24th European Conference on Artificial Intelligence - ECAI 2020
Santiago de Compostela, Spain

Branch Location Problems with Maximum Satisfiability

Oleg Zaikin[1] and **Alexey Ignatiev**[2] and **Joao Marques-Silva**[3]

Abstract. Constrained location problems find a wide range of practical applications. Recent work showed that dedicated brute-force algorithms and greedy approach enable solutions of reasonable efficiency, for a restriction of the general constrained location problem, referred to as the *branch location problem*. This paper extends earlier work in several ways. First, the paper develops propositional encodings for the branch location problem. Second, given that the branch location problem is a restriction of the general constraint location problem, the paper shows that the restricted problem is still hard for NP. Third, the paper devises improved propositional encodings for the branch location problem, which in practice enable not only solving *exactly* a significantly larger class of problems but also effectively *approximating* optimal problem solutions, using state-of-the-art (complete and incomplete) Maximum Satisfiability (MaxSAT) solvers.

1 INTRODUCTION

Facility location problems are representative of the field of *location science* [17, 29]. This field is characterized by a wealth of different optimization problems, finding a wide range of practical applications [32, 23, 36, 28, 44, 42, 33, 20, 16, 2, 18, 19]. Despite the practical relevancy of location problem, existing solutions most often do not resort to the use of constraint programming approaches (including SAT and MaxSAT), albeit a few exceptions have been documented [32, 9, 39, 21].

Among a wide range of variants of facility location problems, one concrete example is the *branch location problem*. The problem formulation considers a number of branches (or facilities) and a number of customers (or customers). The physical distance between each customer and (some of the branches) is represented by a weighted edge. Given a target number of branches to eliminate, the goal of the branch location problem is to decide which branches to eliminate such that the number of dissatisfied customers is minimized. Customers are declared to be dissatisfied if the increase in penalty exceeds a given target threshold Δ.

Example 1 *Throughout the paper, we consider one small example, consisting of three branches* $\{f_1, f_2, f_3\}$ *and six customers* $\{c_1, c_2, c_3, c_4, c_5, c_6\}$. *The purpose of this concrete example is to close one of the branch locations. For each customer, there is a penalty associated with each branch, e.g. it can measure the distance required for the user to visit the branch. For our concrete example, the matrix of penalties is shown in Figure 1. Moreover, we assume* $\Delta = 300$.

	f_1	f_2	f_3
c_1	250	600	700
c_2	1200	700	950
c_3	950	1100	450
c_4	400	950	800
c_5	1150	800	350
c_6	1250	600	1050

(a) Matrix of penalties

	f_1	f_2	f_3
c_1	+1	0	0
c_2	0	0	0
c_3	0	0	+1
c_4	+1	0	0
c_5	0	0	+1
c_6	0	+1	0

(b) Dissatisfied customers ($\Delta = 300$)

Figure 1: Branch location example

As one can observe, for customer c_1, *if branch* f_1 *is to be closed, then customer* c_1 *will increase its penalty from 250 to 600 or 700. Thus, the increase in penalty always exceeds the threshold of 300, and so if branch* f_1 *is removed, the customer* c_1 *will be dissatisfied.*
For this concrete example, the goal is to select one branch to close such that the number of dissatisfied customers is minimized. Figure 1b shows a matrix of dissatisfied customers. It is easy to conclude that the branch to remove should be f_2, *as the number of dissatisfied customers is minimized* [4].

There are more general formulations of the problem, which enable the addition of branch locations [45]. This paper considers the restriction of the problem where only existing branch locations can be removed. Recent work proposed dedicated algorithms for the branch location problem [46], namely a so-called brute force approach, based on exhaustive search, and an alternative greedy approach, with no guarantees of optimality.

Given that the problem formulation is mostly propositional, a natural question is whether available tools (e.g. SAT/MaxSAT or their CP or SMT counterparts) could be effective at solving this problem. This paper confirms that a basic propositional encoding does not enable general purpose solvers to outperform dedicated solutions. Analysis of the basic propositional encoding reveals a limitation of the modeling, which impacts the effectiveness of MaxSAT solvers. As a result, the paper identifies optimizations to the proposed encoding, which are inspired on fairly general insights, and which enable significant performance improvements, including outperforming the recently proposed state-of-the-art solutions [46]. In addition, since branch location is a restriction of the general problem of facility location, the paper shows that the branch location problem is hard for the class NP of decision problems.

[1] Matrosov Institute for System Dynamics and Control Theory SB RAS, Irkutsk, Russia
[2] Faculty of IT, Monash University, Melbourne, Australia
[3] ANITI – Artificial and Natural Intelligence Toulouse Institute, Univ. Toulouse, France

[4] Clearly, for the case when 1 branch is to be removed, there is a straightforward polynomial time solution, consisting of checking (and picking) which column maximizes the number of dissatisfied customers. As shown in this paper, and for an arbitrarily large number of branches to close, the problem is computationally harder, even in the restricted case of branch location problems.

The paper is organized as follows. The concepts and notation used throughout the paper are introduced in Section 2. The branch location problem, i.e. restriction of facility location problems considered in this paper, is presented in Section 3. This section also demonstrates the NP-hardness of the problem. The propositional encoding of the branch location problem is detailed in Section 4. Finally, the experimental results are analyzed in Section 5, and the paper concludes in Section 6.

2 PRELIMINARIES

Basic knowledge of graph-related problems and computational complexity is assumed, including weighted and bipartite graphs, as well as the NP-hard set cover problem [27]. (Alternatively, the reader is referred to standard bibliography [12].)

Facility Location Problems. Facility location problems have been extensively investigated in Operations Research, and a wide range of formulations exist [17, 29]. There has been some work on using constraint programming with specific formulations of facility location [9, 39, 21]. This paper considers a restriction of the more general budget-constrained location (BCL) problem [45], which we now detail[5].

We consider a network with $C = \{1,\ldots,n\}$ nodes, denoting customers, $S = \{j_1,\ldots,j_q\}$ nodes denoting existing facilities and $T = \{j_{q+1},\ldots,j_m\}$ nodes denoting possible new facilities, and $F = S \cup T$, with $|F| = m$. D is a distance matrix, relating customers and facilities, where d_{ij} denotes the distance between customer node $i \in C$ and facility node $j \in F$. Moreover, w_i denotes the demand of customer node $i \in C$, h_j denotes the cost of closing an existing facility associated with node $j \in S$ or opening a new facility associated with node $j \in T$. Finally, p is the target number of facilities and b is the total budget.

Two sets of variables are used. Variable $y_j = 1$ iff the facility $j \in F$ is open, and variable $x_{ij} = 1$ iff the demand of customer $i \in C$ is satisfied by the facility opened at $j \in F$.

