Theoretical Computer Science 575 (2015) 42–55

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

On a class of covering problems with variable capacities in wireless networks

Selim Akl [a,2], Robert Benkoczi [b,*,1], Daya Ram Gaur [b], Hossam Hassanein [a,1], Shahadat Hossain [b,1], Mark Thom [b]

[a] School of Computing, 557 Goodwin Hall, Queen's University, Kingston, Ontario, K7L 2N8, Canada
[b] Department of Mathematics and Computer Science, University of Lethbridge, 4401 University Dr, Lethbridge AB, T1K 3M4, Canada

ARTICLE INFO

Article history:
Received 15 April 2014
Received in revised form 29 September 2014
Accepted 28 October 2014
Available online 4 November 2014

Keywords:
Facility location
Capacitated covering
Approximation algorithms
Column generation
Complexity
Wireless networks

ABSTRACT

We consider the problem of allocating clients to base stations in wireless networks. Two design decisions are the location of the base stations, and the power levels of the base stations. We model the interference, due to the increased power usage resulting in greater serving radius, as capacities that are non-increasing with respect to the covering radius. Clients have demands that are not necessarily uniform and the capacity of a facility limits the total demand that can be served by the facility. We consider three models. In the first model, the location of the base stations and the clients are fixed, and the problem is to determine the serving radius for each base station so as to serve a set of clients with maximum total profit subject to the capacity constraints of the base stations. In the second model, each client has an associated demand in addition to its profit. A fixed number of facilities have to be opened from a candidate set of locations. The goal is to serve clients so as to maximize the profit subject to the capacity constraints. In the third model, the location and the serving radius of the base stations are to be determined. There are costs associated with opening the base stations, and the goal is to open a set of base stations of minimum total cost so as to serve the entire demand subject to the capacity constraints at the base stations. We show that for the first model the problem is NP-complete even when there are only two choices for the serving radius, and the capacities are $1, 2$. For the second model, we give a $1/2$ approximation algorithm. For the third model, we give a column generation procedure for solving the standard linear programming model, and a randomized rounding procedure. We establish the efficacy of the column generation based rounding scheme on randomly generated instances.

1. Introduction

Given a set of client locations in some metric space, the covering facility location problem is to determine an optimal location for a set of facilities that are required to serve the clients. Facilities can serve clients within a prescribed radius

* Corresponding author.
E-mail addresses: akl@cs.queensu.ca (S. Akl), benkoczi@cs.uleth.ca (R. Benkoczi), gaur@cs.uleth.ca (D.R. Gaur), hossam@cs.queensu.ca (H. Hassanein), hossain@cs.uleth.ca (S. Hossain), thom@cs.uleth.ca (M. Thom).

[1] The author acknowledges the funding support received for this research from NSERC.
[2] The author acknowledges the funding support received for this research from Queen's University.

http://dx.doi.org/10.1016/j.tcs.2014.10.044

only, *i.e.* a client is *covered* or served by a single facility if it is within the covering range of the facility, otherwise it is not covered. The covering problem was introduced in [4] and it has been widely used in practice in areas such as the location of emergency vehicles, of retail facilities, or of telecommunication equipment. However, the simple "all or nothing" covering constraint has been found too restrictive for many applications, and several relaxations have been proposed and studied in the last decade. The excellent survey [2] presents three relaxations: (a) the *gradual cover* model where the degree with which a client is served decreases as its distance to the facility increases; (b) the *cooperative cover* model where several facilities can contribute to serving the same client; (c) the *variable covering radius* model where the planner can choose the covering range for the facilities, but the opening cost for the facility increases with its range.

We introduce a new family of covering problems, Covering with Variable Capacities (*CVC*), which addresses the client coverage problem in the presence of interference in wireless networks. *CVC* generalizes the classical capacitated covering due to [15] where an upper bound on the total demand that can be served by a facility is imposed. Facilities correspond to wireless base stations employing omni-directional antennas and clients represent service subscribers. We assume that the location of the clients is given. Demands (bandwidth requirements) and profits (revenue) are associated with the clients.

In our model, every facility has a variable covering range and the facilities need to be located and assigned a covering range. The range can only be increased at the expense of the capacity, as increasing the power of the radio transmitter causes more interference in the network [3,9,10].

Our paper is motivated by the current research in the field of networking aimed at understanding the connection between interference and base station capacity in networks utilizing the popular code division multiple access (CDMA) technology. Several researchers have investigated the idea of using this dependency to improve the performance of networks. For example, Radwan and Hassanein [16] show that there are significant savings in resource utilization for wide band CDMA networks when the range of the base stations is appropriately chosen, and Tam et al. [18] describe a cellular network that exploits this phenomenon. Arguably our models and solutions have immediate applications in the mobile telephony industry. In addition we extend the theory of facility location problems by studying a new class of location problems in which the facilities have variable capacities, and the designer not only has to choose a location for the facility but also the capacity at which the facility should operate.

Problem *CVC* can be defined in any metric space. We study three variants of *CVC*: *CVC* with fixed facilities (or simply *CVC*) where the location of the facilities is given and the objective is to maximize the total profit of the clients served, maximum *CVC* where a set of clients with maximum total profit must be covered by a fixed number of facilities, and set-cover *CVC* where the entire set of clients must be covered by a set of facilities with total minimum cost. The three problems are defined next. We use index notation to refer to clients and facilities, and so u_i for some index i refers to a client and a_j for some index j refers to a facility.

Problem 1 *(*CVC*).*
Input:

- A set $\mathcal{U} = \{u_i : i \in \mathcal{I}\}$ of clients where $\mathcal{I}$ is the index set of clients.
- For each client u_i, a non-negative demand or size s_i and profit p_i.
- A set $\mathcal{A} = \{a_j : j \in \mathcal{J}\}$ of open facilities, where $\mathcal{J}$ is the index set of facilities.
- For each facility a_j, a set R_j of allowed ranges and for each $r \in R_j$ a corresponding capacity c_{jr}. We denote by N_{jr} the set of clients within the covering range r of facility a_j.

Output:

- For each facility a_j, a range $r_j \in R_j$.
- For each facility a_j, a subset of clients denoted by $\mathcal{I}_j \subseteq \mathcal{I}$ that are served exclusively by a_j satisfying $\sum_{i \in \mathcal{I}_j} s_i \leq c_{jr_j}$ and $u_i \in N_{jr_j}$ for all $i \in \mathcal{I}_j$.

Objective:

- To maximize the total profit of clients served, $\max \sum_{i \in \bigcup_{j \in \mathcal{J}} \mathcal{I}_j} p_i$.

Problem 2 *(Maximum* CVC*).*
Input:

- Same as for Problem 1. The set of facilities represents *candidate* facility locations.
- A positive integer k.

Output:

- k facilities are to be opened, indexed by $\mathcal{J}^* \subseteq \mathcal{J}$ where $|\mathcal{J}^*| = k$.

- For each open facility a_j for $j \in \mathcal{J}^*$, a range $r_j \in R_j$.
- For each open facility a_j for $j \in \mathcal{J}^*$, a subset of clients indexed by $\mathcal{I}_j \subseteq \mathcal{I}$ that are served exclusively by a_j so that $\sum_{i \in \mathcal{I}_j} s_i \leq c_{jr_j}$ and $u_i \in N_{jr_j}$ for all $i \in \mathcal{I}_j$.

Objective:

- To maximize the total profit of clients served, $\max \sum_{i \in \bigcup_{j \in \mathcal{J}^*} \mathcal{I}_j} p_i$.

Problem 3 *(Set cover* CVC*).*
Input:

- Same as for Problem 1. The set of facilities represents *candidate* facility locations. The model can be augmented to handle arbitrary costs of opening facilities.

