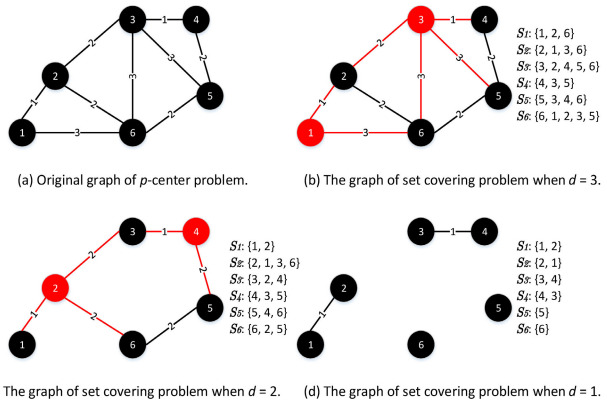


**FIGURE 1. An example of transforming a $p$-center problem into set covering problems with different radii where $p = 2$. (a) shows the original graph, while (b), (c) and (d) present the graphs of its corresponding set covering problems when $d = 3, 2$ and 1, respectively. For the transformed set covering problems, we can obtain a feasible solution by choosing vertices 1 and 3 (labeled with red) in (b), and vertices 2 and 4 in (c) as centers, while there is no feasible solution in (d).**

### B. ENCODING SET COVERING INTO CNF

In order to solve the subproblem with SAT solvers, we need to encode the model into CNF format where the formula is conjunction of clauses and each clause is a disjunction of literals; each literal is a propositional variable $x$ or its negation $\neg x$.

For constraint (1), we can encode it with:

$$\bigvee_{c \in \{c|u \in S_c\}} x_c, \quad u = 1, 2, \ldots, n \tag{3}$$

Since the graphs of different set covering subproblems of a $p$-center problem are different, constraint (1) may be different for a vertex in two different subproblems. For example, there are three sets ($S_1, S_2$ and $S_6$) which contain vertex 1 in Figure 1-(b). Thus, we can obtain a clause $(x_1 \vee x_2 \vee x_6)$ for vertex 1 to ensure it to be covered, while the clause for vertex 1 in Figure 1-(c) is $(x_1 \vee x_2)$.

Constraint (2) is the cardinality constraint, which is independent of the specific graphs of both the original $p$-center problem and the transformed set covering problems. That is to say, it is only related to the values of $n$ and $p$. We encoded the cardinality constraint in sequential counter and parallel counter encoding modes to maintain the cardinality constraint.

The sequential counter encoding mode was introduced in Sinz [22] for the $\sum_{c=1}^{n} x_c \leq p$ cardinality constraint. The encoding mode works by encoding a sequential counter circuit (Figure 2) that sequentially calculates the sums $s_i = \sum_{j=1}^{i} x_j$ which is represented as unary number with $p$ bits $(s_{i,1}, \ldots, s_{i,p})$, and sets the overflow bits $v_i$ to be true if $s_i$ exceeds $p$ $(i = 1, 2, \ldots, n)$. In addition, all of the overflow bits $v_i$ should be zero to ensure that $s_i \leq p$. Then, the cardinality constraint $\sum_{c=1}^{n} x_c \leq p$ can be encoded with $2np + n - 3p - 1$ clauses and $n + (n - 1)p$ variables. For example, the problem in Figure 1 with $n = 6$ and $p = 2$ will introduce 23 clauses and 16 variables. For the sake of space limit, we show a small example with $n = 3$

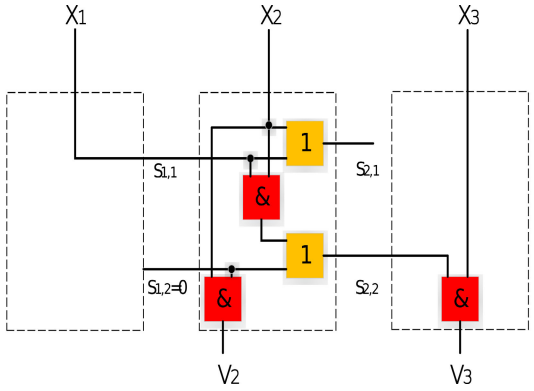


**FIGURE 2.** The sequential counter circuit when $n = 3$ and $p = 2$.

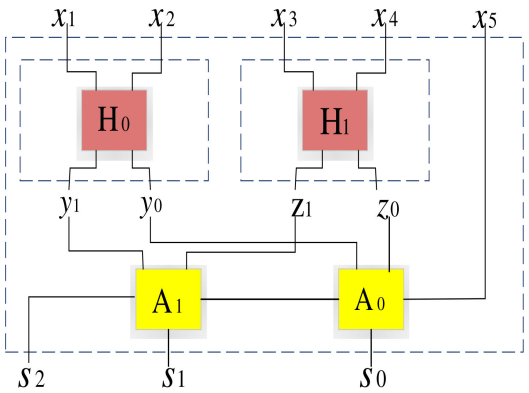


**FIGURE 3.** The parallel counter circuit when $n = 5$. The sub-circuits $H_i$ are half-adders and the $A_i$ are full-adders.

and $p = 2$, where the constraint can be encoded with 8 clauses and 7 variables as follows:

$$\begin{aligned} A_s &= \{(\neg x_1 \vee s_{1,1}), (\neg s_{1,2})\}, \\ B_s &= \{(\neg x_2 \vee s_{2,1}), (\neg s_{1,1} \vee s_{2,1}), \\ &\quad (\neg x_2 \vee s_{1,1} \vee s_{2,2}), (\neg s_{1,2} \vee s_{2,2})\}, \\ C_s &= \{(\neg x_2 \vee \neg s_{1,2}), (\neg x_3 \vee \neg s_{2,2})\}. \end{aligned}$$

The clause set $A_s$ calculates the sum $s_1$ which is decided only by $x_1$, and $B_s$ calculates the sum $s_2$ which is decided by both $x_2$ and $s_1$. In addition, the clause set $C_s$ is to set the overflow bits $v_2$ and $v_3$ to be zero.

Sinz [22] also introduced another encoding mode based on a parallel counter circuit. The counter (Figure 3) recursively splits the $n$-bit input bits $x_i$ $(i = 1, \ldots, n)$ into two halves where the number of true variables is respectively counted. The results of the two halves are represented by $m$-bit $(m = log_2\lfloor n \rfloor)$ binary numbers and added by a standard binary adder to get the result of the current recursive layer. In addition, the $(m + 1)$-bit output of the uppermost layer is the result of the whole circuit. The counter is completely implemented by full-adder and half-adder, both of which can be encoded based on the well-known equations, where each half-adder $(a \oplus b)$ can be encoded by three clauses:

$$\{(a \vee \neg b \vee s_{out}), (\neg a \vee \neg b \vee c_{out}), (\neg a \vee b \vee s_{out})\},$$

and each full-adder $(a \oplus b \oplus c)$ can be encoded by seven clauses:

$$\begin{aligned} &\{(a \vee b \vee \neg c \vee s_{out}), (\neg a \vee b \vee c \vee s_{out}), \\ &\quad (a \vee \neg b \vee c \vee s_{out}), (\neg a \vee \neg b \vee \neg c \vee s_{out})\}, \\ &\{(\neg a \vee \neg b \vee c_{out}), (\neg a \vee \neg c \vee c_{out}), (\neg b \vee \neg c \vee c_{out})\}. \end{aligned}$$

In addition, there is a comparator circuit which forces the $(m + 1)$-bit output value of the parallel counter to be not greater than $p$, and can be encoded by at most $m + 1$ clauses. The encoding of the $\sum_{c=1}^{n} x_c \leq p$ constraint based on the circuit below requires at most $7n - 3\lfloor log_2 n \rfloor - 6$ clauses and $3n - 2$ variables. For example, when $n = 3$ and $p = 2$, the constraint can be encoded as follows:

$$\begin{aligned} A_p + &\{(x_1 \vee x_2 \vee \neg x_3 \vee s_0), (\neg x_1 \vee x_2 \vee x_3 \vee s_0), \\ &\quad (x_1 \vee \neg x_2 \vee x_3 \vee s_0), (\neg a \vee \neg b \vee \neg c \vee s_0)\}, \\ B_p = &\{(\neg x_1 \vee \neg x_2 \vee s_1), (\neg x_1 \vee \neg x_3 \vee s_1), (\neg x_2 \vee \neg x_3 \vee s_1)\}, \\ C_p = &\{\neg s_1 \vee \neg s_0\}. \end{aligned}$$

The clause sets $A_p$ and $B_p$ are generated from the parallel counter circuit which has just one full-adder. The clause set $C_p$ is to ensure the cardinality to be not greater than 2 which comes from the comparator circuit.

### C. DATA REDUCTION FOR SET COVERING

Before encoding the model of set covering into CNF, we can employ two reduction rules, which were introduced by Alber *et al.* [18], to reduce the graph in the preprocessing stage. These two rules are based on exploring local structures of the graph and try to replace them by simpler structures. It is worth mentioning that these rules are implemented with polynomial-time complexity and can ensure that the solution domain of the reduced graph is the same as the original graph for the set covering problem. So, these rules do not change the optimal solution for the set covering problem. Based on this, we can use these two rules to reduce the graph for each set covering subproblem.