The BCL problem is formulated as follows (adapted from [45]):

$$
\begin{array}{lll}
\text{min:} & \sum_{i\in C}\sum_{j\in F} w_i d_{ij} x_{ij} & \\
\text{s.t.:} & \sum_{j\in S} h_j(1-y_j) + \sum_{j\in T} h_j y_j \leq b & \\
 & \sum_{j\in F} x_{ij} = 1 & \forall i \in C \\
 & \sum_{j\in F} y_j = p & \\
 & x_{ij} \leq y_j & \forall i \in C, j \in F \\
 & y_j \in \{0,1\} & \forall j \in F \\
 & x_{ij} \geq 0 & \forall i \in C, j \in F
\end{array}
\qquad (1)
$$

The BCL problem is hard for NP [45]. This paper considers a simplified restriction of the BCL problem, which suffices to model a wide range of branch location problems, e.g. related with location of bank branches, supermarkets, etc.

Satisfiability & Maximum Satisfiability. The definitions for the Boolean Satisfiability problem and the Maximum Satisfiability problems are standard, and are briefly reviewed in this paragraph. (A more detailed account can be found elsewhere [7].) The paper considers propositional formulas represented in conjunctive normal form (CNF). CNF formulas are defined on a set X of propositional variables, and consist of conjunctions of disjunctions of literals, where a literal is either a variable x from X or its negation ($\neg x$). An *assignment* μ is a mapping from X to $\{0,1\}$. A satisfying assignment (also referred to as a *model*) of a formula φ is an assignment such that at least one literal in each clause is assigned value 1. A formula with at least one satisfying assignment is referred to as *satisfiable*, whereas a formula without satisfying assignments is referred to as *unsatisfiable*. Boolean satisfiability (SAT) is the decision problem for propositional logic, and consists of deciding whether a propositional formula is satisfiable. The SAT problem is well-known to be NP-complete, even though in practice modern SAT algorithms tend to defy the expected exponential worst-case run time.

Unsatisfiable (or overconstrained) formulas find a wide range of applications. The study of overconstrained formulas [8] is tightly related with earlier work on model-based diagnosis [40]. For an overconstrained formula, a problem of interest is to identify the largest number of clauses that can be satisfied by any given assignment. This is referred to as the maximum satisfiability (MaxSAT) problem [31]. In practical settings, MaxSAT comprises a number of variants, allowing clauses to be declared as *soft* or as *hard* (i.e. *must* be satisfied) and allowing for soft clauses to be associated with some weight (i.e. the cost of falsifying the clause). The notation (c, w) is used to denote a (soft) clause with some weight, where a hard clause is represented as $(c, \top)$, where $\top$ denotes an infinite weight. MaxSAT is well-known to be hard for NP [27] but, mimicking the success of SAT algorithms, MaxSAT algorithms have also seen significant practical improvements over the last decade.

Although propositional logic is widely accepted as fairly inexpressive, there has been extensive work on modeling decision and optimization problems with propositional logic, extending the reach of SAT and MaxSAT solvers. Encodings of complex constraints into propositional logic have also been extensively studied [5, 4, 43, 7, 3, 35].

3 PROBLEM FORMULATION

This section describes a restricted formulation of the BCL problem and analyzes its complexity. The restriction assumes that no new facilities from T can be added, and focuses solely on the removal of existing facilities from $F = S$. Concretely, assume that a set of customers, a set of facilities, and also distances between customers and facilities are given. It is required to close a given number of facilities to decrease the operational costs. A customer is called *potentially dissatisfied* if the distance from the customer to the closest facility is increased by more than a given threshold after the facilities closure. Informally, the following problem is considered: to decide which of the existing facilities should be closed so that the total number of potentially dissatisfied customers is minimized.

This problem can be naturally formulated on an undirected edge-weighted complete bipartite graph. Nodes where existing facilities are located and nodes where customers reside correspond to vertices of the first and second parts of the bipartite graph, respectively. Every customer vertex has an edge to every facility vertex, and the edge's weight is the distance between the corresponding nodes.

Let us formalize the problem. Consider an undirected edge-weighted complete bipartite graph $G(C, F, E, w)$, where

- $C = \{c_1,\ldots,c_n\}, |C| = n$, is the set of customer vertices;
- $F = \{f_1,\ldots,f_m\}, |F| = m$, is the set of facility vertices;
- $E = C \times F$ is the set of edges between vertices from C and F;
- $w : E \to \mathbb{R}_{\geq 0}$ is a weight function assigning positive real values

[5] A comparison of BCLP with other facility location problems is available elsewhere [33].

(the distances) to the edges between facility nodes and customer nodes.

In addition to the notation defined above, the following constant parameters are defined:

- $p, p \leq m$, is the number of facility vertices required to delete;
- $\Delta, \Delta \geq 0$, is the threshold by which the minimum edge weight of every customer vertex can be increased after deletion of facilities vertices.

Definition 1 (Problem $P^{\Delta} = (G(C,F,E,w),p,\Delta)$) *Given a tuple $P^{\Delta} = (G(C,F,E,w),p,\Delta)$ where $G(C,F,E,w)$ is an undirected edge-weighted complete bipartite graph, p is a positive integer, and Δ is a non-negative real, problem P^{Δ} consists in computing a subset $F' \subseteq F, |F'| = p, p \leq m$, such that if F' is deleted from F, the number of customer vertices in C, whose minimum edge weight is increased by more than Δ, is minimal.*

Clearly, there are $\binom{m}{p}$ different subsets $F' \subseteq F, |F'| = p, p \leq m$.

Given a graph $G(C,F,E,w)$, it can be efficiently determined whether the weight of the edge (c_i, f_j) is no more than Δ greater than the minimum edge weight among all edges from E that are incident to c_i:

$$\alpha_{ij} = \begin{cases} 1, & \text{if } 0 < w(c_i,f_j) \leq min_{1 \leq k \leq m} w(c_i,f_k) + \Delta; \\ 0, & \text{otherwise.} \end{cases}$$

If $\Delta = 0$, then α_{ij} shows whether facility j is the closest to customer i. It is clear that $\sum_{1 \leq k \leq m} \alpha_{ik} \geq 1, 1 \leq i \leq n$. Note, that if $\alpha_{ij} = 1$, then the closure of facility j may make customer i dissatisfied. Otherwise, if $\alpha_{ij} = 0$, then the closure of facility j cannot make customer i dissatisfied.

Let us define a Boolean variable

$$y_j = \begin{cases} 1, & \text{if facility vertex } f_j \text{ will remain undeleted;} \\ 0, & \text{if facility vertex } f_j \text{ is chosen for deletion.} \end{cases}$$

Since p facilities are to be closed, $\sum_{1 \leq j \leq m} y_j = n - p$.