Output:

- A subset of facilities to be opened, indexed by $\mathcal{J}^* \subseteq \mathcal{J}$.
- For each open facility a_j for $j \in \mathcal{J}^*$, a range $r_j \in R_j$.
- For each open facility a_j for $j \in \mathcal{J}^*$, a subset of clients indexed by $\mathcal{I}_j \subseteq \mathcal{I}$ that are served exclusively by a_j so that $\sum_{i \in \mathcal{I}_j} s_i \leq c_{jr_j}$, $u_i \in N_{jr_j}$ for all $i \in \mathcal{I}_j$, and all clients are served $\bigcup_{j \in \mathcal{J}^*} \mathcal{I}_j = \mathcal{I}$.

Objective:

- To minimize the number of open facilities, $\min |\mathcal{J}^*|$.

The special case when clients have unit demand and profit, $s_i = p_i = 1$ for all $i \in \mathcal{I}$, is called the uniform version of the corresponding *CVC* problem. *CVC* generalizes the capacitated covering problem of [15] which corresponds to a *CVC* instance where the set R_j of covering ranges for each facility has cardinality one.

Our contributions We introduce a new class of covering problems with variable capacities that arise in wireless networks. We show that *CVC* with fixed facilities, uniform clients (both the profit and the demand are equal to one for every client) and a capacity function with two ranges serving either two clients or one, is NP-complete. Three natural integer programming formulations for the *CVC* problem with fixed facilities are described but they exhibit a large integrality gap. The formulations can be extended to handle both the set cover and maximum versions. Unfortunately, the formulations are too large to be solved in practice even if we relax the integrality constraints. We give strong evidence that a set cover based linear programming formulation for the set cover *CVC* combined with a simple rounding procedure is very effective at finding approximate solutions in practice. We then give a greedy $\frac{1}{2} - \epsilon$ approximation algorithm, with a simple analysis for the maximum *CVC* where the clients have arbitrary demands and profits and there are no assumptions on the shape of the capacity function. The running time of the greedy algorithm can be significantly reduced for the version of the problem with unitary demands and profits. Finally, we show that all three types of uniform *CVC* can be solved optimally in polynomial time when the clients and the facilities are located on a line and when the facilities have two ranges with capacities 1 and 2. We conjecture that the problems on the line can be solved in polynomial time if facilities are allowed a constant number of ranges.

2. Complexity

We note that the *CVC* problem with a single fixed facility, with arbitrary client demands and profits, and with a single covering range and a fixed capacity that is part of the input, is NP-complete as it is equivalent to the knapsack problem. In this section, we show that the uniform *CVC* problem is NP-complete even when all the fixed facilities are identical and have only two allowed covering ranges $(1, 2)$. For this proof, we consider the two dimensional Euclidean metric space with the Euclidean distance.

The proof is a reduction from a variant of the Planar 3-SAT problem. The Planar 3-SAT problem is the satisfiability problem of a Boolean formula in conjunctive normal form whose clause-variable graph is planar. The clause-variable graph has a vertex for each variable and each clause, and an edge between a variable and a clause vertex if the clause contains the variable either as a positive or as a negated literal. Fellows et al. [6] describe the NP-completeness of a variant of the planar 3-SAT problem in which all the variable vertices are connected in a cycle. Such an instance is called the Var-linked Planar 3-SAT (VLP 3-SAT) problem and was used to prove complexity results for several problems on planar graphs in [6]. For a related problem in which the clauses are linked in a cycle as opposed to the variables, [6] showed that the problem is NP-complete even when each variable belongs to exactly three clauses, once negated and twice positive. We use Var-linked planar 3-SAT with an additional property (see the third property in the definition below) for the reduction. For a proof of the third property, similar to the proof in [6] for the clause linked planar 3-SAT, see [13].

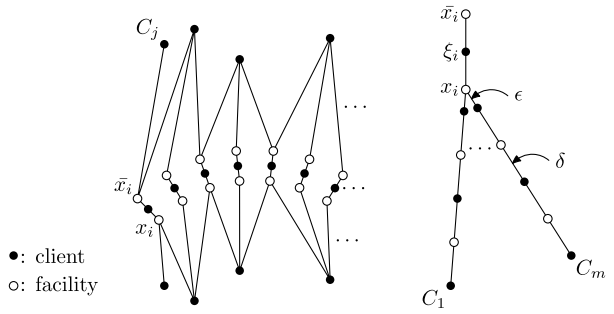


Fig. 1. Reduction from Var-linked Planar 3-SAT. Left: fragment of a planar drawing for the clause-variable graph with variable gadgets (without restriction on the degree of variable nodes). Right: a variable gadget and path gadgets (we assume $m = 3$).

Problem 4 *(Var-linked planar 3-SAT).* Given a formula Φ in CNF such that

- Every clause contains at most three literals.
- Variables follow a linear ordering such that the graph $G_\Phi = (X \cup C, \{(x, c) : x \in C \text{ or } \bar{x} \in C\} \cup \{(x_i, x_{i+1}) : i \in [1 \dots n]\})$ is planar, where $x_{n+1} = x_1$, X is the set of variables, and C is the set of clauses.
- Each variable occurs in exactly three clauses once negated twice positive.

Is Φ satisfiable?

Theorem 1. *The uniform* CVC *problem with fixed facilities (Problem 1) is NP-complete even when the facilities are identical and use only two ranges with capacities in the set* $\{1, 2\}$.

Proof. We consider the decision version of problem *CVC*. Given the location of clients and facilities with two ranges and capacities from the set $\{1, 2\}$, and given a value P, does there exist an assignment of ranges and clients to the facilities which obeys the constraints from Problem 1 so that the total profit of the covered clients is at least P? This problem is clearly in NP since it is trivial to check the lower bound on profit once the set of covered clients is given.

We prove the NP-hardness by giving a reduction from the Var-linked Planar (VLP) 3-SAT problem with the three restrictions. Given an instance of the VLP 3-SAT problem, we first construct a planar drawing of its clause-variable graph with the variables linked. We then modify the drawing by replacing each variable vertex with a three node construction that we call *variable gadget*. The presence of a cycle linking all of the variables allows us to place the variable gadget in such a way that no edge crossings are introduced. We also replace every edge in the planar drawing with a *path gadget*. We construct a geometric *CVC* problem instance by choosing facilities and clients among gadget nodes and by assigning ranges and capacities for each facility. We then show that the original Boolean formula is satisfiable if and only if the *CVC* problem instance admits a solution in which all of the clients are covered.

Construction: Consider a planar drawing D of the clause-variable graph for the VLP 3-SAT problem. We replace every variable vertex y_i by a path consisting of three vertices x_i, ξ_i, and $\bar{x}_i$ where x_i and $\bar{x}_i$ are facilities and ξ_i is a client. We connect x_i to all clauses C_j that contain variable y_i in positive form and $\bar{x}_i$ to all clauses containing y_i negated. Node ξ_i is connected only to x_i and $\bar{x}_i$ (see Fig. 1). We claim that all of these modifications can be made without introducing edge crossings. To see this, let $\delta_D > 0$ be the smallest distance between a node and a non-incident edge in the drawing D of the original graph. We place x_i and $\bar{x}_i$ at a mutual distance $\delta < \delta_D$, and we orient the path $x_i\bar{x}_i$ so that the edges incident to x_i and $\bar{x}_i$ do not cross. Since there are at most three such edges that do not cross the cycle linking all of the variables, an appropriate orientation for the ends x_i and $\bar{x}_i$ of the path can always be found. Note that ξ_i can be covered either by x_i or by $\bar{x}_i$ only, and this covering encodes a consistent truth assignment to the literals. We will refer to $x_i, \bar{x}_i$ as the *literal facilities*.

Next we introduce additional clients C_j, one for each clause j, referred to as the *clause clients*. We connect the clause clients with the literal facilities (corresponding to the three literals in the clause) by a *path gadget* to be defined next (see Fig. 1). The path gadget consists of an alternating sequence of clients and facilities that starts with a client and ends with a facility. The consecutive nodes in a path gadget are some distance δ apart. Each facility on the path gadget has only one range (at distance δ) therefore each facility on the path gadget can cover at most one client, either its predecessor or its successor client on the path. The client (facility) end of the path gadget is referred to as the c-end (f-end) of the path gadget.

