Operations Research Letters 45 (2017) 452–455

Contents lists available at ScienceDirect

Operations Research Letters

journal homepage: www.elsevier.com/locate/orl

Greedy algorithms for the single-demand facility location problem

Sin-Shuen Cheung, David P. Williamson *

School of Operations Research and Information Engineering, Cornell University, Ithaca, NY, 14853, USA

ARTICLE INFO

Article history:
Received 21 September 2016
Received in revised form 7 July 2017
Accepted 8 July 2017
Available online 17 July 2017

Keywords:
Facility location
Approximation algorithm
Greedy algorithm

ABSTRACT

In this note, we give greedy approximation algorithms for the single-demand facility location problem inspired by the greedy algorithms for the min-knapsack problem originally given by Gens and Levner (1979) and later analyzed by Csirik et al. (1991). The simplest algorithm is a 2-approximation algorithm running in $O(n \log n)$ time; in general, we give a $\frac{k+1}{k}$-approximation algorithm running in $O(n^k \log n)$ time.

1. Introduction

The single-demand facility location (SDFL) problem is a well-studied optimization problem. In this problem, we are given a set of potential facilities F to be opened to serve a total demand of D units. Each facility $i \in F$ has an opening cost $f_i \geq 0$, a per-unit shipping cost $c_i \geq 0$, and a capacity limit $u_i > 0$. The goal is to find a subset S of facilities F to open and an assignment $\sigma : S \to \Re^{\geq 0}$ of facilities to demand that minimizes the total cost, such that the capacity of each opened facility is respected and the total demand is assigned. In particular, if we open a subset $S \subseteq F$ and have assignment σ, we need that $\sigma(i) \leq u_i$ for each $i \in S$ and $\sum_{i \in S} \sigma(i) = D$; the goal is to minimize $\sum_{i \in S}(f_i + c_i \cdot \sigma(i))$. We will use $c(S, \sigma) \equiv \sum_{i \in S}(f_i + c_i \cdot \sigma(i))$ to denote the cost of a solution (S, σ). The problem can easily seen to be NP-hard by a reduction from the knapsack problem (see, for instance, Florian et al. [5]).

The single-demand facility location problem is the single-client version of the more general capacitated facility location problem. Although the single-demand facility location problem is NP-hard, it has a fully polynomial-time approximation scheme (FPTAS) given by Carr et al. [3]. A polynomial-time approximation scheme (PTAS) is an algorithm which computes a $(1 + \epsilon)$-approximate solution within polynomial time for any fixed $\epsilon > 0$. An FPTAS further requires that the running time is polynomial in both the input size and $1/\epsilon$. Carr et al. obtain a 2-approximation algorithm for the problem by rounding the solution to an exponentially-sized linear programming relaxation of the problem with flow-cover inequalities. They show how to solve the relaxation via the ellipsoid method. They obtain an FPTAS by using the multiplicative-weight algorithm of Garg and Könemann [6] in combination with a dynamic program to find a most violated constraint. Van Hoesel and Wagelmans [10] give a direct dynamic programming algorithm to obtain an FPTAS. Carnes and Shmoys [2] give a primal–dual 2-approximation algorithm for the problem; it creates a feasible solution to the dual of the flow-cover-based linear programming relaxation given by Carr et al., and uses this dual solution to infer a good solution to the integer primal problem.

In this work, we give a simple greedy algorithm that yields a 2-approximation algorithm for the single-demand facility location problem; we extend this algorithm to a PTAS. Our greedy algorithms are straightforward to understand, and follow the techniques developed by Gens and Levner [7] and later analyzed by Csirik et al. [4] for the minimum knapsack problem. Csirik et al. show that the greedy algorithm of Gens and Levner is a 2-approximation algorithm for min-knapsack, and then refine the algorithm to obtain a 3/2-approximation algorithm. The min-knapsack problem admits an FPTAS, via a straightforward reduction to the standard knapsack problem (see, for instance, Ibarra and Kim [8] for an FPTAS for knapsack). Carnes and Shmoys [2] develop a primal–dual 2-approximation algorithm for min-knapsack. Bienstock and McClosky [1] prove that for any $\epsilon \in (0, 1)$ one can obtain a $(1 + \epsilon)$-approximate solution for the min-knapsack problem by solving a polynomially-sized linear program.

2. Greedy algorithms for the min-knapsack problem

We begin by reviewing the min-knapsack problem, and the greedy algorithm for it given by Gens and Levner [7]. The min-knapsack problem is defined as follows. There are items indexed by $1, \ldots, n$ that can be put into a knapsack. Each item i has volume $c_i > 0$ and value $a_i \geq 0$, and there is a target value D. The goal is to choose a subset $F \subseteq \{1, \ldots, n\}$ of items of minimum

* Corresponding author.
E-mail addresses: sc2392@cornell.edu (S. Cheung), davidpwilliamson@cornell.edu (D.P. Williamson).

http://dx.doi.org/10.1016/j.orl.2017.07.002
0167-6377/

volume such that the total value of items in F is at least D. The standard (max) knapsack problem has the same input except it has a volume limit V rather than a target value D and the goal is to find a subset F of items of maximum value such that the total volume is at most V. The standard version is known to be NP-hard due to Karp [9]. The min-knapsack problem can be shown to be NP-hard via a straightforward reduction from the standard version.