Now, let us define one more set of Boolean variables z_i that show whether the minimum weight among all edges that are incident to customer vertex c_i will increase by more than Δ after deletion of facility vertices F'.

$$z_i = \begin{cases} 1, & \text{if } \sum_{1 \leq k \leq m} \alpha_{ik} y_k = 0; \\ 0, & \text{if } \sum_{1 \leq k \leq m} \alpha_{ik} y_k \geq 1. \end{cases}$$

In other words, if all facilities within distance $R + \Delta$, s.t. R is the distance to the closest facility from customer i, are to be closed, then $z_i = 1$. Otherwise, if at least one of them remains open then $z_i = 0$.

Based on the introduced definitions, optimization problem P^{Δ} can be formulated as follows:

$$\text{min:} \quad \sum_{1 \leq i \leq n} z_i \tag{2}$$

$$\text{s.t.:} \quad \sum_{1 \leq j \leq m} y_j = n - p \tag{3}$$

$$y_j \in \{0,1\} \quad \forall j \in \{1,\ldots,m\} \tag{4}$$

$$z_i \in \{0,1\} \quad \forall i \in \{1,\ldots,n\} \tag{5}$$

Problem P^{Δ} is related to a number of location problems, e.g. to the maximal covering location problem [11] and to the budget-constrained location problem with opening and closing of facilities [45].

To prove that P^{Δ} is NP-hard, consider its special case where $\Delta = 0$. (In this case Δ can be omitted.)

Definition 2 (Problem $P^0 = (G(C,F,E,w),p)$) *Given a tuple $P^0 = (G(C,F,E,w),p)$ where $G(C,F,E,w)$ is an undirected edge-weighted complete bipartite graph $G(C,F,E,w)$ and p is a positive integer, P^0 consists in computing a subset $F' \subseteq F, |F'| = p, p \leq m$, such that if F' is deleted from F, the number of customer vertices in C, whose minimum edge weight is increased, is minimal.*

Also, consider the decision variant of P^0, i.e. the problem of determining whether or not it is possible to delete p facility vertices from F such that $\sum_{1 \leq i \leq n} z_i = 0$. Below we give a formal definition of this decision problem and prove that it is NP-complete.

Definition 3 (Problem $P = (G(C,F,E,w),p)$) *Given an undirected edge-weighted complete bipartite graph $G(C,F,E,w)$ and a positive integer p, problem $P = (G(C,F,E,w),p)$ is to determine whether there exists a subset $F' \subseteq F, |F'| = p$ such that if F' is deleted from F,* there are no *customer vertices in C, whose minimum edge weight is increased.*

Theorem 1 *Problem P is NP-complete.*

Proof. Since for every customer vertex c_i the values of $\alpha_{ik}, 1 \leq k \leq m$ are known, the set of its closest facility vertices (i.e. the facility vertices with minimum edge weight among all facility vertices that have an edge with c_i) can be constructed in polynomial time. Therefore, given a subset of facility vertices $F' \subseteq F, |F'| \geq p$, it can be checked in polynomial time (for every customer vertex) whether or not at least one closest facility vertex belongs to the complement of F', i.e. $F'' = F \setminus F'$. Therefore, P belongs to NP.

The next step is to construct a polynomial-time reduction of an NP-complete problem to P. Consider the *set cover problem* in its decision version [27]. An instance of this problem is $\langle X, \mathscr{F}, k \rangle$, where X is a finite set of elements, $\mathscr{F}$ is a family of subsets of X such that every element of X belongs to at least one subset in $\mathscr{F}$, k is a positive integer. A cover is a subfamily $\mathscr{C} \subseteq \mathscr{F}$ of subsets whose union is X. The set cover problem is to determine whether there exists a cover of X of size k.

Let us construct an algorithm that reduces the set cover problem to problem P. The reduction algorithm takes as input an instance $\langle X, \mathscr{F}, k \rangle$ of the set cover problem, where $|X| = n, |\mathscr{F}| = m$. The output of the reduction algorithm is the instance $\langle G(C,F,E,w), p \rangle$ of problem P, where $p = n - k$ and G is an undirected edge-weighted complete bipartite graph.

The reduction algorithm transforms every element from X into a vertex $c \in C$ and every set $U \in \mathscr{F}$ into a vertex $f \in F$, so $|C| = |X| = n, |F| = |\mathscr{F}| = m$. For every $x \in X$ and $U \in \mathscr{F}$, the algorithm constructs an edge (c,f). The weight $w(c,f)$ of the edge is set to 1 if x occurs in U; otherwise $w(c,f)$ is set to 0. As a result, an undirected weighted complete bipartite graph $G(C,F,E,w)$ is obtained.

It is clear that the proposed reduction works in polynomial time. Now suppose that the answer to the instance $\langle X, \mathscr{F}, k \rangle$ is "yes", i.e. there exists a cover $\mathscr{C} \subseteq \mathscr{F}$ of size k whose union is X. We claim that the answer to the corresponding instance $\langle G(C,F,E,w), n-k \rangle$ is "yes". Indeed, k sets from the cover are transformed to $F'', |F''| = k$, such that for every $c_i \in C$ there exists $f_j \in F'', w(c_i,f_j) = 1$. It means that after performing the deletion of all vertices $F' = F \setminus F'', |F'| = n-k$, at least one closest facility vertex of every customer vertex will remain undeleted.

Conversely, suppose that a problem instance $\langle G(C,F,E,w),p\rangle$ is solved, where $p = n-k$, and the answer to the instance is "yes". Assume that the set of vertices $F'' \subseteq F, |F''| = k$, remains undeleted. Clearly, C corresponds to X, F corresponds to $\mathscr{F}$, while F'' corresponds to a cover $\mathscr{C}, |\mathscr{C}| = k$. Therefore, the answer to the problem instance $\langle X,\mathscr{F},k\rangle$ is "yes".

As a result, the set cover problem is polynomial-time reducible to P, and thus the latter problem is NP-complete. □

The following important corollary follows from Theorem 1.

Corollary 1 *Problem P^{Δ} is NP-hard.*

Proof. (Sketch) NP-completeness of simplified version $P(G(C,F,E,w),p)$ implies that its optimization variant $P^0(G(C,F,E,w),p)$ is NP-hard, from which NP-hardness of the general formulation of P^{Δ} follows. □

In the next section, we show how a propositional encoding of problem $P^{\Delta} = (G(C,F,E,w),p,\Delta)$ can be naturally constructed based on the proposed formulation.

4 MAXSAT FORMULATIONS

This section describes a basic encoding of the P^{Δ} problem into maximum satisfiability[6] followed by a discussion on a few improvements to the basic MaxSAT model.