Both rules can determine some vertices to be centers or not, and remove these determined vertices. Given a graph $G = (N, E)$ and a covering radius $d$, the induced graph $G' = (N, E')$ used for the set covering subproblem can be obtained by removing the edges that cannot cover a node, i.e., whose length is greater than $d$. For $v \in N$, let $\Gamma(v) = \{u | \{u, v\} \in E'\}$ be the neighborhood of $v$, and $\Gamma[v] = \Gamma(v) \cup \{v\}$. Rule 1 partitions $\Gamma(v)$ into three different sets called $\Gamma_1(v)$, $\Gamma_2(v)$ and $\Gamma_3(v)$, respectively. $\Gamma_1(v)$ consists of the vertices which are linked to at least one vertex not in $\Gamma[v]$, while the vertices in $\Gamma_2(v)$ are only linked to the vertices in $\Gamma[v]$ and are linked to at least one vertex in $\Gamma_1(v)$, and $\Gamma_3(v)$ contains the remaining vertices. Rule 1 suggests that, if $\Gamma_3(v)$ is not empty, the vertex $v$ has to be a center, and excluding the vertices in $\Gamma_2(v)$ and $\Gamma_3(v)$ from being centers never cuts off the optimal solution. Then, these determined vertices can be removed from $G$. Rule 2 partitions the neighborhood set $\Gamma(v, w) = \Gamma(v) \cup \Gamma(w)$ of vertices $v$ and $w$ in a similar way, but it judges if the vertices

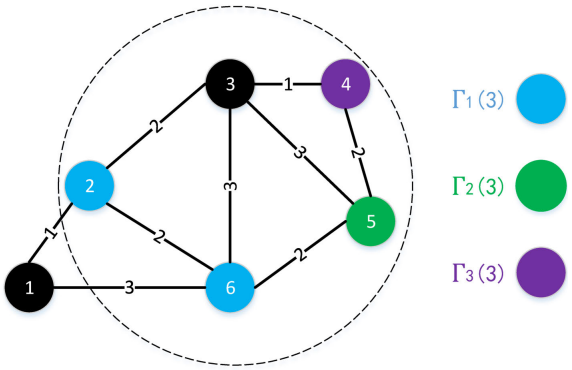


**FIGURE 4.** An example of the reduction rule 1 based on the graph in Figure 1-(b).

can be fixed or removed in a more complex way. Interested readers are referred to [18] for more details.

Let $N_c$ and $N_d$ respectively denote the set of vertices which are determined to be centers, and the set of vertices which are determined not to be centers. The solution of the set covering subproblem with graph $G$ and covering radius $d$ contains all the vertices in $N_c$ and does not contain any vertex in $N_d$. Note that each vertex in $N_d$ can be covered by one vertex in $N_c$, so we need to find at most $p - |N_c|$ centers from $N \backslash (N_c \cup N_d)$ to cover all the vertices of $N \backslash (N_c \cup N_d)$. By doing this, all the vertices of $N$ can be covered by a center and the number of centers is no more than $p$. Let $G'' = (N'', E'')$ and $S''_u = \{u\} \cup \{v \in N'' | (u, v) \in E''\}$, where $N'' = N \backslash (N_c \cup N_d)$ and $E''$ is obtained by removing all the edges related to the vertices in $N_c \cup N_d$, we will solve the following set covering model via SAT solvers:

$$\sum_{c \in \{c | u \in S_c\}} x_c \geq 1, \quad u \in N'' \tag{4}$$

$$\sum_{c=1}^{n} x_c \leq p - |N_c| \tag{5}$$

Figure 4 gives an example of rule 1 based on the graph in Figure 1-(b), which shows that the neighborhood of vertex 3 is partitioned into $\Gamma_1(3) = \{2, 6\}$, $\Gamma_2(3) = \{5\}$ and $\Gamma_3(3) = \{4\}$. Therefore, after employing the reduction rules for the graph in Figure 4, vertex 3 is added into $N_c$, while vertices 2, 4, 5 and 6 are added into $N_d$. Then, we can obtain a reduced graph which only contains vertex 1 by removing $N_c$ and $N_d$ and the set covering model will become $x_1 \geq 1$ and $x_1 \leq 1$, which can be encoded with only one clause $(x_1)$. Thus, $\{x_3, x_1\}$ is a feasible solution for this example.

## IV. COMPUTATIONAL RESULTS

In this section we report the experimental results of different SAT solvers on the classical standard benchmark instances of the $p$-center problem and compare the performance of these SAT solvers with the state-of-the-art algorithms.

### A. SAT SOLVERS

The famous SAT Competition has been organized for 21 times to solve the satisfiability problem since 1992. There are many state-of-the-art SAT solvers submitted to these SAT competitions. One advantage of our method is that it can utilize different SAT solvers in parallel. To solve our problem, we test 7 SAT solvers on the instances from OR-lib [23] and TSP-Lib [24] and eventually choose 3 representative solvers including Maple_CM solver [25], MapleLCMDistChronoBT solver [26] and Sparrow2Riss-2018 solver [27], [28].

The Maple_CM solver is a new conflict-driven clause learning (CDCL) solver which was developed by extending clause minimization to original clauses based on Maple_LCM [29]. The Maple_LCM solver got the first place of the main track in SAT Competition 2017 which was obtained by implementing the learnt clause minimization approach by using unit propagation based on the solver MapleCOMSPS_DRUP [30], [31].

The MapleLCMDistChronoBT is the winner solver of the main track of SAT Competition 2018. The solver was based on the SAT Competition 2017 winner, Maple_LCM_Dist [29], and was updated with chronological backtracking (configuration $\{T = 100, C = 4000\}$) based on the results in [32].

The Sparrow2Riss-2018 solver is the first prize of the random track of SAT Competition 2018 which is a combination of the solvers Sparrow [33] and Riss. Sparrow is a Stochastic Local Search (SLS) solver that uses promising variables and probability distribution based selection heuristics, while Riss is a CDCL solver which is based on MINISAT [34] search engine and GLUCOSE 2.2 [35], [36]. As SLS solvers cannot prove unsatisfiability, they combined Sparrow and Riss by first trying to solve the SAT problem with Sparrow, which is limited for the execution with $5 \cdot 10^8$ flips, and then running Riss which starts from the solution obtained by Sparrow. By doing this, the solver's overall behavior can still be deterministic. However, Gaussian Elimination and Cardinality Constraint reasoning are applied in the solver for our experiments, which means that it cannot emit proof for unsatisfiable instances.

### B. PROBLEM INSTANCES AND EXPERIMENTAL PROTOCOL

We carry out computational experiments on three representative sets of instances from OR-Lib [23] and TSP-Lib [24], respectively, which were widely used by previous state-of-the-art reference algorithms.

#### 1) OR-LIB INSTANCES

The first set of instances consists of 40 randomly generated instances with $|N|$ ranging from 100 to 900 and $p$ ranging from 5 to 90. The graphs in this set are not complete, so we need to calculate the shortest path between each pair of nodes to get a complete graph.

#### 2) TSP-LIB INSTANCES

Both the second and third sets of instances are from TSP-Lib, which are real world application instances from the task of drilling holes in printed circuit boards and are usually used as the benchmarks for various routing and location problems. The two sets respectively consist of 15 instances from u1060

($|N| = 1060$) and 15 instances from u1817 ($|N| = 1817$). For these 30 instances, $p$ ranges from 10 to 150 and the nodes are given with two-dimensional coordinates. Thus, we need to calculate the Euclidean distance between each pair of nodes to get the whole distance matrix, i.e., a complete graph.

Our algorithm was programmed in C++ and the experiments were conducted on a Linux PC with Intel Xeon CPU E5-2609 v2 2.50 GHZ processor and 32 GB RAM with 8 cores. The time limit for each given radius of each instance was set to 4 hours. Besides, we test each instance for five times to get the average running time.

### C. EXPERIMENTAL DESIGN

For each instance, we try all possible covering radii from the previous best radius reported in the literature. Due to the independence of set covering subproblems, we solve 8 (the number of cores) subproblems with different covering radii of an instance at the same time. Once a feasible solution is obtained for a certain radius, all the subproblems with larger radii are stopped and the subsequent smaller radii are tried. The computation of an instance stops if a subproblem with a radius $d_{i-1}$ outputs 'SAT' and a subproblem with a radius $d_i$ outputs 'UNSAT', or when the time limit is met. For each given radius, we can get a graph where all the edges whose lengths are larger than the given radius are deleted. First, we employ the data reduction rules mentioned in Section III-C for the graph to get a simpler one. Next, we encode the instance for the given radius with a simplified graph into CNF format by sequential counter and parallel counter. So, there are two different SAT instances to solve with SAT solvers for each given radius. We finally solve these SAT problems transformed from set covering subproblems to judge if the given radius is feasible for the $p$-center problem by using Maple_CM, MapleLCMDistChronoBT and Sparrow2Riss-2018.

Note that a SAT instance encoding a $p$-center instance consists of a relatively small subset of very long clauses only containing positive literals (constraint 1) and a large subset of clauses encoding the cardinality constraints (constraint 2). This fact is not exploited by the general-purpose SAT solvers we test in this paper, but could be exploited by a SAT solver specialized to solve the $p$-center problem.

### D. RESULTS OF SAT SOLVERS

In this section, we conduct experiments for solving the feasibility subproblems by the three SAT solvers on the three sets of 70 instances. Table 1 shows the computational results of the three solvers on instances from OR-Lib for the given radius, while Table 2 and Table 3 respectively show the results on the sets of u1060 and u1817 instances from TSP-Lib. In these three tables, columns SEQ and PAR in Maple_CM, MapleLCMDistChronnoBT and Sparrow2Riss-2018 show the average CPU time ('−' means timeout) for each given radius in the sequential counter encoding and the parallel counter encoding modes. Note that there is just one row for one instance if all the solvers can get the same minimum radius for both encoding modes in the given time limit. Otherwise, there will be multi-row results for one instance with different covering radius. From Tables 1–3, we can get the minimum radius for all the instances in different encoding modes that each solver can obtain within the time limit. The minimum radius of one instance in certain encoding mode for one solver is the best solution of the $p$-center problem that the solver can obtain for the instance in the encoding mode. For example, the best solution of pmed33 obtained by Maple_CM or MapleLCMDistChronnoBT in any encoding mode is 29, while Sparrow2Riss-2018 can get 28 in the SEQ encoding mode. Row Num. shows the number of instances for which the solver can solve the instances to optimality or reach the best known results and row Avg. gives the average time of solving the instances to their corresponding best results.