For each clause j, we use three path gadgets to connect client C_j to the literals that occur in the clause. The c-end of the path gadget is connected to the facility x_i provided literal y_i occurs in clause j. As each literal occurs in three clauses, once negated and twice positive (property (iii) in VLP-SAT), the positive literal facilities have two path gadgets connecting the literal with the c-end of the path gadgets. We ensure that the connections have length $\epsilon < \delta$. Similarly the negated literal facilities have exactly one path gadget connecting the literal with the c-end. Next, we set the ranges for the literal

facilities. The negated literals $\bar{x}_i$ have only one capacity range δ with capacity one, at which they can cover either ξ_i or the c-end of some path gadget connected to $\bar{x}_i$. A positive literal x_i has two ranges, at range ϵ with capacity two, it can cover the c-end of both the path gadgets connected to it. At range δ with capacity one, it can cover only the client ξ_i (see Fig. 1). The f-end of the path gadget is connected to the client C_j and the length of the connection is δ. Note that the only way to cover client C_j is by the f-end of some path gadget. In turn this implies that the c-end of at least one of the path gadgets connected to C_j does not cover the corresponding client ξ.

It is not difficult to argue that the total size of all gadgets is still polynomial in the size of the VLP 3-SAT problem. Our main concern is to ensure that the distance between consecutive nodes on a path gadget is small enough so that no facility covers clients from other path gadgets, but sufficiently large so that the size of the path gadgets remains polynomial. Let n be the number of vertices in the clause-variable graph which equals the total number of variables and clauses. The clause-variable graph can be drawn on a grid of size $(n-1)\times(n-1)$ in $O(n)$ time [5,17]. The smallest distance δ between a node on the grid and a line segment on the grid is $\Omega(\frac{1}{n^2})$ [12]. Each path gadget has $O(n^3)$ client and facility nodes and therefore the total size of the *CVC* problem is $O(n^4)$.

($\Rightarrow$) Let y_i be an assignment that satisfies all of the clauses. We show that there is a feasible solution for the *CVC* problem that covers all clients. First set the ranges of the variable gadget facilities as follows.

$$r_{x_i} = \begin{cases} \delta, & \text{if } y_i = 0 \\ \epsilon, & \text{if } y_i = 1 \end{cases} \qquad r_{\bar{x}_i} = \begin{cases} \delta, & \text{if } y_i = 1 \\ \epsilon, & \text{if } y_i = 0 \end{cases}$$

This assignment allows client ξ_i to be covered either by x_i or by $\bar{x}_i$ (Fig. 1). In particular x_i covers ξ_i iff y_i is false. If facility x_i is *free* (that is, it does not cover client ξ_i), then it is used to cover the two c-ends of the path gadgets it is connected to, and this forces an assignment on each of the path gadgets. If facility $\bar{x}_i$ is *free*, then it is used to cover the c-end of the one path gadget it is connected to, and this forces an assignment on the path gadget. In both of the cases above, the f-end is free to cover C_j. If the c-end of a path gadget is connected to a literal facility that is not free, then also we have a forcing assignment on the path gadget, in particular the f-end of the path gadget covers its predecessor client on the path, and so on. Both of the assignments above cover all of the clients on the path gadgets (with c-ends connected either to a free or not free literal facility). It remains to be shown that the client C_j is covered as well. This follows immediately, because each clause has one literal x_i set to true hence the literal facility x_i is free (by the assignment above) to cover the c-end of the path gadget. This leaves the f-end of the path gadget free to cover client C_j.

($\Leftarrow$) Recall that a literal facility x_i is free if it does not cover the corresponding client ξ_i. Set y_i to be true if x_i is free. This assignment is consistent as ξ_i is covered by exactly one of the two literal facilities. We claim that this assignment also satisfies all the clauses. Suppose that clause j is not satisfied. This implies that the three c-ends of the path gadgets incident on C_j are connected to literal facilities that are not free. So the only way to cover the c-ends is by the facilities on the path gadget, this leaves C_j uncovered, a contradiction. □

Remark. If the facilities have only one allowed range then the problem can be solved optimally in polynomial time using maximum flow. Consider the bipartite graph over the facilities and the clients. A client is connected to a facility if it lies in the range of the facility. The source node s is connected to the clients via edges with some high capacity. The facilities are connected to the terminus node t, and the capacity of each such edge equals the capacity given the range. All the other edges have capacity one. A maximum s–t flow in the graph provides an optimal solution to the *CVC* problem when there is a single range for every facility. In light of this, the NP-completeness result above is the best possible. It should also be noted that, if the number of facilities f and the number of ranges r are fixed constants, one can solve a constant number (r^f) of flow problems to discover the optimal solution.

3. Algorithms for *CVC* problem with fixed facilities

In this section, we discuss several approaches to solve *CVC* with fixed facilities. Unless stated otherwise, our results apply to arbitrary metric spaces. We give three integer programming formulations and show that they exhibit a large integrality gap even in the case of unit client demands. We then examine the connection with the separable assignment problem, that implies the existence of a $(1-1/e)$ approximation based on rounding the solution of an exponential linear program. Next we describe a simple greedy approximation algorithm with a performance ratio of $1/2$. We argue that the problem with unit demands can be solved exactly in polynomial time if the clients and facilities, restricted to two ranges with capacities $1, 2$, are located on a line.

3.1. Compact integer programming formulations for CVC with fixed facilities

The integer programs discussed next can be extended to handle set cover and maximum *CVC* problems. They can also be readily modified to account for different demands and profits that are associated with the clients.

Without loss of generality, we assume that the capacity is a non-increasing function of the range of a facility. Let k be the total number of clients and f the total number of facilities. The first model is a natural IP formulation for the problem. We reuse the notation for clients and facilities from earlier. Two types of variables are defined next.

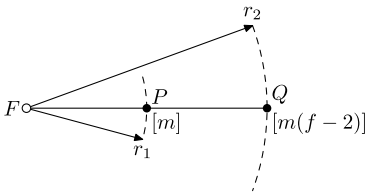


Fig. 2. Instance with integrality gap $f-1$.

$$x_{ua} = \begin{cases} 1, & \text{if client } u \text{ is served by facility } a \\ 0, & \text{otherwise} \end{cases}$$

$$y_{ar} = \begin{cases} 1, & \text{if facility } a \text{ uses range } r \\ 0, & \text{otherwise.} \end{cases}$$

An optimal solution to the uniform *CVC* covering problem is given by the following integer program. N_{ar} is the set of clients that lie within range r for facility a.

$$\max \sum_{u,a} x_{ua} \quad \text{subject to}$$

$$\sum_{r \in R_a} y_{ar} \leq 1, \quad \forall 1 \leq a \leq f \tag{1}$$

$$\sum_{a=1}^{f} x_{ua} \leq 1, \quad \forall 1 \leq u \leq k \tag{2}$$

$$x_{ua} \leq \sum_{r:u \in N_{ar}} y_{ar}, \quad \forall 1 \leq a \leq f,\ 1 \leq u \leq k \tag{3}$$

$$\sum_{u=1}^{k} x_{ua} \leq \sum_{r \in R_a} c_{ar} \cdot y_{ar}, \quad \forall 1 \leq a \leq f \tag{4}$$

Constraint (1) ensures that at most one range is assigned to every facility. Constraint (2) states that a client may be assigned to at most one facility. Constraint (3) links the variables x and y and allows a client to be served by a facility only if the range chosen for the facility covers the client. The last inequality enforces the capacity constraint. The integer program above can be modified to account for demands and profits associated with clients.

Next we exhibit instances with integrality gap $f-1$, even when the facilities and the clients are restricted to be on a line. Furthermore, there are only two ranges associated with each facility, and the clients are uniform in that the demands and the profits are equal to one.