Basic Encoding. The basic model is straightforward and follows the problem definition outlined in Section 3. It creates a *partial* CNF formula $\varphi \triangleq H \wedge S$. (Recall that the hard part H contains clauses that must be satisfied while the soft part S represents the cost function, i.e. the optimization criterion.) Given n customers and m facilities, formula φ makes use of variables $y_j \in \{0,1\}$ and $z_i \in \{0,1\}$ s.t. $1 \leq j \leq m$ and $1 \leq i \leq n$.

Formula H is represented as a conjunction $H = H_1 \wedge H_2$ s.t.:

$$H_1 \triangleq \Big(\sum_{1\leq j\leq m} y_j = n-p\Big) \tag{6}$$

$$H_2 \triangleq \bigwedge_{1\leq i\leq n} (\neg z_i \leftrightarrow \bigvee_{1\leq j\leq m} \alpha_{ij} y_j) \tag{7}$$

It should be noted that every α_{ij} here is a Boolean constant determined *once*, based on the given distances between the customers and facilities and the value of the threshold Δ. Formula H_2 encodes that if a customer is satisfied then at least one of the relevant facilities[7] should remain, and vice versa. Also, observe that H_1 can be simply represented by any of the cardinality encodings known, e.g. see [5, 4, 1, 22, 43, 3, 38, 35].

The soft part comprises the following set of clauses:

$$S \triangleq \{(\neg z_i, 1) \mid 1 \leq i \leq n\} \tag{8}$$

Clearly, every soft clause $(\neg z_i, 1)$ represents a preference to satisfy the corresponding customer.[8] Also note that each soft clause has weight 1, i.e. each customer contributes to the cost function equally.

[6] Due to the tight relationship between pseudo-Boolean (PB) solving (resp. optimization) and SAT (resp. MaxSAT), adapting the following ideas for constructing a PB encoding of the studied problem is straightforward. The same holds for the integer linear programming (ILP) and constraint programming (CP) technology.

[7] As mentioned above, for each customer, the list of the relevant facilities includes all the facilities that are within distance $R+\Delta$, where $R = min_{1\leq k\leq m} w_{ij}$, i.e. R is the distance to the closest facility.

[8] Customers' demands are modeled as an optimization criterion because we aim at maximizing the number of satisfied customers.

Example 2 *Consider the problem shown in Example 1. It has 3 branches and 6 customers. This means the sets of variables used in the encoding is $\{y_1,y_2,y_3\} \cup \{z_1,z_2,\ldots,z_6\}$. Given the threshold value $\Delta = 300$, we can obtain all the constants α_{ij}:*

$$\begin{array}{lll} \alpha_{11}=1, & \alpha_{12}=0, & \alpha_{13}=0,\\ \alpha_{21}=0, & \alpha_{22}=1, & \alpha_{23}=1,\\ \alpha_{31}=0, & \alpha_{32}=0, & \alpha_{33}=1,\\ \alpha_{41}=1, & \alpha_{42}=0, & \alpha_{43}=0,\\ \alpha_{51}=0, & \alpha_{52}=0, & \alpha_{53}=1,\\ \alpha_{61}=0, & \alpha_{62}=1, & \alpha_{63}=0. \end{array}$$

Therefore and given that $p = 1$, the MaxSAT formula encoding the problem is $\varphi \triangleq H \wedge S$ s.t.

$$H = \left\{ \begin{array}{c} \mathrm{CNF}(y_1+y_2+y_3=2) \\ \wedge \\ \begin{array}{lllll} (z_1 \vee y_1) & \wedge & (\neg z_1 \vee \neg y_1) & \wedge & \\ (z_2 \vee y_2 \vee y_3) & \wedge & (\neg z_2 \vee \neg y_2) & \wedge & (\neg z_2 \vee \neg y_3) \\ (z_3 \vee y_3) & \wedge & (\neg z_3 \vee \neg y_3) & \wedge & \\ (z_4 \vee y_1) & \wedge & (\neg z_4 \vee \neg y_1) & \wedge & \\ (z_5 \vee y_3) & \wedge & (\neg z_5 \vee \neg y_3) & \wedge & \\ (z_6 \vee y_2) & \wedge & (\neg z_6 \vee \neg y_2) & & \end{array} \end{array} \right\}$$

and

$$S = \left\{ \begin{array}{l} (\neg z_1,1) \wedge (\neg z_2,1) \wedge (\neg z_3,1) \wedge \\ (\neg z_4,1) \wedge (\neg z_5,1) \wedge (\neg z_6,1) \end{array} \right\}$$

Here function CNF() *is meant to clausify the argument sum using one of the cardinality encodings.*

Although it is fairly simple and *unweighted*, this MaxSAT model suffers from one significant drawback — it introduces $n+m$ variables, where n is the number of customers while m is the number of facilities (plus a polynomial number of *auxiliary variables* used when encoding (6) to CNF). Clearly, the number of customers can be significantly large, especially if compared with the number of facilities. This may not only result in a large number of soft clauses that a MaxSAT solver has to deal with but it may also harden SAT oracle calls to make by the MaxSAT solver, which is confirmed by the experimental results shown in Section 5.

Eliminating Variables. To address this downside, one may pose a question of whether or not it is possible either to reduce the number of z_i variables, or to dispose of them completely. The answer to this question turns out to be positive. Indeed, from (7) it is immediate that occurrences of variables z_i in the soft clauses can be simply replaced by the equivalent subformulas $\neg(\bigvee_{1\leq j\leq m} \alpha_{ij} y_j)$.

By applying such substitution steps (7) with respect to every variable z_i, one may help dispose of all z_i variables. As a result of such transformation, one obtains a modified formula $\varphi' \triangleq H_1 \wedge S'$, which represents an improved MaxSAT encoding of the considered problem where the modified soft subformula S' is the following:

$$S' \triangleq \{(\bigvee_{1\leq j\leq m} \alpha_{ij} y_j, 1) \mid 1 \leq i \leq n\} \tag{9}$$

Example 3 *Consider the running example problem and its MaxSAT encoding presented in Example 2. Applying the substitution steps (7) leads to MaxSAT formula $\varphi' = H_1 \wedge S'$ equivalent to φ s.t.*

$$S' = \left\{ \begin{array}{lllll} (y_1,1) & \wedge & (y_2 \vee y_3,1) & \wedge & (y_3,1) & \wedge \\ (y_1,1) & \wedge & (y_3,1) & \wedge & (y_2,1) & \end{array} \right\}$$

Aggregating Soft Clauses. Observe that although all variables z_i are now removed from the MaxSAT formula, the number of soft clauses still equals m. However, as one can immediately observe, further optimization of the encoding is possible due to the new conjunction S' of soft clauses having a significant number of duplicate clauses (for instance, see clauses $(y_1, 1)$ and also $(y_3, 1)$, each appearing twice).