From Table 1, one observes that in the same encoding mode, for both SEQ and PAR, Maple_CM and MapleLCMDistChronnoBT can get the same best solution for all the 40 instances, which means that Maple_CM is consistent with MapleLCMDistChronnoBT in terms of the solution quality on these small instances in OR-Lib. Sparrow2Riss-2018 has a little worse performance in terms of the solution quality and computational time than other two solvers in PAR encoding mode, but outperforms them in SEQ encoding mode for it can obtain better results for 9 instances in the SEQ mode. Besides, Sparrow2Riss-2018 in the SEQ mode outperforms other two solvers for it has the smallest average computational time. This experiment demonstrates that both encoding mode and SAT solver have important affects on the search performance of the algorithm.

From Table 2 and Table 3, one observes that in the same encoding mode, whether SEQ or PAR, all the three solvers perform the same in terms of the solution quality. It is worth mentioning that they can obtain the optimal solutions for all the 30 large instances from TSP-Lib in the SEQ encoding mode. In addition, Maple_CM outperforms MapleLCMDistChronnoBT in terms of the computational time to reach the best results on the set of u1060 instances, while it is opposite on the set of u1817 instances. Both of them are better than Sparrow2Riss-2018 in terms of the computational time.

### E. COMPARISON WITH OTHER REFERENCE ALGORITHMS

In this subsection, we compare the best results of the three SAT solvers with three exact algorithms (ELP, IP* and Daskin*) and two metaheuristic algorithms (PBS and GRASP/PR) in the literature. The total computational time of each instance using SAT solvers can be taken as the computational time of the set covering subproblem with the smallest radius that can be solved for the following reasons: 1) The set covering subproblems with different radii of an instance can be solved in parallel. 2) The initial coverage radius is set as the previous best known result for each instance, which is a common setting in solving many combinatorial optimization problems, such as graph coloring [19], maximum clique [37], etc. 3) The number of radii between the previous best known

**TABLE 1.** Computational results of three SAT solvers on the set of 40 instances from OR-Lib.

| Instance | $n$ | $p$ | $radius$ | Maple_CM (s) | | MapleLCMDistChronnoBT (s) | | Sparrow2Riss-2018 (s) | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | SEQ | PAR | SEQ | PAR | SEQ | PAR |
| pmed01 | 100 | 5 | 127 | 0.05 | 0.09 | 0.03 | 1.10 | 0.82 | 0.54 |
| pmed02 | 100 | 10 | 98 | 0.13 | 0.22 | 0.66 | 2.65 | 0.69 | 0.20 |
| pmed03 | 100 | 10 | 93 | 0.06 | 0.24 | 0.6 | 0.20 | 0.43 | 2.93 |
| pmed04 | 100 | 20 | 74 | 0.03 | 0.02 | 0.03 | 0.04 | 0.3 | 0.20 |
| pmed05 | 100 | 33 | 48 | 0.02 | 0.02 | 0.02 | 0.02 | 0.11 | 0.10 |
| pmed06 | 200 | 5 | 84 | 0.86 | 0.45 | 5.05 | 4.78 | 1.05 | 0.54 |
| pmed07 | 200 | 10 | 64 | 0.75 | 0.28 | 8.11 | 3.86 | 0.47 | 3.17 |
| pmed08 | 200 | 20 | 55 | 0.28 | 0.13 | 5.76 | 102.32 | 0.29 | 12.73 |
| pmed09 | 200 | 40 | 37 | 0.06 | 0.04 | 2.43 | 2.48 | 0.45 | 0.19 |
| pmed10 | 200 | 67 | 20 | 0.05 | 0.05 | 0.03 | 0.04 | 0.53 | 0.31 |
| pmed11 | 300 | 5 | 59 | 1.85 | 2.19 | 11.35 | 14.38 | 2.33 | 7.27 |
| pmed12 | 300 | 10 | 51 | 1.86 | 2.71 | 9.33 | 10.63 | 1.26 | 17.74 |
| pmed13 | 300 | 30 | 36 | 3820.47 | 29.74 | 2721.83 | 2359.05 | 0.79 | 1602.73 |
| pmed14 | 300 | 60 | 26 | 1.55 | 0.71 | 106.15 | 4.35 | 0.75 | 6.09 |
| pmed15 | 300 | 100 | 18 | 0.06 | 0.02 | 0.09 | 0.07 | 0.49 | 0.17 |
| pmed16 | 400 | 5 | 47 | 0.22 | 0.12 | 3.27 | 5.93 | 5.06 | 2.72 |
| pmed17 | 400 | 10 | 39 | 51.97 | 7.00 | 40.2 | 18.29 | 2.51 | 33.81 |
| pmed18 | 400 | 40 | 29 | 2698.59 | 10.37 | 28.29 | 18.54 | 1.31 | 1.42 |
| | | | 28 | - | - | - | - | 2.67 | - |
| pmed19 | 400 | 80 | 19 | 1.2 | 0.59 | 9.38 | 3.39 | 1.32 | 0.41 |
| | | | 18 | - | - | - | - | 6.34 | - |
| pmed20 | 400 | 133 | 13 | 0.18 | 0.04 | 1.2 | 2.28 | 0.63 | 14.56 |
| pmed21 | 500 | 5 | 40 | 0.9 | 0.70 | 11.54 | 5.93 | 5.48 | 5.17 |
| pmed22 | 500 | 10 | 38 | 4365.15 | 527.03 | 920.76 | 966.53 | 24.79 | 8298.82 |
| pmed23 | 500 | 50 | 23 | 32.55 | 25.11 | 2540.55 | 14.07 | 1.88 | 18.53 |
| | | | 22 | - | - | - | - | 15.45 | - |
| pmed24 | 500 | 100 | 16 | 4.54 | 1.29 | 10.98 | 8.17 | 1.95 | 0.84 |
| | | | 15 | 3933.48 | 1334.35 | 7603.2 | 1069.37 | 1.39 | - |
| pmed25 | 500 | 167 | 11 | 0.44 | 0.05 | 3.44 | 2.97 | 0.46 | 3.03 |
| pmed26 | 600 | 5 | 38 | 1.04 | 0.89 | 10.47 | 10.66 | 3.87 | 2.73 |
| pmed27 | 600 | 10 | 32 | 235.3 | 31.83 | 507.57 | 332.48 | 4.27 | 9.34 |
| pmed28 | 600 | 60 | 19 | 4.57 | 0.68 | 11.12 | 5.78 | 2.23 | 0.55 |
| | | | 18 | - | - | - | - | 1.56 | - |
| pmed29 | 600 | 120 | 13 | 19.09 | 0.48 | 16.75 | 13.81 | 1.42 | 3.67 |
| pmed30 | 600 | 200 | 9 | 1.22 | 0.04 | 8.28 | 3.12 | 0.78 | 76.98 |
| pmed31 | 700 | 5 | 30 | 3.52 | 3.67 | 19.6 | 10.87 | 6.47 | 8.66 |
| pmed32 | 700 | 10 | 29 | 21.41 | 4408.36 | 3252.86 | 94.64 | 8.05 | 59.82 |
| pmed33 | 700 | 70 | 17 | 21.59 | 0.68 | 18.96 | 12.26 | 3.74 | 1.07 |
| | | | 16 | - | 210.51 | - | 493.53 | 3.56 | 7.49 |
| | | | 15 | - | - | - | - | 51.8 | - |
| pmed34 | 700 | 140 | 11 | 4018.62 | 84.00 | 32.36 | 23.72 | 2.83 | 4.08 |
| pmed35 | 800 | 5 | 30 | 2.6 | 1.91 | 21.27 | 10.59 | 5.35 | 7.26 |
| pmed36 | 800 | 10 | 28 | 4.33 | 13.08 | 51.74 | 18.08 | 12.98 | 6.63 |
| | | | 27 | - | - | - | - | 39.81 | - |
| pmed37 | 800 | 80 | 16 | 343.01 | 9.18 | 26.27 | 15.36 | 4.9 | 2.57 |
| | | | 15 | - | - | - | - | 10.53 | - |
| pmed38 | 900 | 5 | 29 | 1.59 | 0.88 | 1.18 | 5.12 | 9.78 | 5.74 |
| pmed39 | 900 | 10 | 24 | 2.62 | 0.65 | 17.98 | 16.61 | 17.6 | 16.11 |
| | | | 23 | - | - | - | - | 203.22 | - |
| pmed40 | 900 | 90 | 14 | 84.91 | 21.57 | 7397.21 | 36.99 | 11.82 | 2.03 |
| | | | 13 | - | - | - | - | 28.01 | - |
| Num. | | | | 31 | 31 | 31 | 31 | **40** | 30 |
| Avg. | | | | 531.77 | 207.69 | 494.37 | 163.94 | **11.32** | 339.71 |

radius and the best radius that our algorithm can obtain is less than 8 for each instance. Because the reduction rules are designed for the set covering subproblems, the reference algorithms cannot directly use these reduction rules since these reference algorithms solve the original $p$-center problem where the objective is to minimize the coverage radius. This is also one of the advantages of our approach that existing reduction rules for set covering can be used. For this reason, we just cited the existing results in the corresponding references.