Let all the f facilities be placed at the same location and let the clients be positioned on one of two points denoted P and Q for which the covering capacities are m and 1 respectively (Fig. 2). Equivalently, each facility has two ranges r_1 and r_2 with capacities m and 1 respectively. Let m clients be placed on P and $m(f-2)$ clients be located on Q. In the optimal solution one facility covers the m clients at point P, and the remaining $f-1$ facilities cover additional $f-1$ clients from point Q. Therefore the optimal integral solution has value $m+f-1$. Consider a fractional solution that assigns $x_{ua} = \frac{1}{f}$ for all pairs a and u. Let $y_{ar_1} = (f-1)/f$ and $y_{ar_2} = 1/f$ for all a. Constraints (1), (2), and (3) are satisfied trivially. We only need to verify constraint (4). Indeed, $\frac{m(f-1)}{f} \leq \frac{m(f-1)}{f} + \frac{1}{f}$. The value of the fractional solution is $m(f-1)$, hence the integrality gap is $\frac{m(f-1)}{m+f-1}$ equals $f-1$ for large m.

The second integer program differs from the first in that it explicitly models the difference in capacity between consecutive ranges. It has the same number of variables and a slightly higher number of constraints but the constraints have a simpler structure. The formulation is similar to [19] model for the uncapacitated facility location. This formulation forms also the base for the third integer program that has fewer variables and constraints. Variable x_{ua} has the same interpretation as for integer program (1)–(4), but variable y_{ar} is 1 if the range chosen for facility a is larger or equal to the range r. To simplify the notation, we consider that range r is indexed by its position in a non-decreasing order of all the ranges for a facility. The objective function is the same, but the set of constraints is the following.

$$y_{ar} \geq y_{a,r+1}, \quad \forall 1 \leq a \leq f,\ r < |R_a| \tag{5}$$

$$\sum_{a=1}^{f} x_{ua} \leq 1, \quad \forall 1 \leq u \leq k \tag{6}$$

$$x_{ua} \leq y_{ar}, \quad \forall 1 \leq u \leq k,\ 1 \leq a \leq f,\ r \text{ is the smallest range covering } u \tag{7}$$

$$\sum_{u=1}^{k} x_{ua} \leq c_a - \sum_{r>1} d_{ar} \cdot y_{ar}, \quad \forall 1 \leq a \leq f \tag{8}$$

Coefficient d_{ar} represents the decrease in the capacity arising from the increase in the range from $r-1$ to range r and c_a is the maximum capacity corresponding to range 1. The connection between x and y variables is ensured by constraint (5). If y_{ar} is 1 because of inequality (7) then the variables $y_{ar'}$ for $r' < r$ are also 1.

Finally we obtain the third formulation without the variables y_{ar} as follows:

$$\sum_{a=1}^{f} x_{ua} \leq 1, \quad \forall 1 \leq u \leq k \tag{9}$$

$$\sum_{s=1}^{k} x_{sa} \leq c_a - D_{ua} \cdot x_{ua}, \quad \forall 1 \leq a \leq f,\ 1 \leq u \leq k, \tag{10}$$

where D_{ua} is the loss in capacity, relative to the maximum capacity, for facility a if it serves client u. For each facility a, we have k capacity constraints, and the farthest client served by a determines the capacity of the facility. These formulations have a large integrality gap as well. In addition, the quadratic number of constraints makes these formulations impractical for instances with more than one hundred clients and facilities given the current technology. In Section 5.1, we present a column generation and randomized rounding procedure for the set cover *CVC* problem that seems to work well for most instances generated uniformly at random in two dimensions.

3.2. Known approximation results

Problem 1 is related to the separable assignment problem (SAP). Given a set of bins and a set of items, a value f_{ij} for placing item j in bin i, and for each bin i, a family of sets of items $\mathcal{I}_i$ that fit in bin i, the SAP problem is to pack items into bins so as to maximize the total value of packed items. We note that only subsets of the sets in $\mathcal{I}_i$ can be packed into bin i. SAP was studied by [8] who gave two approximation algorithms, an LP-rounding based $1-1/e$ approximation algorithm, and a local search based $1/2$ approximation algorithm. Given an instance of the *CVC* problem, construct an instance of SAP as follows: each facility corresponds to a bin, for a given range r and location i, we know the capacity c_r of the facility. Define $\mathcal{I}_{i,r}$ to be the set of all valid subsets of total size at most c_r for candidate location i of the facility, let $\mathcal{I}_i = \bigcup_{r \in R} \mathcal{I}_{i,r}$. This reduction implies the existence of a $1-1/e$, LP rounding based approximation algorithm for the *CVC* problem. It should be noted that the size of the resulting LP is exponential, however it can be solved in polynomial time as there exists a separation oracle [8]. The separable assignment problem is reducible to the problem of maximization of a sub-modular function over matroids (SFM); [8] attributes this reduction to Chekuri. Fisher et al. [7] describe two $1/2$ approximation algorithms for the SFM problem; one greedy and the other local search based. Following the reduction (SAP $\propto$ SFM) in [8] the MAX *CVC* problem can also be reduced to SFM.

In the next section, we give a simple and direct $1/2$ approximation algorithm for the MAX *CVC* problem. The bound in Theorem 3 for our greedy algorithm also follows from [7]. In light of the reductions above, the results of [20] and [8] imply an $1-1/e$ approximation algorithm for the maximum *CVC* problem.

3.3. A simple greedy algorithm for CVC with fixed facilities

The greedy algorithm relies on the Knapsack algorithm. We denote by $K(\alpha)$ an α-approximation algorithm for the knapsack problem, and by $V(\alpha)$ the greedy algorithm for *CVC* with fixed facilities that uses $K(\alpha)$ as a subroutine. The crucial idea is to examine facilities in an arbitrary but fixed order and, for each facility, to decompose the *CVC* instance into several independent knapsack instances, one for each range and capacity. The solution to the *CVC* instance is then constructed greedily, by selecting the maximum profit knapsack solutions (see Algorithm 1). The algorithm outputs the total profit of clients covered, the set of clients covered by each facility, and the range for each facility.

Intuitively, one expects that the knapsack subroutine in Algorithm $V(\alpha)$ would not be much worse than the optimal assignment of clients to facilities in the *CVC* instance. The reason why there might be a significant difference between the total profit packed by Algorithm $K(\alpha)$ and the optimal solution is that Algorithm $K(\alpha)$ runs on an instance consisting of those objects not already covered by previous iterations in $V(\alpha)$, and the optimal *CVC* uses too many of these missing objects. However, the "missing" objects are by definition covered by $V(\alpha)$ as well, only they are covered by different facilities. Next we formalize this intuition.

Theorem 2. *Algorithm $V(\alpha)$ computes an $\frac{\alpha}{\alpha+1}$ approximate solution to the* CVC *problem with fixed facilities.*

Proof of Theorem 2. Let a_j be the facility examined by the algorithm in the jth iteration. Let Q_j be the set of clients assigned to a_j in the optimal *CVC* solution but not available to Algorithm $V(\alpha)$ because they were covered by $V(\alpha)$ in

Algorithm 1 $V(\alpha)$

let U be the set of all available clients
for all $j \in \mathcal{J}$, consider facility a_j **do**
  for all ranges $r \in R_j$ **do**
    use Algorithm $K(\alpha)$ to solve the knapsack instance with capacity c_{jr} and set of items N_{jr}.
    let p_{jr} be the profit returned by Algorithm $K(\alpha)$, and let $\mathcal{I}_{jr}$ be the set of packed clients.
  end for
  let $p_j \leftarrow \max_{r \in R_j} p_{jr}$ (p_j is the solution with largest profit)
  let $\mathcal{I}_j$ be the set of clients packed by the most profitable solution
  let r_j be the covering range for the most profitable solution.
  let $\mathcal{I} \leftarrow \mathcal{I} \setminus \mathcal{I}_j$.
end for
return $\sum_{j=1}^{f} p_j$

Fig. 3. Tight example for the $\frac{1}{2}$-factor approximation algorithm for uniform *CVC* with fixed facilities.

previous iterations $1, \ldots, j-1$. Let OPT_j be the set of clients assigned to a_j in the optimal solution. Let $A_j = OPT_j \setminus Q_j$, be the set of clients in the optimal solution that are available to the knapsack subroutine of Algorithm $V(\alpha)$.