By aggregating all l duplicates of a soft clause $(\bigvee_{1\leq j\leq m} \alpha_{ij} y_j, 1)$ and replacing them with a unique *weighted* soft clause $(\bigvee_{1\leq j\leq m} \alpha_{ij} y_j, l)$, one can obtain a new set of weighted soft clauses S'' of a size much smaller than $|S'|$.

Example 4 *Applying the reasoning above to the running example results in the "optimized" MaxSAT formula* $\varphi'' = H_1 \wedge S''$ *s.t.*

$$S'' = \left\{ \begin{array}{ccccc} (y_1, 2) & \wedge & (y_2 \vee y_3, 1) & \wedge & \\ (y_3, 2) & \wedge & (y_2, 1) & & \end{array} \right\}$$

Although the optimizations overviewed here might look simplistic, they turn out to be a crucial factor significantly affecting the performance of the MaxSAT-based approach to branch location problems, as confirmed by the experimental results in Section 5.

5 EXPERIMENTAL RESULTS

This section details the conducted experimental evaluation comparing the performance of the proposed MaxSAT-based approach to the branch location problems against the state of the art in the area [46]. The evaluation aimed at comparing (1) the performance of complete approaches to the problem and (2) the ability of incomplete solvers to approximate optimal problem solutions within a short period of time.

Experimental Setup. The experiments were performed on the *StarExec cluster*[9]. Each process was run on an Intel Xeon E5-2609 2.40GHz processor with 128 GByte of memory, in CentOS. The memory limit for each individual process was set to 32 GByte. The time limit used was either 1800s (for complete solvers) or 60s (for incomplete solvers).

Benchmarks. To construct instances of the branch location problem formulated in Section 3, we used data provided by one of the largest regional banks in Saint Petersburg, Russia. The data contains information about 58 bank branches and approximately 850 thousand their customers.

Following [46], the distance between a bank branch and a customer was calculated as the minimum distance between the branch and the customer's points of interest, e.g. home, work, payment clusters. Payment clusters were calculated by the DBSCAN clustering algorithm [15] based on customers' transactions. Customers transactions from September 2017 to January 2019 were available for this purpose. It turned out, that at least one point of interest is known only for 349 562 customers. This means that distances to bank branches could be calculated only for these customers, that is why we operated only with them. Since branches of the bank naturally correspond to facilities, following the notation of Section 3, we obtain the branch location problem $P^{\Delta} = (G(C, F, E, w), p, \Delta)$, where $|C| = 349562$ and $|F| = 58$. Based on the described data, we considered instances for each $p \in \{1, \ldots, 29\}$. Also, Δ varied from 100 to 500 meters, with increments of 100 meters. In total, 145 instances of the branch location problem were considered. For each of them, two benchmark instances were generated using the *original* (also referred to as *simplistic* or *ordinary*) and *optimized* MaxSAT encodings described in Section 4.

Figure 2 shows the number of variables and clauses in the generated instances. The x-axis shows instances' indices, where the first five of them represent problem instances for $p = 1$ and Δ from 100 to 500 with increments of 100, the next five indices correspond to $p = 2$ and Δ from 100 to 500 with increments of 100, and so on. As can be seen in the figure, the number of variables (clauses) in the *ordinary* instances is always close to the number of customers (two times greater than the number of customers). Also, one can observe that the *optimized* instances are dramatically smaller.

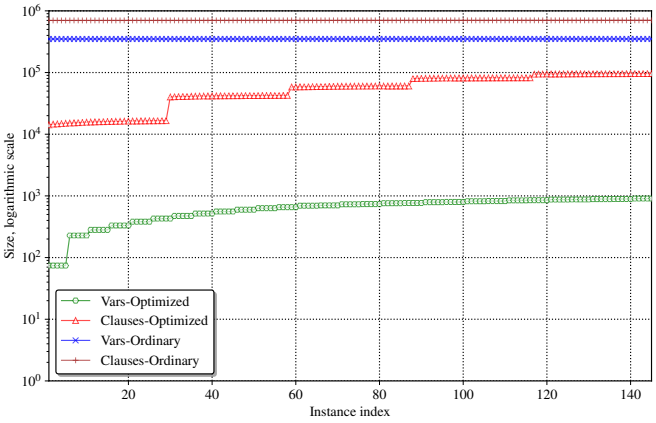


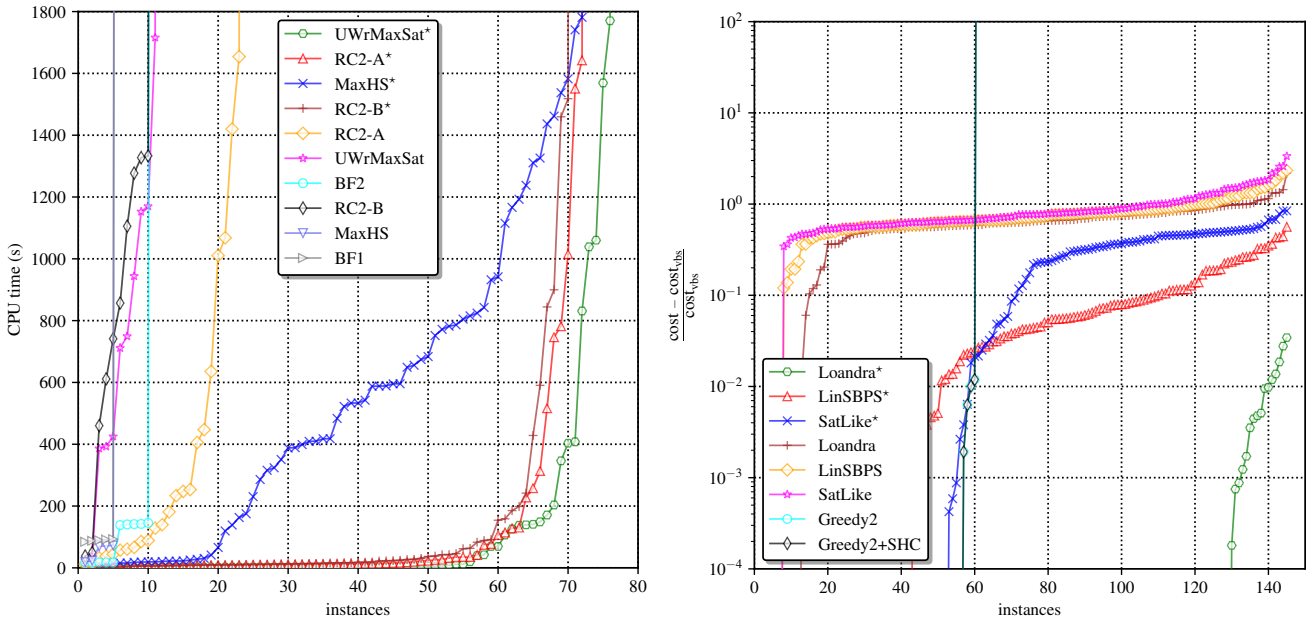
Figure 2: Comparison of *original* and *optimized* encodings by the number of variables and clauses in th generated instances. The y-axis is shown in the logarithmic scale.