Table 4 presents the best results of SAT solvers on the 40 OR-Lib instances and makes comparison with ELP, IP*, Daskin*, PBS and GRASP/PR. Table 5 and Table 6 respectively show the best results of SAT solvers on the u1060 and u1817 instances and make comparison with ELP, PBS and GRASP/PR. Columns $f_{best}$ and cpu respectively show the best values obtained by an algorithm and their corresponding computational time ($\epsilon = 0.001$). Row Num. presents the number of instances for which the optimal or the best known results can be obtained for the corresponding solver. As PBS, GRASP/PR and SAT solvers use floating point data but the reference exact algorithms use integer data, a floating point result is considered to be identical to the integer result if the rounded floating point result obtained by PBS, GRASP/PR

**TABLE 2.** Computational results of three SAT solvers on the set of instances from u1060.

| Instance | $n$ | $p$ | $radius$ | Maple_CM (s) | | MapleLCMDistChronnoBT (s) | | Sparrow2Riss-2018 (s) | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | SEQ | PAR | SEQ | PAR | SEQ | PAR |
| u1060 | 1060 | 10 | 2273.08 | 1.29 | 1.41 | 4.53 | 12.66 | 156.85 | 206.16 |
| u1060 | 1060 | 20 | 1580.80 | 8.03 | 20.42 | 18.36 | 26.74 | 154.08 | 182.95 |
| u1060 | 1060 | 30 | 1207.77 | 6.43 | 10.94 | 15.08 | 29.11 | 178.17 | 181.81 |
| u1060 | 1060 | 40 | 1020.56 | 9.17 | 18.09 | 22.13 | 19.74 | 194.75 | 170.74 |
| u1060 | 1060 | 50 | 904.92 | 14.91 | 68.39 | 20.23 | 64.77 | 194.37 | 204.01 |
| u1060 | 1060 | 60 | 781.17 | 9.80 | 25.71 | 12.12 | 17.52 | 221.21 | 164.36 |
| u1060 | 1060 | 70 | 710.75 | 8.46 | 26.34 | 13.71 | 25.37 | 314.30 | 165.44 |
| u1060 | 1060 | 80 | 652.16 | 14.41 | 18.81 | 14.04 | 17.99 | 460.26 | 177.11 |
| u1060 | 1060 | 90 | 607.87 | 10.50 | 3.20 | 6.43 | 4.30 | 258.24 | 145.02 |
| u1060 | 1060 | 100 | 570.01 | 16.28 | 6.05 | 7.05 | 9.64 | 170.50 | 144.72 |
| u1060 | 1060 | 110 | 538.84 | 24.54 | 6.46 | 11.79 | 11.19 | 441.83 | 144.87 |
| u1060 | 1060 | 120 | 510.27 | 13.97 | 4.57 | 11.91 | 6.78 | 450.63 | 147.70 |
| u1060 | 1060 | 130 | 499.65 | 11.53 | 2.45 | 6.52 | 3.52 | 72.21 | 137.82 |
| u1060 | 1060 | 140 | 452.46 | 4.03 | 0.59 | 1.66 | 3.38 | 46.56 | 136.15 |
| u1060 | 1060 | 150 | 447.01 | 4.11 | 0.38 | 2.72 | 3.80 | 5.52 | 135.28 |
| Num. | | | | 15 | 15 | 15 | 15 | 15 | 15 |
| Avg. | | | | **10.50** | 14.25 | 11.22 | 17.10 | 221.30 | 162.94 |

**TABLE 3.** Computational results of three SAT solvers on the set of instances from u1817.

| Instance | $n$ | $p$ | $radius$ | Maple_CM (s) | | MapleLCMDistChronnoBT (s) | | Sparrow2Riss-2018 (s) | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | SEQ | PAR | SEQ | PAR | SEQ | PAR |
| u1817 | 1817 | 10 | 457.91 | 14.62 | 14.32 | 42.61 | 39.86 | 219.16 | 263.48 |
| u1817 | 1817 | 20 | 309.01 | 139.85 | 250.93 | 80.93 | 380.74 | 404.22 | 937.79 |
| u1817 | 1817 | 30 | 240.99 | 190.07 | 1496.21 | 123.61 | 1208.47 | 773 | 8994.46 |
| u1817 | 1817 | 40 | 209.45 | 190.48 | 1690.86 | 47.09 | 203.85 | 834.74 | 3012.15 |
| u1817 | 1817 | 50 | 184.91 | 520.15 | 1245.65 | 173.81 | 1873.19 | 1409.76 | 10288.13 |
| u1817 | 1817 | 60 | 162.64 | 184.80 | 378.75 | 134.17 | 308.36 | 821.99 | 3006.40 |
| u1817 | 1817 | 70 | 148.11 | 79.56 | 126.39 | 46.17 | 165.86 | 945.34 | 832.86 |
| u1817 | 1817 | 80 | 136.79 | 43.64 | 104.271 | 31.44 | 66.17 | 974.71 | 752.08 |
| | | | 136.78 | 6684.76 | - | 2202.98 | - | 4014.11 | - |
| | | | 136.77 | 1412.67 | - | 2313.89 | - | 2753.98 | - |
| u1817 | 1817 | 90 | 129.52 | 331.26 | 99.52 | 146.26 | 148.82 | 1555.73 | 8982.34 |
| | | | 129.51 | 548.72 | - | 370.95 | - | 3846.73 | - |
| u1817 | 1817 | 100 | 126.99 | 200.96 | 49.91 | 124.24 | 49.25 | 1010.77 | 355.09 |
| u1817 | 1817 | 110 | 109.25 | 116.59 | 478.84 | 84.25 | 260.09 | 943.06 | 425.97 |
| u1817 | 1817 | 120 | 107.76 | 79.46 | 27.47 | 54.48 | 26.41 | 929.37 | 214.41 |
| u1817 | 1817 | 130 | 107.75 | 2829.27 | 87.93 | 151.78 | 52.9 | 1135.31 | 803.22 |
| | | | 107.27 | 2268.75 | - | 6943.52 | - | 2806.46 | - |
| | | | 104.74 | 12389.20 | - | 1558.68 | - | 6150.07 | - |
| | | | 104.73 | 1117.65 | - | 941.17 | - | 3593.87 | - |
| u1817 | 1817 | 140 | 101.6 | 679.78 | 743.96 | 385.56 | 197.89 | 1689.55 | 3651.55 |
| u1817 | 1817 | 150 | 91.6 | 269.70 | 314.06 | 227.99 | 983.45 | 1230.56 | 5204.02 |
| Num. | | | | 15 | 12 | 15 | 12 | 15 | 12 |
| Avg. | | | | 383.00 | 568.11 | **343.39** | 474.79 | 1427.07 | 3098.86 |

or SAT solvers equals the integer result obtained by the exact algorithms. There may be dozens of different edges in one unit of solution, so the solutions obtained by the reference exact algorithms are a little rough to some extent. Besides, the $f_{best}$ column of ELP shows the optimal values obtained by ELP, where the optimal solution that is not obtained by ELP is marked with"?".

From Table 4, one observes that all the algorithms can obtain the optimal solutions for all the 40 instances. We can also observe that the average time of SAT solvers for the 40 OR-Lib instances is faster than IP* and Daskin* but slower than ELP, PBS and GRASP/PR. The reason might lie in the fact that these instances are relatively small and easy and transforming these instances into SAT may miss the original problem structure such that algorithms specially designed for the $p$-center problem, such as PBS and GRASP/PR, can solve them more easily. Table 5 shows that the results of SAT solvers can match the previous best known results for all the u1060 instances. From Table 6, one observes that solving the $p$-center problem via SAT solvers can improve the previous best known results for 3 cases compared with GRASP/PR (u1817 with $p = 80, 130, 150$). In addition, for the instances of TSP-Lib, the SAT solvers can obtain the results showed in Tables 5 and 6 with less average computational time, and the results of TSP-Lib instances obtained by SAT solvers can be proven to be optimal solutions (See Section V-B for the proof).

In sum, we present the summarized computational statistics of all the solvers on the two datasets (OR-Lib and TSP-Lib) in Table 7, where column Sparrow gives the statistics of the Sparrow2Riss-2018 solver, column Exact shows the average statistics of the two exact SAT solvers (with the SEQ encoding), and column Ref presents the statistics of the best reference algorithm GRASP/PR since it outperforms