If X is a subset of clients, then p_X denotes the total profit for the clients in X. Term $p_{K(\alpha,r,j)}$ represents the total profit for the solution returned by Algorithm $K(\alpha)$ on the knapsack instance used by Algorithm $V(\alpha)$ with facility a_j and range r.

Let r_j be the range chosen by Algorithm $V(\alpha)$ for facility a_j. Because r_j was chosen greedily (with maximum profit) by Algorithm $K(\alpha)$, and the clients in A_j were available in the jth iteration,

$$p_{K(\alpha,r_j,j)} \geq \alpha p_{A_j}.$$

Summing up over all facilities a_j, we obtain

$$\sum_{j=1}^{f} p_{A_j} \leq \frac{1}{\alpha} \sum_{j=1}^{f} p_{K(\alpha,r_j,j)} = \frac{V}{\alpha},$$

where V is the total profit returned by Algorithm $V(\alpha)$. Since set Q_j represents the clients covered by both the optimal *CVC* solution and the solution returned by Algorithm $V(\alpha)$, we also have,

$$\sum_{j=1}^{f} p_{Q_j} \leq V.$$

Summing the last two inequalities we obtain immediately

$$OPT \leq V\left(1 + \frac{1}{\alpha}\right),$$

and the theorem follows. □

For objects with unit sizes and profits, the knapsack problem has trivial optimal solutions, and therefore $\alpha = 1$. By Theorem 2 we get a performance ratio of 1/2.

Corollary 1. *The greedy algorithm for the uniform* CVC *problem with fixed facilities has an approximation factor of* $\frac{1}{2}$.

Next we show that the bound of $\frac{1}{2}$ is tight for the uniform *CVC* problem. Consider the following instance on the real line (Fig. 3): the two facilities are at points 1 and 3, and the two clients are on points 0 and 2. Each facility has only one range (radius 1) with capacity 1. If the greedy approximation algorithm chooses the facilities in the order (1, 3) and the client on point 2 is assigned to the facility on point 1, then the facility at point 3 cannot cover the client at point 0. In the optimal solution, the facility at point 1 (3) covers the client at point 0 (2).

Remark. The tight example also tells us that a modification of the approximation algorithm in which we use the facility that covers the most number of clients (greedily) in each iteration also has performance ratio $\frac{1}{2}$.

Next we analyze the running time for the approximation algorithm. For each facility there are at most n knapsack instances. If we use the FPTAS for Knapsack due to [11] with running time $O(\frac{n^3}{\epsilon})$ to obtain a $\alpha = 1 - \epsilon$ approximate

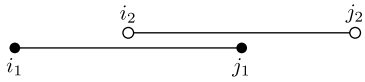


Fig. 4. Clients i_l and j_l are extreme points of intervals I_l, $l \in \{1, 2\}$, for the proof of Lemma 1.

solution to the Knapsack instance, then the total running time is $O(f\frac{n^4}{\epsilon})$ where n is the number of clients and f is the number of facilities. For other algorithms for the knapsack problem, see [14]. For the uniform *CVC* problem the running time can be further reduced to $O(fn\log n)$. In this case, the best knapsack solution for each iteration can be obtained by sorting all clients in non-decreasing order of their distance to the facility (time complexity $O(n\log n)$) and finding the maximum range r_j for which the capacity of the facility is larger than the number of clients covered (time complexity $O(n)$). The same running time complexity can be obtained for the general *CVC* problem if we choose $\alpha = \frac{1}{2}$, that is, if we use the greedy algorithm for the knapsack problem [14]. In each iteration, we can find the profit of the knapsack sub-problem by answering an orthogonal two dimensional range search query [1]. The set of points over which the range query is invoked corresponds to the set of clients in the *CVC* instance and the coordinates of the points are defined by profit density and distance to the facility. The range search query is unbounded and can be implemented in $O(\log n)$ time. The data structures required can be initialized in $O(n\log n)$ time per facility.

4. Algorithms for maximum *CVC*

In this section, we discuss algorithms to solve the maximum *CVC* problem in arbitrary metric spaces. The integer programming formulations discussed in Section 3.1 can be extended in a natural way to handle candidate locations for facilities instead of facilities that are already open. We omit the details. We explain how the simple greedy algorithm from Section 3.3 is extended. Finally, we give a dynamic programming algorithm that solves exactly uniform instances of the maximum *CVC* problem with points on a line when facilities are restricted to two ranges with capacities $1, 2$.

4.1. A greedy algorithm for maximum CVC

Algorithm 1 can be extended to handle maximum *CVC* as follows. The algorithm consists of k iterations, where k is the number of facilities to be opened. In each iteration, we solve a knapsack instance for all remaining candidate facilities and all available covering ranges. We open the facility corresponding to the most profitable knapsack solution found. With this change, the conditions necessary for the proof of Theorem 2 are still satisfied and we obtain the following result.

Theorem 3. *There exists a greedy algorithm that computes an* $\frac{\alpha}{\alpha+1}$ *approximate solution to the maximum* CVC *problem.*

4.2. Maximum CVC with clients and facilities with capacities $1, 2$ on a line

Consider a set of n clients u_i for $i \in \{1, \ldots n\}$ located on a line L and listed in increasing order of their coordinates on L. For simplicity, we refer to the points on the line by their index. Let p_i represent the profit of client i. Facilities may be located anywhere on L between the extreme points 1 and n. We assume that the facilities are identical and that they must be assigned a covering range from a finite set R of allowed ranges with capacity c_r for $r \in R$. The problem asks for a placement of k facilities on the line, an assignment of the covering range for each of the facilities, and an allocation of clients to the facilities that satisfies capacity constraints, so that the total profit of the covered clients is maximized.

In Section 2 we proved that *CVC* with fixed facilities and two ranges with capacities 2 and 1 is NP-hard in the two dimensional plane. Here, we prove that the same problem can be solved in time polynomial in n when the points are restricted to a line, even when the facilities are not fixed. We denote by $d_{i,j}$ the distance between points i and j along the line, and by $p_{i,j}$ the maximum profit that can be reaped by a single facility from the set of clients $\{i, \ldots, j\}$. The following theorem gives a standard dynamic programming algorithm to solve this problem optimally.

Theorem 4. *Let* $Z_p(j)$ *represent the maximum total profit of clients from set* $\{1, \ldots, j\}$ *that can be covered by* p *facilities. By definition,* $Z_0(j) = 0$ *for all* $j \in \{1, \ldots, n\}$. *If* $|R| = 2$ *and* $\{c_r : r \in R\} = \{1, 2\}$, *then*

$$Z_p(j) = \max\Big\{Z_p(j-1),\ \max_{\substack{i\\ i\le j}}\big\{Z_{p-1}(i-1) + p_{i,j}\big\}\Big\}. \tag{11}$$

The proof of the theorem relies on the observation that an optimal solution exists where the sets defined by clients assigned to the same facility are separable. This is true as long as the facilities have two allowed capacities with values 1 and 2. The observation is formalized in the next lemma.

Lemma 1. *Let* f_1 *and* f_2 *be two facilities chosen in an optimal solution. If the facilities are allowed only two ranges with capacities* 1 *and* 2, *and if* I_1 *and* I_2 *are the smallest intervals on the line that contain the clients assigned to* f_1 *and* f_2 *respectively, then there exist an optimal solution where* $I_1 \cap I_2 = \emptyset$.