Benchmark instances generated this way are different from those studied in [46]. The differences are the following: (i) the new benchmarks reflect more information on customers transactions, customer payment clusters (and so bank-to-customer distances) including four more months of observations, thus, representing user data more accurately. (ii) the new benchmarks consider all 58 bank branches, which was not the case in [46] where two subsets of size 17 and 51 were considered. (iii) a constant value of Δ equal to 200 meters was considered in [46]; and (iv) $p \in \{1, \ldots, 10\}$ was considered in [46]. As a result, [46] analyzed only 20 problem instances in contrast to the current evaluation having 145 instances. Furthermore and given the above, most of the newly constructed 145 instances are more challenging and, thus, more interesting from the practical perspective. This is because all benchmarks considered in the present study encode instances of a *real* practical problem. The solutions obtained (especially for the large values of p that were not studied earlier in [46]) are of particular interest for the bank that provided us with the data. The reason is that the bank aims at significantly reducing the network of its branches in order to decrease the operational costs involved.

Solvers. The MaxSAT approach proposed in the paper was assessed using the state-of-the-art MaxSAT solvers that were top-ranked at MaxSAT Evaluation 2019 [10]. These include 3 complete MaxSAT solvers: RC2 [25, 26], MaxHS [13], and UWrMaxSat [37]. While RC2 and UWrMaxSat exploit *core-guided* algorithms [34], MaxHS builds on *implicit hitting set enumeration* [10]. Additionally, three best performing (according to MaxSAT Evaluation 2019)

[9] https://www.starexec.org/

[10] https://maxsat-evaluations.github.io/2019/



(a) number of instances solved by complete solvers in 1800 sec.

(b) approximation quality of incomplete solvers in 60 sec.

Figure 3: Cactus plots showing the results on the considered benchmark set.

incomplete MaxSAT solvers were selected: Loandra [6], LinSBPS [14], and SatLike [30]. All complete and incomplete solvers were ran on problem instances encoded with the *original* and *optimized* MaxSAT encodings described in Section 4. In the following, the name of each MaxSAT solver dealing with the optimized encoding is augmented with an asterisk, e.g. *Name**.

As the state-of-the-art approach, three solvers from [46] were considered. The first one is referred to as BF1 and implements a brute force algorithm and, hence, is a complete approach. The other two solvers, referred to as Greedy1 and Greedy1+SHC, are incomplete. While the former solver represents a greedy algorithm the latter implements a combination of the greedy algorithm with the simple hill climbing algorithm [41]. The following needs to be mentioned with respect to the implementation of BF1, Greedy1, and Greedy1+SHC. To calculate the number of potentially dissatisfied customers, a list of the relevant branches is required for every customer. Note, that such a list for customer i corresponds to bank branches $1 \leq j \leq m$ for which $\alpha_{ij} = 1$ (see Section 3). As in [46], the implementations of BF1, Greedy1, and Greedy1+SHC construct such lists every time a set of bank branches (candidates for closure) is given.

Note that in contrast to these solvers, the MaxSAT encodings proposed in Section 4 exploit the idea that given a customer, such a list is constructed only once. Inspired by these insights, we also improved all solvers from [46]. This resulted in new solvers being created, namely BF2, Greedy2, and Greedy2+SHC, respectively, that are also considered in the following evaluation.

Results. Figure 3 shows two cactus plots depicting the results of the performed experimental assessment. First, let us focus on the performance of the complete solvers shown in Figure 3a. Observe that MaxSAT solvers dealing with the optimized encoding significantly outperform all the other competitors. Concretely, the best performance is demonstrated by UWrMaxSat*, which solves 76 instances (out of 145). RC2-A* comes second with 72 instances solved. RC2-B* and MaxHS* solve 70 and 72 benchmarks, respectively. Also, it is

Table 1: Detailed performance of the complete solvers tested. Columns 2–5 represent the number of instances solved per solver within the given number of seconds. Also note that each value shown in column $i \in \{3,4,5\}$ represents the sum of the respective number of instances solved and the value shown in column $i-1$.

Solver	by 10sec	by 100sec	by 1000sec	by TO	*unsolved*
BF1	0	5	5	5	*140*
BF2	0	5	10	10	*135*
MaxHS	0	5	5	5	*140*
MaxHS*	3	20	60	72	*73*
RC2-A	0	10	19	23	*117*
RC2-A*	28	59	69	72	*73*
RC2-B	0	2	6	10	*135*
RC2-B*	21	59	68	70	*75*
UWrMaxSat	0	2	8	11	*134*
UWrMaxSat*	53	60	72	76	*69*

not surprising that the original brute force solver BF1 demonstrates the worst performance among all the competitors and can cope with 5 instances only. What is surprising, however, is that MaxHS working with the original problem encoding performs the same way. Although this phenomenon is yet to be explained, our intuition is that MaxHS is "confused" by the majority of variables representing the customers, which affects the number of iterations of the hitting set algorithm, and those variables are redundant (note that are not present in the optimized encoding). The improved version BF2 is more capable and its efficiency is on par with RC2-B and UWrMaxSat dealing with the original problem encoding. Concretely, BF2 is able to successfully deal with 10 instances while RC2-B and UWrMaxSat solve 10 and 11 benchmarks, respectively. Note that RC2-A performs better and manages to solve 23 instances. This can be explained by the fact that it does not apply heuristic unsatisfiable core minimization used in RC2-B and UWrMaxSat and, instead, employs unsatisfiable core trimming [26]. Detailed information on the number of instances

solved by each solver by 10, 100, 1000 seconds, timeout, as well as the number of unsolved instances is provided in Table 1.

It turned out, that the greater the values of p and Δ are, the more difficult the benchmarks become. For instance, UWrMaxSat*, which is the best performing complete solver, solved the instances generated for all combinations of values of Δ (100, 200, 300, 400, 500) and $1 \leq p \leq 13$; for $p = 14$, it managed to solve instances for all values of Δ except for 500; for $p = 15$, it successfully dealt with $\Delta = 100$ and $\Delta = 200$ while for $16 \leq p \leq 20$, it could only cope with $\Delta = 100$. Note that none of the instances generated for $p \geq 21$ were solved.