**TABLE 4. Comparison with the reference algorithms on the set of 40 instances from OR-Lib.**

| Instance | $n$ | $p$ | $f_{opt}$ | ELP | | IP* | | Daskin* | | PBS | | GRASP/PR | | SAT Solvers | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | $f_{best}$ | cpu | $f_{best}$ | cpu | $f_{best}$ | cpu | $f_{best}$ | cpu | $f_{best}$ | cpu | $f_{best}$ | cpu |
| pmed01 | 100 | 5 | 127 | 127 | 0.70 | 127 | 4.05 | 127 | 2.09 | 127 | $< \epsilon$ | 127 | $< \epsilon$ | 127 | 0.82 |
| pmed02 | 100 | 10 | 98 | 98 | 0.20 | 98 | 1.52 | 98 | 1.45 | 98 | 0.01 | 98 | $< \epsilon$ | 98 | 0.69 |
| pmed03 | 100 | 10 | 93 | 93 | 0.10 | 93 | 1.81 | 93 | 1.35 | 93 | 0.06 | 93 | 0.01 | 93 | 0.43 |
| pmed04 | 100 | 20 | 74 | 74 | 0.10 | 74 | 1.01 | 74 | 0.92 | 74 | $< \epsilon$ | 74 | $< \epsilon$ | 74 | 0.3 |
| pmed05 | 100 | 33 | 48 | 48 | 0.10 | 48 | 1.49 | 48 | 0.73 | 48 | $< \epsilon$ | 48 | $< \epsilon$ | 48 | 0.11 |
| pmed06 | 200 | 5 | 84 | 84 | 0.30 | 84 | 13.53 | 84 | 9.01 | 84 | 0.02 | 84 | $< \epsilon$ | 84 | 1.05 |
| pmed07 | 200 | 10 | 64 | 64 | 0.50 | 64 | 5.09 | 64 | 4.31 | 64 | 0.01 | 64 | $< \epsilon$ | 64 | 0.47 |
| pmed08 | 200 | 20 | 55 | 55 | 0.40 | 55 | 5.31 | 55 | 3.34 | 55 | 0.01 | 55 | $< \epsilon$ | 55 | 0.29 |
| pmed09 | 200 | 40 | 37 | 37 | 0.10 | 37 | 3.46 | 37 | 2.66 | 37 | $< \epsilon$ | 37 | $< \epsilon$ | 37 | 0.45 |
| pmed10 | 200 | 67 | 20 | 20 | 0.30 | 20 | 2.76 | 20 | 2.57 | 20 | $< \epsilon$ | 20 | $< \epsilon$ | 20 | 0.53 |
| pmed11 | 300 | 5 | 59 | 59 | 1.00 | 59 | 11.67 | 59 | 16.25 | 59 | 0.04 | 59 | 0.01 | 59 | 2.33 |
| pmed12 | 300 | 10 | 51 | 51 | 1.30 | 51 | 12.03 | 51 | 12.25 | 51 | 0.01 | 51 | $< \epsilon$ | 51 | 1.26 |
| pmed13 | 300 | 30 | 36 | 36 | 0.80 | 36 | 14.43 | 36 | 8.23 | 36 | 0.05 | 36 | $< \epsilon$ | 36 | 0.79 |
| pmed14 | 300 | 60 | 26 | 26 | 0.90 | 26 | 6.61 | 26 | 6.81 | 26 | 0.01 | 26 | $< \epsilon$ | 26 | 0.75 |
| pmed15 | 300 | 100 | 18 | 18 | 1.00 | 18 | 4.43 | 18 | 4.40 | 18 | $< \epsilon$ | 18 | $< \epsilon$ | 18 | 0.49 |
| pmed16 | 400 | 5 | 47 | 47 | 1.60 | 47 | 30.01 | 47 | 28.10 | 47 | 0.01 | 47 | $< \epsilon$ | 47 | 5.06 |
| pmed17 | 400 | 10 | 39 | 39 | 2.10 | 39 | 30.88 | 39 | 27.06 | 39 | 0.02 | 39 | $< \epsilon$ | 39 | 2.51 |
| pmed18 | 400 | 40 | 28 | 28 | 1.40 | 28 | 12.49 | 28 | 13.17 | 28 | 0.13 | 28 | 0.01 | 28 | 2.67 |
| pmed19 | 400 | 80 | 18 | 18 | 0.40 | 18 | 9.89 | 18 | 10.16 | 18 | 1.08 | 18 | 0.57 | 18 | 6.34 |
| pmed20 | 400 | 133 | 13 | 13 | 1.80 | 13 | 13.53 | 13 | 9.30 | 13 | 0.10 | 13 | 0.03 | 13 | 0.63 |
| pmed21 | 500 | 5 | 40 | 40 | 5.20 | 40 | 56.40 | 40 | 56.01 | 40 | 0.01 | 40 | 0.01 | 40 | 5.48 |
| pmed22 | 500 | 10 | 38 | 38 | 4.30 | 38 | 495.20 | 38 | 60.78 | 38 | 1.12 | 38 | 0.24 | 38 | 24.79 |
| pmed23 | 500 | 50 | 22 | 22 | 1.20 | 22 | 28.52 | 22 | 16.45 | 22 | 2.11 | 22 | 0.14 | 22 | 15.45 |
| pmed24 | 500 | 100 | 15 | 15 | 4.50 | 15 | 14.64 | 15 | 12.59 | 15 | 0.06 | 15 | 0.04 | 15 | 1.39 |
| pmed25 | 500 | 167 | 11 | 11 | 2.70 | 11 | 13.06 | 11 | 10.28 | 11 | 0.05 | 11 | 0.02 | 11 | 0.46 |
| pmed26 | 600 | 5 | 38 | 38 | 6.10 | 38 | 401.60 | 38 | 104.20 | 38 | 0.03 | 38 | $< \epsilon$ | 38 | 3.87 |
| pmed27 | 600 | 10 | 32 | 32 | 8.20 | 32 | 78.24 | 32 | 65.23 | 32 | 0.04 | 32 | $< \epsilon$ | 32 | 4.27 |
| pmed28 | 600 | 60 | 18 | 18 | 2.10 | 18 | 39.51 | 18 | 19.31 | 18 | 0.13 | 18 | 0.04 | 18 | 1.56 |
| pmed29 | 600 | 120 | 13 | 13 | 5.10 | 13 | 32.00 | 13 | 23.61 | 13 | 0.05 | 13 | 0.03 | 13 | 1.42 |
| pmed30 | 600 | 200 | 9 | 9 | 5.40 | 9 | 34.72 | 9 | 17.22 | 9 | 0.80 | 9 | 0.35 | 9 | 0.78 |
| pmed31 | 700 | 5 | 30 | 30 | 8.10 | 30 | 303.00 | 30 | 122.60 | 30 | 0.03 | 30 | $< \epsilon$ | 30 | 6.47 |
| pmed32 | 700 | 10 | 29 | 29 | 45.20 | 29 | 447.00 | 29 | 116.80 | 29 | 0.31 | 29 | 0.05 | 29 | 8.05 |
| pmed33 | 700 | 70 | 15 | 15 | 3.10 | 15 | 94.07 | 15 | 33.11 | 15 | 81.75 | 15 | 1.56 | 15 | 51.8 |
| pmed34 | 700 | 140 | 11 | 11 | 6.50 | 11 | 50.23 | 11 | 29.66 | 11 | 0.04 | 11 | 0.04 | 11 | 2.83 |
| pmed35 | 800 | 5 | 30 | 30 | 13.70 | 30 | 183.90 | 30 | 123.30 | 30 | 0.10 | 30 | 0.04 | 30 | 5.35 |
| pmed36 | 800 | 10 | 27 | 27 | 34.50 | 27 | 3602.00 | 27 | 110.50 | 27 | 0.96 | 27 | 0.55 | 27 | 39.81 |
| pmed37 | 800 | 80 | 15 | 15 | 2.00 | 15 | 105.80 | 15 | 49.02 | 15 | 0.27 | 15 | 0.08 | 15 | 10.53 |
| pmed38 | 900 | 5 | 29 | 29 | 18.50 | 29 | 251.00 | 29 | 273.10 | 29 | 0.03 | 29 | 0.03 | 29 | 9.78 |
| pmed39 | 900 | 10 | 23 | 23 | 27.30 | 23 | 5817.00 | 23 | 208.70 | 23 | 26.30 | 23 | 0.77 | 23 | 203.22 |
| pmed40 | 900 | 90 | 13 | 13 | 7.80 | 13 | 240.80 | 13 | 462.90 | 13 | 0.46 | 13 | 0.25 | 13 | 28.01 |
| Num. | | | | 40 | | 40 | | 40 | | 40 | | 40 | | 40 | |
| Avg. | | | | | 5.67 | | 336.87 | | 51.99 | | 2.91 | | 0.12 | | 11.32 |

**TABLE 5. Comparison with the reference algorithms on the set of instances from u1060.**

| Instance | $n$ | $p$ | $f_{opt}$ | ELP | | PBS | | GRASP/PR | | SAT Solvers | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | $f_{best}$ | cpu | $f_{best}$ | cpu | $f_{best}$ | cpu | $f_{best}$ | cpu |
| u1060 | 1060 | 10 | 2273.08 | 2273 | 53 | 2273.08 | 138.11 | 2273.08 | 1.31 | 2273.08 | 1.29 |
| u1060 | 1060 | 20 | 1580.80 | 1581 | 2778 | 1580.80 | 659.44 | 1580.80 | 14.88 | 1580.80 | 8.03 |
| u1060 | 1060 | 30 | 1207.77 | 1208 | 298 | 1207.77 | 36.84 | 1207.77 | 3.19 | 1207.77 | 6.43 |
| u1060 | 1060 | 40 | 1020.56 | 1021 | 366 | 1020.56 | 47.73 | 1020.56 | 3.26 | 1020.56 | 9.17 |
| u1060 | 1060 | 50 | 904.92 | 905 | 383 | 904.92 | 233.13 | 904.92 | 218.85 | 904.92 | 14.91 |
| u1060 | 1060 | 60 | 781.17 | 781 | 233 | 781.17 | 103.12 | 781.17 | 7.75 | 781.17 | 9.8 |
| u1060 | 1060 | 70 | 710.75 | 711 | 135 | 710.76 | 109.56 | 710.75 | 116.91 | 710.75 | 8.46 |
| u1060 | 1060 | 80 | 652.16 | 652 | 60 | 652.16 | 142.11 | 652.16 | 316.57 | 652.16 | 14.41 |
| u1060 | 1060 | 90 | 607.87 | 608 | 38 | 607.88 | 63.15 | 607.87 | 7.09 | 607.87 | 10.5 |
| u1060 | 1060 | 100 | 570.01 | 570 | 29 | 570.01 | 17.54 | 570.01 | 19.04 | 570.01 | 16.28 |
| u1060 | 1060 | 110 | 538.84 | 539 | 30 | 538.84 | 160.73 | 538.84 | 66.46 | 538.84 | 24.54 |
| u1060 | 1060 | 120 | 510.27 | 510 | 44 | 510.28 | 107.65 | 510.27 | 397.85 | 510.27 | 13.97 |
| u1060 | 1060 | 130 | 499.65 | 500 | 44 | 499.65 | 118.71 | 499.65 | 58.18 | 499.65 | 11.53 |
| u1060 | 1060 | 140 | 452.46 | 452 | 46 | 452.46 | 318.48 | 452.46 | 127.39 | 452.46 | 4.03 |
| u1060 | 1060 | 150 | 447.01 | 447 | 50 | 447.01 | 10.59 | 447.01 | 4.37 | 447.01 | 4.11 |
| Num. | | | | 15 | | 15 | | 15 | | 15 | |
| Avg. | | | | | 305.8 | | 151.13 | | 90.87 | | 10.50 |