Proof. Case 1) Assume that I_1 and I_2 are overlapping as in Fig. 4. Let $I_1 \cap I_2 = [i_2, j_1] \neq \emptyset$. We point out that j_1 is within the covering range of f_2 and i_2 is within the covering range of f_1, by convexity. Since all clients have unit demand, we can change the assignment of j_1 from f_1 to f_2 and the assignment of i_2 from f_2 to f_1. Repeating this process with the newly obtained intervals I_1 and I_2, we eventually obtain a solution with cost equal to the optimal where intervals I_1 and I_2 are disjoint.

Case 2) Assume by contradiction that $I_1 \subseteq I_2$. When the set R or allowed ranges is arbitrary, this is in fact possible. However, for $|R| = 2$ and capacities with values 1 and 2, if $I_1 \subseteq I_2$, then f_2 must be assigned the larger of the ranges and its capacity must therefore be equal to 1. We can thus move f_2 over to the single client it is serving and we can assign it the smaller range, and we have reduced this condition to Case 1. □

Proof of Theorem 4. Consider an optimal solution to $Z_p(j)$. Two possibilities exist:

Case 1: Client j is served by a facility. Then, according to Lemma 1, the facility covering j may cover additional clients between i and j for some $i \leq j$ while the clients from $\{1, \ldots, i-1\}$ may be served by the remaining facilities. In this case, the value of cost $Z_p(j)$ is determined by the second argument of function max in (11).

Case 2: Client j is not served in the optimal solution. The cost function $Z_p(j)$ is determined by the value of the first argument of function max in (11). □

The dynamic programming algorithm calculates the value of $Z_p(j)$ for all $p \in \{1, \ldots, k\}$ and $j \in \{1, \ldots, n\}$. The optimal solution to the maximum *CVC* problem on a line with capacities, restricted to be in the set $\{1, 2\}$, is given by $Z_k(n)$. The computation can be carried out in time polynomial in n since computing $p_{i,j}$ amounts to simply finding the two most profitable clients from the set $\{i, \ldots, j\}$. The dynamic programming algorithm for capacities 1 and 2 is simple. It is an open question whether the dynamic programming algorithm can be generalized to an arbitrary set of ranges. We conjecture that this can be achieved by generalizing the definition of term $p_{i,j}$, but the time complexity of the algorithm will be exponential in $|R|$.

We outline next a procedure to compute $p_{i,j}$ using a range tree which might be useful in the generalization of the result. We construct a 1-D range tree [1] having the ordered set of points $\{1, \ldots, n\}$ as leaves. We store at an internal node x, two lists of values corresponding to the leaves that are descendants of x: a list P_x of profit values in non-increasing order and the list S_x of the prefix sums of P_x. If we denote by $l(i)$ the i-th entry of list L, then the following relations are satisfied.

$$p_x(i) \geq p_x(j), \quad \text{for } i \leq j$$
$$s_x(j) = \sum_{i=1}^{j} p_x(i)$$

The lists P_x and S_x determine the maximum profit $p_{i,j}$ by binary search over profit values as follows. Given i and j, let r be the smallest covering range covering all points in $\{i, \ldots, j\}$, *i.e.*, $2r \geq d_{i,j}$. The value of range r can be obtained in $O(n + |R|)$ total amortized time if the values $p_{i,j}$ are queried for decreasing values of i given a fixed j. Let p be the profit value of the current iteration in this binary search procedure. We can find the number of points in $\{i, \ldots, j\}$ whose profit is greater than or equal to p, by binary search over the lists L_x of each of the $O(\log n)$ internal nodes x which cover the interval $\{i, \ldots, j\}$ in the range tree. If this total number of points is larger than c_r, we increase p in the master binary search procedure, otherwise we decrease p until we can no longer adjust p. The value of $p_{i,j}$ is determined by appropriately looking up the entries of lists S_x in the range tree. There are $O(\log n)$ iterations in the master binary search routine, and at each iteration, $O(\log n)$ range tree nodes are searched by binary search, therefore $p_{i,j}$ can be computed in $O(\log^3 n)$.

We point out that for the case of capacities 1 and 2, the range tree nodes need only maintain the best and second best profits and the analysis is considerably simplified.

5. Algorithms for set cover *CVC*

The integer programming formulations described in Section 3.1 can be adapted to the set cover setting in arbitrary metric spaces. Unfortunately, the large number of constraints in the formulations makes it difficult even to compute the optimal solution to the LP relaxation for the model. We describe a heuristic based on linear programming and column generation and provide evidence that the rounding algorithm proposed is effective in practice.

5.1. Column generation and rounding for set cover CVC

Here we give a weighted set cover formulation for the *CVC*. In the weighted set cover, a set U of elements called the universe is given along with a set $\mathcal{S}$ of subsets of U. Each subset $S \in \mathcal{S}$ has a known weight w_S. The goal is to find a minimum weight cover, *i.e.* a set $\mathcal{F} \subseteq \mathcal{S}$ of subsets of the universe U whose union equals U and that minimizes $\sum_{S \in \mathcal{F}} w_S$.

We now describe how to transform an instance of the set cover *CVC* to an instance of the weighted set cover. We use the notation introduced in Problems 1–3. Recall that a feasible allocation of clients $\mathcal{I}_j$ to facility a_j, that is assigned a range r must satisfy the following three conditions.

Algorithm 2 Rounding

```
Let S be the set of generated columns.
Let x*_S for S ∈ S be the optimal LP solution.
F ← ∅.
repeat
  for all S ∈ S do
    x_S ← 1 with probability x*_S, otherwise x_S ← 0.
    F ← F ∪ {S : x_S = 1}
  end for
until F is a set cover
for all S ∈ F in random order do
  if F \ {S} is a set cover then
    F ← F \ {S}
  end if
end for
Return F
```

$$j \in \mathcal{J}^* \tag{12}$$

$$\sum_{i \in \mathcal{I}_j} s_i \leq c_{ar}, \tag{13}$$

$$u_i \in N_{jr}, \quad \text{for all } i \in \mathcal{I}_j \tag{14}$$

Constraint (12) states that clients can only be assigned to open facilities. Inequality (13) is the capacity constraint and inequality (14) is the coverage constraint.

We can construct an equivalent weighted set cover instance as follows. The universe U corresponds to the set of clients. Set $\mathcal{S}$ corresponds to the set of all feasible assignments of clients to candidate facility a_j for a covering range $r \in R_j$ that satisfies constraints (13)–(14), for all choices of a and r. The size of set $\mathcal{S}$ is exponential in the number of clients and facilities. The linear programming (LP) relaxation of the weighted set cover is solved by column generation. In practice the column generation procedure seems to work well for computing the optimal solution to the LP relaxation.

5.1.1. Column generation

Consider the LP relaxation of the weighted set cover problem,

$$\min \sum_{S \in \mathcal{S}} w_S x_S, \tag{15}$$

$$\sum_{S, u \in S} x_S \geq 1, \quad \forall u \in U, x_S \geq 0. \tag{16}$$

By convention, $x_S = 1$ if subset S is chosen in the cover, otherwise $x_S = 0$. Problem (15) is called the master problem. It is much simpler than the compact formulations for *CVC* since the coverage and capacity constraints are implicit in the structure of the columns of the constraint matrix.

The master problem is solved iteratively. In each iteration, only a small set of variables x_S are explicit. Once an optimal fractional solution to the master problem (15) is computed, we compute a new set $\mathcal{I}_{j'}$ of clients that are served by some facility $a_{j'}$ at some covering range $r \in R_{j'}$ subject to capacity and coverage constraints and that minimizes the reduced cost $\rho_{j'} = 1 - \sum_{i \in \mathcal{I}_{j'}} y_i$, where y_i is the optimal dual value of the master problem (15) corresponding to client i.