The second part of the experiment consists of comparing approximation quality obtained by various incomplete solvers within 60 seconds. The rationale here is that in many practical settings (e.g. see [24]), it may be too expensive for a user to wait for a long time, for instance for a half-an-hour, before obtaining an exact solution and it makes sense, instead, to settle for a reasonable approximation if can be computed in a short period of time.

In order to assess the quality of approximate solutions reported by incomplete approaches, a specific *virtual best solver* (VBS) was constructed. If any of the complete solvers compute the cost of the exact optimal solution for a given instance, the VBS is set to *replicate* this optimal solution. Note that out of 145 instances, the complete solvers alltogether manage to solve 82 instances, i.e. for each of these 82 instances the exact optimal solution is known, which facilitates the assessment of incomplete approaches. For the remaining 63 problem instances, the VBS *computes* the smallest cost solution among the approximate solutions reported by all the considered incomplete approaches.

Figure 3b shows a cactus plot depicting the quality of solutions for all incomplete approaches. For a benchmark instance, if the VBS cost is denoted by $cost_{vbs}$ and the approximate solution cost reported by an incomplete solver is denoted by $cost$, the quality of this approximate solution is expressed as the formula $\frac{cost - cost_{vbs}}{cost_{vbs}}$. Hence, each line shown in Figure 3b essentially illustrates the dynamics of the *normalized error* value for the approximate solution per solver, given the VBS.

Observe that BF1, Greedy1, and Greedy+SHC do not appear in the plot. The reason is that they do not report solutions for any of the instances within 60 seconds. Also note that the quality of solutions reported by MaxSAT solvers dealing with the optimized encoding is extremely good. Indeed, the average error for LinSBPS* and SatLike* is around 0.1 and does not go above 1. This means that the cost of the solutions reported by these solvers is at most twice as large as the VBS cost and on average it is greater than the VBS cost by 10%. Moreover, the best solution quality is demonstrated by Loandra*, whose solutions contribute to the VBS most of the time (concretely, for 129 instances). Finally, for the remaining 16 instances, the solutions of Loandra* are never worse than the VBS solutions by more than 10%. The detailed statistics on the correlation between the number of instances and how far the computed cost for these instances per solver is from the VBS cost is provided in Table 2.

Regarding the MaxSAT solvers dealing with the original encoding, the average error is around 100%, i.e. their cost typically doubles the VBS cost. It is an interesting result, which is yet to be understood. As for the BF2, Greedy2, and Greedy2+SHC the cost of their solutions is good for the 60 instances while for the remaining 85 instances they are unable to compute any approximation.

Summary. The experimental evaluation presented in this section confirms the practicality of the proposed MaxSAT-based approach for the branch location problem, in terms of complete problem solving but also in terms of effective solution approximation. Both complete and incomplete MaxSAT solvers are shown to significantly outperform their counterparts representing the current state of the art in the area (including the improved variants of the state-of-the-art solvers). Furthermore, from the practical point of view, the results shown illustrate a good trade-off between exact and approximate solvers and open a number of possibilities to combine complete and incomplete solving for the branch location problems.

Table 2: Analysis of the cost obtained by the incomplete solvers compared to the *VBS cost*.

Solver	Number of instances with cost $\leq (1+k) \times cost_{vbs}$ s.t.					
	$k=0$	$k=0.001$	$k=0.01$	$k=0.1$	$k=1$	$k=10$
Greedy1	0	0	0	0	0	0
Greedy1+SHC	0	0	0	0	0	0
Greedy2	56	56	58	60	60	60
Greedy2+SHC	56	56	58	60	60	60
LinSBPS	8	8	8	8	122	145
LinSBPS*	42	45	50	110	145	145
Loandra	12	12	12	14	134	145
Loandra*	129	132	140	145	145	145
SatLike	7	7	7	7	110	145
SatLike*	51	55	58	71	145	145

6 CONCLUSIONS

Facility location problems are representative of the field of location science [29]. Despite the wide range of practical applications, and to our best knowledge, constraint-based solutions for facility location problems are mostly non-existent. This paper investigates the concrete case of branch location problems [46], a special case of budget-constraint location [45]. The paper shows that a simple encoding of branch location into MaxSAT is competitive in practice when compared with recently proposed approaches, either based on brute force search and or incomplete greedy solutions [45]. Nevertheless, the insights revealed by this effort served to devise changes to the basic model, which enable practical MaxSAT solvers to perform well in practice, extensively outperforming existing solutions.

The modeling insights revealed by this work can be used in other settings, and so a natural line of work will be to identify other facility location problems where similar modeling solutions also enable MaxSAT solvers to excel. Moreover, since the proposed techniques are independent of the concrete problem being solved, another line of work will be to investigate settings where the work in the paper can also be applied. Finally, a natural line of work would be making a comparison of the proposed MaxSAT approach with the PB, ILP and CP technology as the corresponding problem encodings can be easily constructed based on the proposed ideas.

ACKNOWLEDGEMENTS

We thank anonymous reviewers for their thoughtful and constructive comments that made it possible to significantly improve the quality of the present paper. We are grateful to professor Alexander Semenov for his valuable comments regarding the proof of Theorem 1. Oleg Zaikin is funded by Russian Science Foundation (project No. 16-11-10046). Joao Marques-Silva is supported by the AI Interdisciplinary Institute ANITI. ANITI is funded by the French “Investing for the Future – PIA3” program under the Grant agreement n^o ANR-19-PI3A-0004.