**TABLE 6. Comparison with the reference algorithms on the set of instances from u1817.**

| Instance | $n$ | $p$ | $f_{opt}$ | ELP | | PBS | | GRASP/PR | | SAT Solvers | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | $f_{best}$ | cpu | $f_{best}$ | cpu | $f_{best}$ | cpu | $f_{best}$ | cpu |
| u1817 | 1817 | 10 | 457.91 | 458 | 2700 | 457.91 | 5316.60 | 457.91 | 604.53 | 457.91 | 42.61 |
| u1817 | 1817 | 20 | 309.01 | 310? | 4920 | 309.01 | 10243.00 | 309.01 | 4068.06 | 309.01 | 80.93 |
| u1817 | 1817 | 30 | 240.99 | 250? | 1650 | 240.99 | 1605.50 | 240.99 | 1239.97 | 240.99 | 123.61 |
| u1817 | 1817 | 40 | 209.45 | 210? | 6420 | 209.46 | 193.70 | 209.45 | 308.29 | 209.45 | 47.09 |
| u1817 | 1817 | 50 | 184.91 | 187? | 9840 | 184.91 | 1128.90 | 184.91 | 471.94 | 184.91 | 173.81 |
| u1817 | 1817 | 60 | 162.64 | 163 | 1260 | 162.65 | 837.30 | 162.64 | 469.43 | 162.64 | 134.17 |
| u1817 | 1817 | 70 | 148.11 | 148 | 420 | 148.11 | 191.80 | 148.11 | 19.66 | 148.11 | 46.17 |
| u1817 | 1817 | 80 | 136.77 | 137 | 1140 | 136.80 | 127.50 | 136.80 | 12.42 | **136.77** | 2313.89 |
| u1817 | 1817 | 90 | 129.51 | 130? | 7202 | 129.54 | 2963.50 | 129.51 | 3859.05 | 129.51 | 370.95 |
| u1817 | 1817 | 100 | 126.99 | 127 | 300 | 127.01 | 146.40 | 126.99 | 2.35 | 126.99 | 124.24 |
| u1817 | 1817 | 110 | 109.25 | 109 | 420 | 109.25 | 13772.40 | 109.25 | 6954.89 | 109.25 | 84.25 |
| u1817 | 1817 | 120 | 107.76 | 108 | 120 | 107.78 | 80.10 | 107.76 | 5.25 | 107.76 | 54.48 |
| u1817 | 1817 | 130 | 104.73 | 108? | 3720 | 107.75 | 11.20 | 107.75 | 7.04 | **104.73** | 941.17 |
| u1817 | 1817 | 140 | 101.60 | 105? | 4020 | 101.61 | 4949.30 | 101.60 | 30.95 | 101.60 | 385.56 |
| u1817 | 1817 | 150 | 91.60 | 94? | 5640 | 101.60 | 314.00 | 92.44 | 1236.55 | **91.60** | 227.99 |
| Num. | | | | 7 | | 6 | | 12 | | **15** | |
| Avg. | | | | | 3318.13 | | 2792.08 | | 1286.03 | | 343.39 |

other reference algorithms in terms of both solution quality and run time. Rows Num. and Avg. present the number of instances for which the optimal or the best known results can be obtained and the average computational time to reach the best known results for the corresponding solvers, respectively.

From Table 7, one observes that for the random instances in OR-Lib Sparrow2Riss-2018, which is from the random track and is a specialized solver for random instances, can hit the lower bounds for all the instances in OR-Lib, while other exact SAT solvers hit the lower bounds for only 31 instances.

**TABLE 7.** The statistics of all solvers on two datasets OR-Lib and TSP-Lib.

| | OR-Lib | | | TSP-Lib | | |
|---|---|---|---|---|---|---|
| | Sparrow | Exact | Ref | Sparrow | Exact | Ref |
| Num. | 40 | 31 | 40 | 30 | 30 | 27 |
| Avg. | 11.32 | 513.07 | 0.12 | 824.15 | 187.03 | 688.44 |

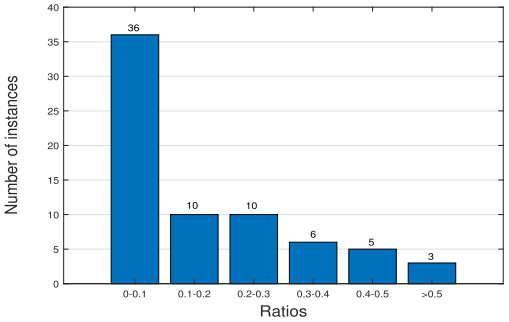


**FIGURE 5.** The distribution of the ratios of $k$ (the number of determined centers) to $p$ for the 70 instances.

For the real world application instances in TSP-Lib, all the SAT solvers reach the best known results for all the 30 instances where the exact SAT solvers need much less time than Sparrow2Riss-2018. This phenomenon shows that the exact SAT solvers are more suitable for solving the real world application instances while it is relatively easy for the random solver Sparrow2Riss 2018 for solving the random instances. In addition, although the reference algorithms in the literature can obtain equal or better results with less computational time than the SAT solvers on the random instances, the exact SAT solvers can obtain better results with less computational time than the reference algorithms on the real world application instances, which is of practical significance.

## V. ANALYSIS AND DISCUSSION

### A. IMPORTANCE OF DATA REDUCTION

In order to evaluate the effectiveness of the data reduction rules to the $p$-center problem approach via SAT solvers, we conduct experiments to compare the two versions of the algorithm with and without the data reduction rules. Note that the data reduction can determine some centers. Let $k$ be the number of centers determined by the data reduction. Figure 5 shows that the ratio $k/p$ is less than 0.2 for most instances, which means the impact of data reduction is small for these instances. But for some instances, we can improve the computational efficiency dramatically. Table 8 presents the computation results of the two versions of the algorithm in the SEQ mode for some instances when using the Maple_CM solver, where column $t_1(s)$ and $t_2(s)$ respectively show the average computational time of the listed instances with and without the data reduction rules, and Ratio shows the ratio of the number of centers determined by data reduction to $p$. From Table 8, one observes that pmed29 can determine 30% centers by data reduction and the time for solving pmed29 by Maple_CM with and without data reduction is 19.09s and 3764.20s when the given radius is 13 and the encoding mode is SEQ. Furthermore, data reduction can improve the solution quality for 4 instances: pemd20, pmed24, pmed30 and pmed34, which shows the importance of the data reduction rule.

**TABLE 8.** The computational results of two algorithms with and without data reduction in the SEQ mode when using Maple_CM.

| Instance | $n$ | $p$ | $radius$ | $t_1(s)$ | $t_2(s)$ | Ratio |
|---|---|---|---|---|---|---|
| pmed08 | 200 | 20 | 55 | 0.28 | 56.26 | 20.00% |
| pmed09 | 200 | 40 | 37 | 0.06 | 0.34 | 37.50% |
| pmed10 | 200 | 67 | 20 | 0.05 | 0.33 | 49.00% |
| pmed13 | 300 | 30 | 36 | 3820.47 | 10150.89 | 30.00% |
| pmed14 | 300 | 60 | 26 | 0.64 | 2283.09 | 31.66% |
| pmed15 | 300 | 100 | 18 | 0.06 | 1.28 | 50.00% |
| pmed20 | 400 | 133 | 13 | 0.18 | - | 51.10% |
| pmed24 | 500 | 100 | 15 | 3933.48 | - | 37.00% |
| pmed25 | 500 | 167 | 11 | 0.44 | 2895.74 | 43.10% |
| pmed29 | 600 | 120 | 13 | 19.09 | 3764.20 | 30.00% |
| pmed30 | 600 | 200 | 9 | 1.22 | - | 51.10% |
| pmed34 | 700 | 140 | 11 | 4018.62 | - | 24.20% |
| u1060 | 1060 | 70 | 710.75 | 8.46 | 19.52 | 22.80% |
| u1060 | 1060 | 90 | 607.87 | 10.50 | 22.86 | 31.10% |
| u1060 | 1060 | 130 | 499.65 | 11.53 | 27.89 | 34.60% |
| u1060 | 1060 | 140 | 452.46 | 4.03 | 22.05 | 48.57% |
| u1060 | 1060 | 150 | 447.01 | 4.11 | 20.68 | 48.00% |
| u1817 | 1817 | 80 | 136.77 | 1412.67 | 2327.21 | 3.75% |
| u1817 | 1817 | 120 | 107.76 | 79.46 | 157.96 | 16.66% |
| u1817 | 1817 | 130 | 104.73 | 1117.65 | 2129.85 | 8.46% |
| u1817 | 1817 | 150 | 91.60 | 269.70 | 449.86 | 4.20% |

### B. THE PROOF OF OPTIMALITY

The $p$-center problem is transformed to a series of feasibility set covering subproblems with different covering radius which are solved by SAT solvers. Note that if an exact SAT solver tackles an unsatisfiable instance, the solver can prove the unsatisfiability of the instance if it outputs 'UNSAT'. Thus, it is guaranteed that there are no feasible solutions for a feasibility set covering subproblem if the output of the SAT solver is 'UNSAT'. Furthermore, we can prove that $d_{i-1}$ is the optimal solution of the $p$-center problem if the output of SAT solver is 'SAT' for $d_{i-1}$ and 'UNSAT' for $d_i$, because this means that there does not exist a feasible solution for the set covering subproblems with the radii which are shorter than $d_{i-1}$, i.e., there does not exist a feasible solution $X$ with $f(X) < d_{i-1}$ for the $p$-center problem..