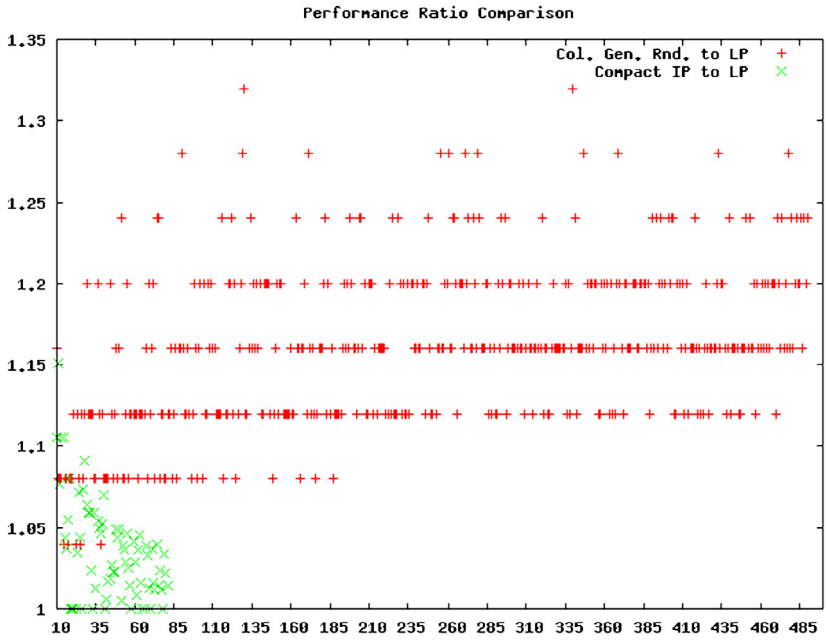
If there is no column with negative reduced cost then the optimal fractional solution to the master problem is also an optimal fractional solution for the set cover *CVC* and the iterative procedure stops. Otherwise, the new column corresponding to set $\mathcal{I}_{j'}$ is added to the constraint matrix of the master problem and the entire procedure is repeated.

We now describe the sub-problem, the procedure for generating the column with negative reduced cost. We iterate over every candidate facility location a_j and covering range $r \in R_j$. To minimize ρ_j, we seek a subset of clients reachable from facility a_j with range r that maximizes the sum of the dual variables subject to the capacity constraint. This is a knapsack problem with the set of items $i \in N_{jr}$, profit y_i and knapsack capacity c_{jr}. In particular, when the demand is uniform, this knapsack problem can be solved exactly by a greedy procedure.

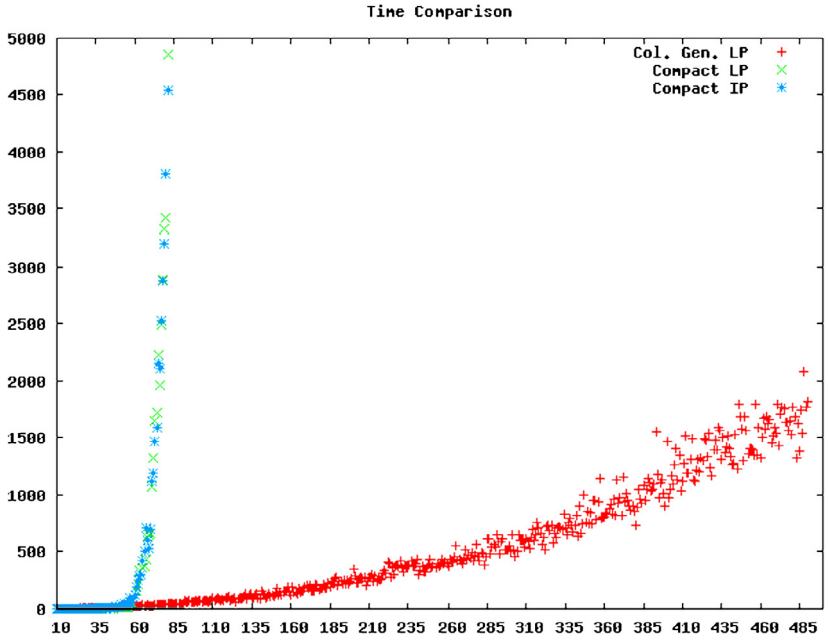
5.1.2. Rounding

At the end of the column generation phase, we have an optimal fractional solution to the set cover *CVC*. If the solution is not integral, we round the fractional solution to obtain an integral solution. The solution may not be optimal, and the value of the linear programming solution is a lower bound on the optimal solution which gives us a measure of the quality of the solution of the rounding step.

The rounding proceeds in two phases. The two phases are as in Algorithm 2. An average of 20 rounding experiments is used to determine the performance ratio for a single instance.



(a) Performance ratio

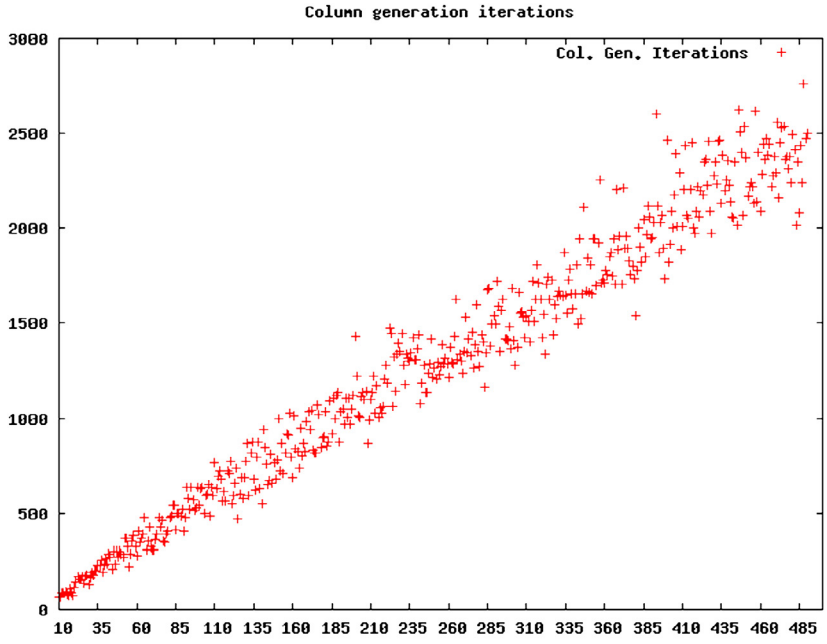


(b) Running time in seconds

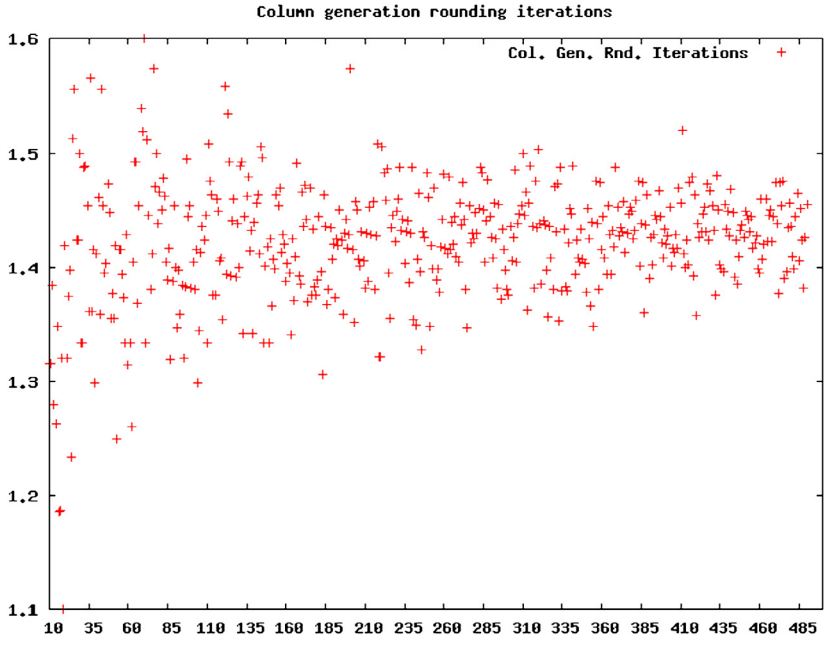
Fig. 5. Results for random planar set cover *CVC* – I.