References

[1] Carlos Ansótegui and Felip Manyà, 'Mapping problems with finite-domain variables to problems with boolean variables', in *SAT*, pp. 1–15, (2004).
[2] Alireza Boloori Arabani and Reza Zanjirani Farahani, 'Facility location dynamics: An overview of classifications and applications', *Computers & Industrial Engineering*, **62**(1), 408–420, (2012).
[3] Roberto Asín, Robert Nieuwenhuis, Albert Oliveras, and Enric Rodríguez-Carbonell, 'Cardinality networks and their applications', in *SAT*, pp. 167–180, (2009).
[4] Olivier Bailleux and Yacine Boufkhad, 'Efficient CNF encoding of boolean cardinality constraints', in *CP*, pp. 108–122, (2003).
[5] Kenneth E. Batcher, 'Sorting networks and their applications', in *AFIPS*, pp. 307–314, (1968).
[6] Jeremias Berg, Emir Demirovic, and Peter J. Stuckey, 'Core-boosted linear search for incomplete maxsat', in *CPAIOR*, pp. 39–56, (2019).
[7] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, eds. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.
[8] Elazar Birnbaum and Eliezer L. Lozinskii, 'Consistent subsets of inconsistent systems: structure and behaviour', *J. Exp. Theor. Artif. Intell.*, **15**(1), 25–46, (2003).
[9] Sally C Brailsford, Chris N Potts, and Barbara M Smith, 'Constraint satisfaction problems: Algorithms and applications', *European journal of operational research*, **119**(3), 557–581, (1999).
[10] Karthekeyan Chandrasekaran, Richard M. Karp, Erick Moreno-Centeno, and Santosh S. Vempala, 'Algorithms for implicit hitting set problems', in *SODA*, pp. 614–629, (2011).
[11] Richard Church and Charles ReVelle, 'The maximal covering location problem', *Papers of the Regional Science Association*, **32**(1), 101–118, (Dec 1974).
[12] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to Algorithms, 3rd Edition*, MIT Press, 2009.
[13] Jessica Davies and Fahiem Bacchus, 'Solving MAXSAT by solving a sequence of simpler SAT instances', in *CP*, pp. 225–239, (2011).
[14] Emir Demirovic and Peter J. Stuckey, 'Techniques inspired by local search for incomplete maxsat and the linear algorithm: Varying resolution and solution-guided search', in *CP*, pp. 177–194, (2019).
[15] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu, 'A density-based algorithm for discovering clusters in large spatial databases with noise', in *KDD*, pp. 226–231, (1996).
[16] Reza Zanjirani Farahani, Nasrin Asgari, Nooshin Heidari, Mahtab Hosseininia, and Mark Goh, 'Covering problems in facility location: A review', *Computers & Industrial Engineering*, **62**(1), 368–407, (2012).
[17] Reza Zanjirani Farahani and Masoud Hekmatfar, *Facility location: concepts, models, algorithms and case studies*, Springer, 2009.
[18] Reza Zanjirani Farahani, Masoud Hekmatfar, Alireza Boloori Arabani, and Ehsan Nikbakhsh, 'Hub location problems: A review of models, classification, solution techniques, and applications', *Computers & Industrial Engineering*, **64**(4), 1096–1109, (2013).
[19] Reza Zanjirani Farahani, Masoud Hekmatfar, Behnam Fahimnia, and Narges Kazemzadeh, 'Hierarchical facility location problem: Models, classifications, techniques, and applications', *Computers & Industrial Engineering*, **68**, 104–117, (2014).
[20] Reza Zanjirani Farahani, Maryam SteadieSeifi, and Nasrin Asgari, 'Multiple criteria facility location problems: A survey', *Applied Mathematical Modelling*, **34**(7), 1689–1709, (2010).
[21] Mohammad M Fazel-Zarandi, Oded Berman, and J Christopher Beck, 'Solving a stochastic facility location/fleet management problem with logic-based benders' decomposition', *IIE Transactions*, **45**(8), 896–911, (2013).
[22] Ian P. Gent and Peter Nightingale, 'A new encoding of alldifferent into SAT', in *ModRef*, pp. 95–110, (2004).
[23] Peter Haug, 'A multiple-period, mixed-integer-programming model for multinational facility location', *Journal of Management*, **11**(3), 83–96, (1985).
[24] Alexey Ignatiev, Mikolás Janota, and João Marques-Silva, 'Towards efficient optimization in package management systems', in *ICSE*, pp. 745–755, (2014).
[25] Alexey Ignatiev, António Morgado, and João Marques-Silva, 'PySAT: A Python toolkit for prototyping with SAT oracles', in *SAT*, pp. 428–437, (2018).
[26] Alexey Ignatiev, Antonio Morgado, and Joao Marques-Silva, 'RC2: An efficient MaxSAT solver', *Journal on Satisfiability, Boolean Modeling and Computation*, **11**, 53–64, (2019).
[27] Richard M. Karp, 'Reducibility among combinatorial problems', in *Complexity of Computer Computations*, pp. 85–103, (1972).
[28] John G Klincewicz, 'Hub location in backbone/tributary network design: a review', *Location Science*, **6**(1-4), 307–335, (1998).
[29] Gilbert Laporte, Stefan Nickel, and Francisco Saldanha da Gama, *Location science*, volume 528, Springer, 2015.
[30] Zhendong Lei and Shaowei Cai, 'Solving (weighted) partial maxsat by dynamic local search for SAT', in *IJCAI*, pp. 1346–1352, (2018).
[31] Chu Min Li and Felip Manyà, 'Maxsat, hard and soft constraints', In Biere et al. [7], 613–631.
[32] Robert F Love and James G Morris, 'Solving constrained multi-facility location problems involving l_p distances using convex programming', *Operations Research*, **23**(3), 581–587, (1975).
[33] M Teresa Melo, Stefan Nickel, and Francisco Saldanha-Da-Gama, 'Facility location and supply chain management–a review', *European journal of operational research*, **196**(2), 401–412, (2009).
[34] Antonio Morgado, Federico Heras, Mark H. Liffiton, Jordi Planes, and Joao Marques-Silva, 'Iterative and core-guided MaxSAT solving: A survey and assessment', *Constraints*, **18**(4), 478–534, (2013).
[35] Toru Ogawa, Yangyang Liu, Ryuzo Hasegawa, Miyuki Koshimura, and Hiroshi Fujita, 'Modulo based CNF encoding of cardinality constraints and its application to maxsat solvers', in *ICTAI*, pp. 9–17, (2013).
[36] Susan Hesse Owen and Mark S Daskin, 'Strategic facility location: A review', *European journal of operational research*, **111**(3), 423–447, (1998).
[37] Marek Piotrow, 'UWrMaxSat - a new MiniSat+-based solver in MaxSAT evaluation 2019', in *MaxSAT Evalution*, (2019).
[38] Steven David Prestwich, 'CNF encodings', In Biere et al. [7], 75–97.
[39] Philippe Refalo, 'Linear formulation of constraint programming models and hybrid solvers', in *CP*, pp. 369–383, (2000).
[40] Raymond Reiter, 'A theory of diagnosis from first principles', *Artif. Intell.*, **32**(1), 57–95, (1987).
[41] Stuart Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, 3rd edn., 2009.
[42] Güvenç Şahin and Haldun Süral, 'A review of hierarchical facility location models', *Computers & Operations Research*, **34**(8), 2310–2331, (2007).
[43] Carsten Sinz, 'Towards an optimal CNF encoding of boolean cardinality constraints', in *CP*, pp. 827–831, (2005).
[44] Lawrence V Snyder, 'Facility location under uncertainty: a review', *IIE transactions*, **38**(7), 547–564, (2006).
[45] Qian Wang, Rajan Batta, Joyendu Bhadury, and Christopher M. Rump, 'Budget constrained location problem with opening and closing of facilities', *Computers and Operations Research*, **30**(13), 2047 – 2069, (2003).
[46] Oleg Zaikin, Ivan Derevitskii, Klavdiya Bochenina, and Janusz A. Holyst, 'Optimizing spatial accessibility of company branches network with constraints', in *ICCS*, pp. 332–345, (2019).