Although Sparrow2Riss-2018 can get the optimal solutions for all the 70 instances, it needs to apply the Gaussian Elimination and Cardinality Constraint reasoning which cause the solver to be unable to emit proof for unsatisfiable instances. So, we have to use Maple_CM or MapleLCMDistChronnoBT, i.e., the exact SAT solvers to prove the optimality for these 70 instances. We can prove the optimality for 29 out of 40 random instances in OR-Lib, and all the 30 real application instances in TSP-Lib, thus closing these instances (the optimal solution of the 30 instances are listed by $f_{opt}$ in Table 2 and Table 3). Although the previous exact method ELP claimed to prove the optimality for 22 of the 30 instances in TSP-Lib, it can only give integer lower bound for the $p$-center problem. That is to say, the exact solution obtained by ELP is the rounded integer value of the real exact solution. To our best knowledge, our proposed SAT-based method proves the optimality for all the 30 TSP-Lib instances by giving the real exact solutions for the first time, showing its high performance for real world application instances.

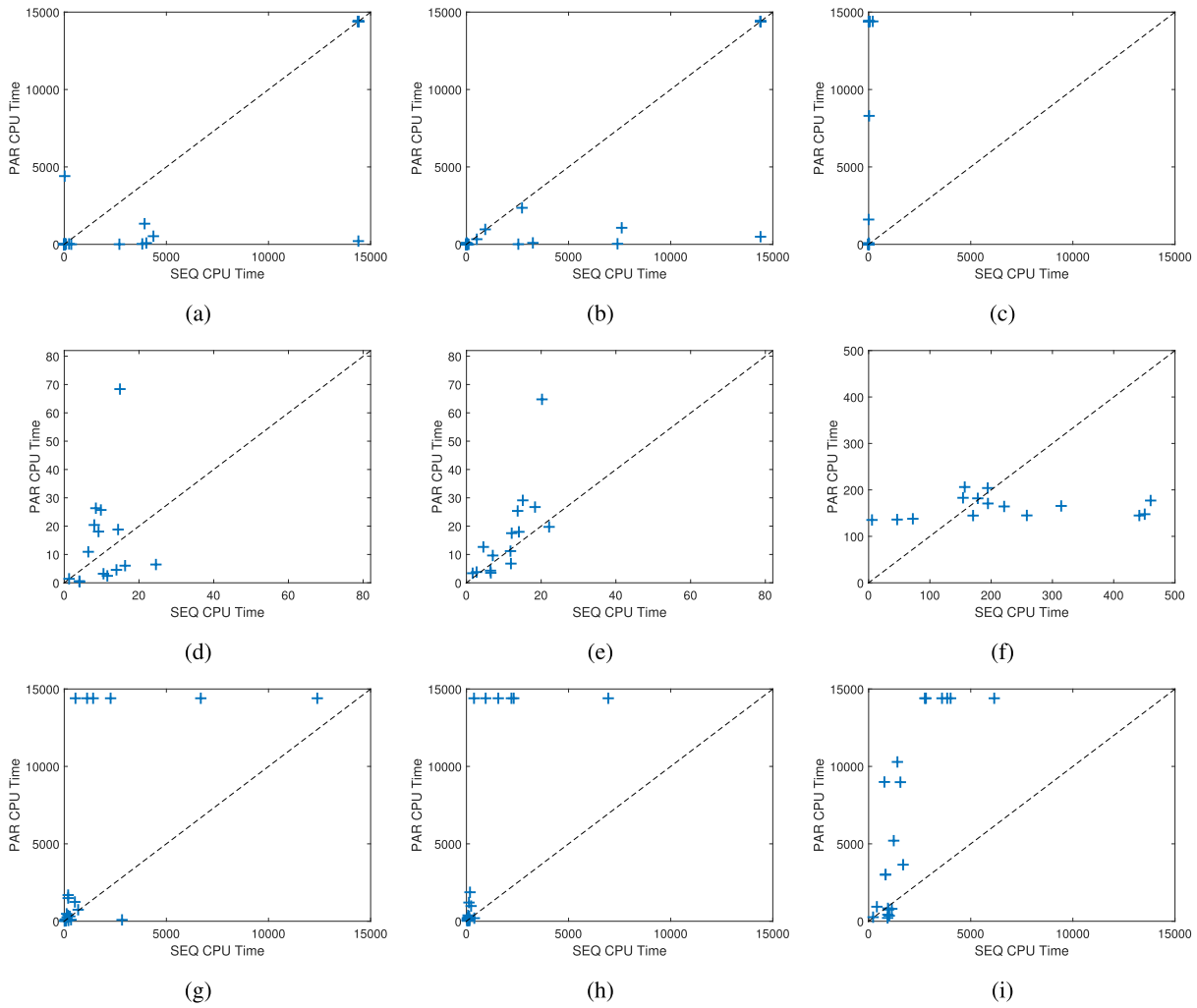


**FIGURE 6.** Scatter plots comparing SEQ runtime (x-axis) and PAR runtime (y-axis) of the three solvers on the three sets of instances (The three rows from top to bottom in sequence are pmed, u1060 and u1817. The three columns from left to right are Maple_CM, MapleLCMDistChronnoBT and Sparrow2Riss-2018).

### C. THE COMPARISONS OF ENCODING MODES

Figure 6 shows the scatter plots comparing SEQ and PAR encoding modes for the three SAT solvers on the sets of pmed, u1060 and u1817 instances. A point $(x, y)$ in the plots corresponds to an instance, where $x(y)$ represents the solving time in seconds of SEQ (PAR). A point $(x, y)$ where $x = 14400$ ($y = 14400$) means that the instance was not solved within the time limit.

Although all the 70 instances have a much smaller number of clauses and variables when using the PAR encoding mode, which is very efficient in terms of used memory, the solver's performance may be worse due to the lack of propagation strength in the PAR encoding mode. From Figure 6a - Figure 6c, one observes that the PAR encoding mode outperforms SEQ on the pmed instances for most instances when using Maple_CM and MapleLCMDistChronnoBT, while it is opposite for Sparrow2Riss-2018. Figure 6d - Figure 6f show that the two encoding modes have similar performance in terms of computational efficiency on the set of u1060 instances for all the three solvers. Figure 6g - Figure 6i show that the SEQ encoding mode outperforms PAR in terms of computational efficiency on the set of u1817 instances for all the three solvers.

## VI. CONCLUSION

In this paper, we present a new method for solving the $p$-center problem via set covering and SAT, which is the first SAT-based method for solving the $p$-center problem to our knowledge. Our proposed approach has the advantage of allowing the problem to be solved in parallel due to the independence of the set covering subproblems. The method consists of transforming the problem to a finite series of feasibility set covering subproblems, the simplification of these feasibility set covering subproblems, encoding them into CNF format with different encoding modes and then solving the encoded subproblems using state-of-the-art SAT solvers. The encoding modes that we use include sequential counter and parallel counter, and the SAT solvers consist of Maple_CM, MapleLCMDistChronoBT and Sparrow2Riss-2018.

We compare the computational results of 70 well-known benchmark instances for different encoding modes when using different SAT solvers. We find that Sparrow2 Riss-2018 is the best solver for solving these three sets of 70 instances when using the sequential counter encoding mode, for it can obtain the best solutions for all the instances. However, it also has the worst time performance when using the parallel counter mode, which indicates the importance of choosing an appropriate encoding mode.

This work demonstrates that SAT-based approaches provide a competitive alternative for solving the *p*-center problem. In fact, our SAT based approach improves the previous best known results for 3 instances and equals the best known ones for the remaining instances, meaning that our SAT-based approach is better than the previous approaches in terms of the best solution quality. Considering the notorious NP-hardness of the *p*-center problem, we believe this is significant.

In addition, this work establishes, for the first time to our best knowledge, a bridge between the SAT and *p*-center problems, opening promising perspectives. In fact, the results presented in the paper are obtained by only using general-purpose SAT solvers. On the one hand, SAT solving is a very fast developing field and our results make it clear that any progress in that field can be beneficial to solve the *p*-center problem. On the other hand, SAT solvers could be specialized for the *p*-center problem, by taking its particular features into account, which would greatly improve the performance of SAT solvers for the *p*-center problem.

It is noteworthy that our proposed SAT-based method proves the optimality for all the 30 TSP-Lib instances by giving the real exact solutions for the first time and thus closes these instances, showing its high performance for real world application instances. Combined with the proof by the the exact algorithms (ELP and IP*) in the literature on the optimality of solutions for the instances from OR-Lib, we can conclude that our proposed algorithm can obtain the optimal solutions for all the 70 instances. The study in this work inspires us that similar method can be applied for solving other challenging combinatorial optimization problems.

## REFERENCES

[1] S. L. Hakimi, "Optimum locations of switching centers and the absolute centers and medians of a graph," *Oper. Res.*, vol. 12, no. 3, pp. 450–459, Jun. 1964.

[2] S. L. Hakimi, "Optimum distribution of switching centers in a communication network and some related graph theoretic problems," *Oper. Res.*, vol. 13, no. 3, pp. 462–475, Jun. 1965.