5.1.3. Simulations

To evaluate the effectiveness of the column generation procedure and the randomized rounding procedure on the set cover *CVC*, we generated close to 500 geometric instances consisting of n points placed uniformly at random in the unit square representing both candidate facility locations and clients. The number of points n varies between 10 and 500 in unit increments. Each candidate facility is assigned 5 covering ranges in the interval $(0, 1)$ and five capacities from the set $\{1, \ldots, 5\}$ uniformly at random in such a way that the capacity is a non-increasing function of the covering range. We also use a compact formulation similar to the first integer programming formulation to compute optimal integral solutions using GNU MILP solver. The column generation, the rounding, and the compact integer and linear formulations were coded in



(a) Total number of columns generated



(b) Number of rounding iterations

Fig. 6. Results for random planar set cover *CVC – II*.

Octave version 3.4.3 and the GNU LP solver was used. The simulations were run on an 16 core (8 CPU) machine running CentOS 2.6 64 GNU/Linux.

The results are depicted in Figs. 5 and 6 on pages 53 and 54. The x axis represents the number of clients in the instances solved. Fig. 5(a) shows the performance ratio, *i.e.* the ratio between the cost of the cover obtained by column generation and rounding and the cost of the optimal solution to the LP relaxation. The experiments include the integrality gap for the compact formulation on the instances for which the integer program reported an optimal solution. From Fig. 5(b) we see clearly that the compact formulations cannot handle instances beyond 90 clients. In contrast, the running time for the column generation with rounding seems to scale well with the size of the problem instance, fact also supported by Fig. 6(a)

that shows an almost linear dependency of the number of generated columns with the problem size. Finally, Fig. 6(b) shows that the average number of rounding iterations is approximately equal to 1.4 and it seems to be independent of the size of the problem. The average was calculated over twenty rounding experiments.

Set cover CVC *with clients and facilities on a line* The set cover *CVC* problem on the line is easier than its maximum counterpart. The problem is defined as in Section 4.2 except there are no profits assigned with clients in set $\{1, \ldots, n\}$ and no integer k is given. The goal is to open the smallest number of facilities to cover all of the clients.

Lemma 1 stating the separability of the client-facility assignment in the optimal solution holds as long as facilities are allowed two ranges with capacities 1 and 2. The solution is trivial in this case as pairs of consecutive clients are covered by one facility only if they are within the reach of a facility at the smaller covering range.

If the dynamic programming algorithm of Section 4.2 generalizes to an arbitrary set of ranges, then we conjecture that this generalization will be applicable to the set cover variant via a greedy algorithm.

6. Conclusions

In this paper, we introduce a new class of covering problems called "Covering with Variable Capacities" which generalizes the well known covering and capacitated facility location problems. We define three classes of problems, set cover *CVC*, maximum *CVC*, and *CVC* with fixed facilities. Our model is inspired from an application in wireless cellular networks, but is appropriate for other applications in facility location as well. We show that *CVC* with fixed facilities and uniform demands is NP-hard. In contrast, under the same restrictions, the capacitated facility location problem can be solved in polynomial time using flows. We give evidence that random instances of the set cover *CVC* have a small integrality gap on average and we propose an efficient column generation algorithm.

We describe a simple greedy algorithm for the maximum *CVC* that achieves a performance bound of $\frac{1}{2} - \epsilon$. The running time of the algorithm is determined by the knapsack FPTAS sub-problem. However, for a slightly worse performance bound of $\frac{1}{3}$, we can solve the general instance *CVC* problem efficiently, in $O(fn \log n)$ time, where f is the number of facilities and n the number of clients. In practice, $f \ll n$. For the uniform *CVC* instance, knapsack can be solved optimally, and the greedy algorithm achieves a performance bound of $\frac{1}{2}$ with the same time complexity.

Open problems We have shown that *CVC* with fixed facilities and with two covering ranges with capacities 1 and 2 is NP-hard in two dimensions, and it is solvable in polynomial time when the points are restricted on a line. The complexity of *CVC* on a line with fixed facilities and an arbitrary set of covering ranges is still open. Finally, *CVC* can be investigated with capacity functions other than the step function, for example linear continuous functions.

Acknowledgements

The authors thank the anonymous referees for their comments that helped with the presentation.

References

[1] M.d. Berg, O. Cheong, M.v. Kreveld, M. Overmars, Computational Geometry: Algorithms and Applications, 3rd edition, Springer-Verlag TILLS, Santa Clara, CA, USA, 2008.
[2] O. Berman, Z. Drezner, D. Krass, Generalized coverage: new developments in covering location models, Comput. Oper. Res. 37 (10) (2010) 1675–1687.
[3] D. Catrein, L.A. Imhof, R. Mathar, Power control, capacity, and duality of uplink and downlink in cellular CDMA systems, IEEE Trans. Commun. 52 (10) (2004) 1777–1785.
[4] R. Church, C. ReVelle, The maximal covering location problem, Papers Reg. Sci. Assoc. 32 (1) (1974) 101–118.
[5] H. De Fraysseix, J. Pach, R. Pollack, How to draw a planar graph on a grid, Combinatorica 10 (1) (1990) 41–51.
[6] M. Fellows, J. Kratochvil, M. Middendorf, F. Pfeiffer, The complexity of induced minors and related problems, Algorithmica 13 (3) (1995) 266–282.
[7] M.L. Fisher, G.L. Nemhauser, L.A. Wolsey, An analysis of approximations for maximizing submodular set functions – II, Math. Program. Stud. 8 (1978) 73–87.
[8] L. Fleischer, M.X. Goemans, V.S. Mirrokni, M. Sviridenko, Tight approximation algorithms for maximum general assignment problems, in: Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm, SODA 2006, ACM, 2006, pp. 611–620.
[9] S. Hanly, Congestion measures in DS-CDMA networks, IEEE Trans. Commun. 47 (3) (1999) 426–437.
[10] H. Holma, A. Toskala, WCDMA for UMTS: HSPA Evolution and LTE, 4th edition, Wiley, 2007.
[11] O.H. Ibarra, C.E. Kim, Fast approximation algorithms for the knapsack and sum of subset problems, J. ACM 22 (4) (Oct. 1975) 463–468.
[12] S. Kwek, K. Mehlhorn, Optimal search for rationals, Inform. Process. Lett. 86 (1) (2003) 23–26.
[13] J. Manuch, D.R. Gaur, Fitting protein chains to cubic lattice is NP-complete, J. Bioinform. Comput. Biol. 6 (1) (2008) 93–106.
[14] S. Martello, P. Toth, Knapsack Problems: Algorithms and Computer Implementations, revised edition, John Wiley & Sons, Chichester, NY, 1990.
[15] H. Pirkul, D.A. Schilling, The maximal covering location problem with capacities on total workload, Manag. Sci. 37 (2) (1991) 233–248.
[16] A. Radwan, H. Hassanein, Capacity enhancement in CDMA cellular networks using multi-hop communication, in: Proceedings of the 11th IEEE Symposium on Computers and Communications, ISCC 2006, IEEE, 2006, pp. 832–837.
[17] W. Schnyder, Embedding planar graphs on the grid, in: Proceedings of the Second Annual ACM-SIAM Symposium on Discrete Algorithm, SODA 1990, ACM, 1990, pp. 138–148.
[18] Y.H. Tam, H.S. Hassanein, S.G. Akl, R. Benkoczi, Optimal multi-hop cellular architecture for wireless communications, in: Proceedings of the 31st IEEE Conference on Local Computer Networks, LCN 2006, IEEE, 2006, pp. 738–745.
[19] A. Tamir, A. Kolen, Covering problems, in: P. Mirchandani, R.L. Francis (Eds.), Discrete Location Theory, Wiley, New York, 1990, pp. 263–304.
[20] J. Vondrak, Optimal approximation for the submodular welfare problem in the value oracle model, in: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, STOC 2008, ACM, 2008, pp. 67–74.