[3] O. Kariv and S. L. Hakimi, "An algorithmic approach to network location Problems. I: The P-centers," *SIAM J. Appl. Math.*, vol. 37, no. 3, pp. 513–538, Dec. 1979.

[4] E. Minieka, "The m-center problem," *SIAM Rev.*, vol. 12, no. 1, pp. 138–139, 1970.

[5] R. S. Garfinkel, A. W. Neebe, and M. R. Rao, "Them-center problem: Minimax facility location," *Manage. Sci.*, vol. 23, no. 10, pp. 1133–1142, Jun. 1977.

[6] S. Elloumi, M. Labbé, and Y. Pochet, "A new formulation and resolution method for the P-center problem," *Informs J. Comput.*, vol. 16, no. 1, pp. 84–94, Feb. 2004.

[7] M. Daskin, "Network and discrete location: Models, algorithms and applications," *J. Oper. Res. Soc.*, vol. 48, no. 7, pp. 763–764, 1995.

[8] H. Calik and B. C. Tansel, "Double bound method for solving the P-center location problem," *Comput. Oper. Res.*, vol. 40, no. 12, pp. 2991–2999, Dec. 2013.

[9] M. S. Daskin, "A new approach to solving the vertex P-center problem to optimality: Algorithm and computational results," *Commun. Oper. Res. Soc. Jpn.*, vol. 45, no. 9, pp. 428–436, 2000.

[10] T. Ilhan and M. C. Pinar, "An efficient exact algorithm for the vertex p-center problem," Dept. Ind. Eng., Bilkent Univ., Ankara, Turkey, Tech. Rep., 2001.

[11] A. Al-khedhairi and S. Salhi, "Enhancements to two exact algorithms for solving the vertex P-center problem," *J. Math. Model. Algorithms*, vol. 4, no. 2, pp. 129–147, Jun. 2005.

[12] N. Mladenović, M. Labbé, and P. Hansen, "Solving the P-center problem with tabu search and variable neighborhood search," *Networks*, vol. 42, no. 1, pp. 48–64, Aug. 2003.

[13] W. Pullan, "A memetic genetic algorithm for the vertex P-center problem," *Evol. Comput.*, vol. 16, no. 3, pp. 417–436, Sep. 2008.

[14] A.-H. Yin, T.-Q. Zhou, J.-W. Ding, Q.-J. Zhao, and Z.-P. Lv, "Greedy randomized adaptive search procedure with path-relinking for the vertex P-center problem," *J. Comput. Sci. Technol.*, vol. 32, no. 6, pp. 1319–1334, Nov. 2017.

[15] S. A. Cook, "The complexity of theorem-proving procedures," in *Proc. 3rd Annu. ACM Symp. Theory Comput.*, 1971, pp. 151–158.

[16] A. Van Gelder, "Another look at graph coloring via propositional satisfiability," *Discrete Appl. Math.*, vol. 156, no. 2, pp. 230–243, Jan. 2008.

[17] T. Soh, K. Inoue, N. Tamura, M. Banbara, and H. Nabeshima, "A SAT-based method for solving the two-dimensional strip packing problem," *Fundamenta Informaticae*, vol. 102, nos. 3–4, pp. 467–487, 2010.

[18] J. Alber, M. R. Fellows, and R. Niedermeier, "Polynomial-time data reduction for dominating set," *J. ACM*, vol. 51, no. 3, pp. 363–384, May 2004.

[19] Z. Lü and J.-K. Hao, "A memetic algorithm for graph coloring," *Eur. J. Oper. Res.*, vol. 203, no. 1, pp. 241–250, May 2010.

[20] X. Wu, Z. Lü, and P. Galinier, "Restricted swap-based neighborhood search for the minimum connected dominating set problem," *Networks*, vol. 69, no. 2, pp. 222–236, Mar. 2017.

[21] Z. Xing and Z. Shi, "A fast and deterministic algorithm for consensus set maximization," *IEEE Access*, vol. 6, pp. 26175–26180, 2018.

[22] C. Sinz, "Towards an optimal CNF encoding of Boolean cardinality constraints," in *Proc. Int. Conf. Princ. Pract. Constraint Program.*, 2005, pp. 827–831.

[23] J. E. Beasley, "OR-library: Distributing test problems by electronic mail," *J. Oper. Res. Soc.*, vol. 41, no. 11, pp. 1069–1072, Nov. 1990.

[24] G. Reinelt, "TSPLIB—A traveling salesman problem library," *ORSA J. Comput.*, vol. 3, no. 4, pp. 376–384, Nov. 1991.

[25] C.-M. Li, F. Xiao, M. Luo, F. Manyà, Z. Lü, and Y. Li, "Clause vivification by unit propagation in CDCL SAT solvers," *Artif. Intell.*, vol. 279, Feb. 2020, Art. no. 103197.

[26] V. Ryvchin and A. Nadel, "Maple_LCM_Dist_ChronoBT: Featuring chronological backtracking," in *Proc. SAT Competition*, 2018, p. 29.

[27] A. Balint and N. Manthey, "Sparrow + CP3 and SparrowToRiss," in *Proc. SAT Competition*, 2013, pp. 87–88.

[28] M. J. H. Heule, M. J. Järvisalo, and M. Suda, "SPARROWTORISS 2018," in *Proc. SAT Competition*, 2018, pp. 38–39.

[29] M. Luo, C.-M. Li, F. Xiao, F. Manyá, and Z. Lü, "An effective learnt clause minimization approach for CDCL SAT solvers," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 703–711.

[30] J. H. Liang, V. Ganesh, P. Poupart, and K. Czarnecki, "Learning rate based branching heuristic for SAT solvers," in *Proc. Int. Conf. Theory Appl. Satisfiability Test.*, 2016, pp. 123–140.

[31] J. H. Liang, C. Oh, V. Ganesh, K. Czarnecki, and P. Poupart, "MapleCOMSPS, MapleCOMSPS LRB, MapleCOMSPS CHB," in *Proc. SAT Competition*, 2016, p. 52.

[32] A. Nadel and V. Ryvchin, "Chronological backtracking," in *Theory and Applications of Satisfiability Testing—SAT*. Springer, 2018, pp. 111–121.

[33] A. Balint and A. Fröhlich, "Improving stochastic local search for SAT with a new probability distribution," in *Proc. Int. Conf. Theory Appl. Satisfiability Test.*, 2010, pp. 10–15.

[34] N. Eén and N. Sörensson, "An extensible SAT-solver," in *Proc. Int. Conf. Theory Appl. Satisfiability Test.*, 2003, pp. 502–518.

[35] G. Audemard and L. Simon, "Predicting learnt clauses quality in modern SAT solvers," in *Proc. 21st Int. Joint Conf. Artif. Intell.*, 2009, pp. 399–404.

[36] N. Manthey, "Coprocessor 2.0—A flexible CNF simplifier," in *Proc. Int. Conf. Theory Appl. Satisfiability Test.*, 2012, pp. 436–441.

[37] Q. Wu and J.-K. Hao, "An adaptive multistart tabu search approach to solve the maximum clique problem," *J. Combinat. Optim.*, vol. 26, no. 1, pp. 86–108, Jul. 2013.

**XIAOLU LIU** received the B.E. degree in system engineering and the Ph.D. degree in management science and engineering from the National University of Defense Technology, China, in 2006 and 2011, respectively. She is currently an Associate Professor with the College of System Engineering, National University of Defense Technology. Her research interests include artificial intelligence and metaheuristics, solving distribution management, and satellite scheduling problems.

**YUAN FANG** received the B.S. degree in computer science and technology from the Huazhong University of Science and Technology, China, in 2018, where he is currently pursuing the B.E. degree in computer software and theory with the Laboratory of Smart Computing and Optimization, School of Computer Science and Technology. His research interests include heuristic algorithm and program optimization.

**JIAMING CHEN** received the B.E. degree in system engineering from the National University of Defense Technology, China, in 2018, where he is currently pursuing the master's degree in management science and engineering. His research interests include computation intelligence, evolutionary computation, and adaptive meta-heuristic or hyperheuristic for solving large-scale combinatorial optimization and resource scheduling problems.

**ZHOUXING SU** received the B.E. degree in computer science and technology from the Huazhong University of Science and Technology, China, in 2014, where he is currently pursuing the Ph.D. degree in computer software and theory with the Laboratory of Smart Computing and Optimization, School of Computer Science and Technology. His research interests include metaheuristics and mathematical programming to solve real-world applications, such as personnel rostering, inventory routing, and facility location problem.

**CHUMIN LI** received the B.S. and Ph.D. degrees in computer science from the University of Technology of Compiegne, France, in 1985 and 1990, respectively. He is currently a Professor of computer science with the University of Picardie Jules Verne. His research interests include the practical resolution of NP-hard problems, including SAT, CSP, MaxSAT, MinSAT, MaxClique, and GCP. He is particularly interested in the intrinsical relationships between these problems. One of his research directions is to find and exploit these relationships to solve them. A recent example is the exploitation of the relationships between MaxSAT and MaxClique to solve MaxClique.

**ZHIPENG LÜ** received the B.S. degree in applied mathematics from Jilin University, China, in 2001, and the Ph.D. degree in computer software and theory from the Huazhong University of Science and Technology, China, in 2007. He was a Postdoctoral Research Fellow with the LERIA, Department of Computer Science, University of Angers, France, from 2007 to 2011. He is currently a Professor with the School of Computer Science and Technology, Huazhong University of Science and Technology, and the Director of the Laboratory of Smart Computing and Optimization. His research interests include artificial intelligence, computational intelligence, operations research and adaptive metaheuristics for solving largescale real-world and theoretical combinatorial optimization, and constrained satisfaction problems.