We now review the Gens and Levner [7] greedy algorithm for min-knapsack. First, sort items by non-increasing order of ratio $\frac{a_i}{c_i}$ and redefine the indices by this ordering so that $\frac{a_1}{c_1} \geq \cdots \geq \frac{a_n}{c_n}$. Now define sequences of *small items* and *big items* as follows. Let k_1 be the index such that $\sum_{i=1}^{k_1} a_i < D \leq \sum_{i=1}^{k_1+1} a_i$ and let $S_1 := \{1, \ldots, k_1\}$. Let k_2 be the index larger than k_1 such that $\sum_{i \in S_1} a_i + a_{k_2} < D$ and $\sum_{i=1}^{k_1} a_i + a_l \geq D$ for any $l : k_1 < l < k_2$. Define $B_1 := \{a_{k_1+1}, \ldots, a_{k_2-1}\}$. Let k_3 be the index larger than k_2 such that $\sum_{i \in S_1} a_i + \sum_{i=k_2}^{k_3} a_i < D \leq \sum_{i \in S_1} a_i + \sum_{i=k_2}^{k_3+1} a_i$. Define $S_2 := \{a_{k_2}, \ldots, a_{k_3}\}$, and so on. We can inductively define the small item sets S_j and big items sets B_j so that $S_j = \{a_{k_{2(j-1)}}, \ldots, a_{k_{2j-1}}\}$ and $B_j = \{a_{k_{2j-1}+1}, \ldots, a_{k_{2j}-1}\}$ for all j such that S_j or B_j is defined. Then we have that

$$\sum_{i \in \cup_{j=1}^{l} S_j} a_i + a_r \geq D$$

holds for any l and $r \in B_l$.

Let $\mathcal{C}$ be the set of solutions with the form $\cup_{j=1}^{l} S_j \cup \{r\}$ for $r \in B_l$. The greedy algorithm finds the minimum-cost solution in $\mathcal{C}$ and returns it. Csirik et al. [4] show that this algorithm is a 2-approximation algorithm for the min-knapsack problem.

3. A greedy 2-approximation algorithm for the single-demand facility location problem

In this section, we develop a greedy 2-approximation algorithm for the single-demand facility location problem that is inspired by the min-knapsack algorithm of Gens and Levner [7] and its analysis by Csirik et al. [4].

3.1. Notation and concepts

We use the 3-tuple (f_i, c_i, u_i) to denote a facility i whose opening cost, per unit connection cost, and available capacity are f_i, c_i and u_i respectively. We assume that f_i, c_i and u_i are all nonnegative real numbers for any facility i. An instance of the single-demand facility location problem is denoted as $\mathrm{SL}(\{(f_i, c_i, u_i)\}_{i \in F}, D)$, where $\{(f_i, c_i, u_i)\}_{i \in F}$ is the set of available facilities and the positive real number D is the demand; we define this notation since to obtain our PTAS we will need to create modified instances of the problem. Define $\rho_i = \frac{f_i + c_i u_i}{u_i}$. We reindex facilities so that $\rho_1 \leq \rho_2 \leq \cdots \leq \rho_n$.

Now we define the notion of *small facilities* and *big facilities* as in Csirik et al. [4].

Definition 1. Given the problem $\mathrm{SL}(\{(f_i, c_i, u_i)\}_{i \in F}, D)$, inductively define the following indices and sets.

- Let S_1 be the subset $\{1, \ldots, k_1\} \subseteq F$, where k_1 is the index such that $\sum_{i=1}^{k_1} u_i < D$ and $\sum_{i=1}^{k_1} u_i + u_{k_1+1} \geq D$;
- Let B_1 be the subset $\{k_1 + 1, \ldots, k_2 - 1\} \subseteq F$, where k_2 is the index larger than k_1 such that $\sum_{i \in S_1} u_i + u_l \geq D$ for all $l \in \{k_1 + 1, \ldots, k_2 - 1\}$ and $\sum_{i \in S_1} u_i + u_{k_2} < D$.

In general, we have that

- S_l is the subset $\{k_{2l-2}, \ldots, k_{2l-1}\} \subseteq F$, where k_{2l-1} is the index larger than k_{2l-2} such that $\sum_{i \in \cup_{r=1}^{l-1} S_r} u_i + \sum_{i=k_{2l-2}}^{k_{2l-1}} u_i < D$ and $\sum_{i \in \cup_{r=1}^{l-1} S_r} u_i + u_{k_{2l-1}+1} \geq D$;
- B_l is the subset $\{k_{2l-1} + 1, \ldots, k_{2l} - 1\} \subseteq F$, where k_{2l} is the index larger than k_{2l-1} such that $\sum_{i \in \cup_{r=1}^{l} S_r} u_i + u_s \geq D$ for all $s \in \{k_{2l-1} + 1, \ldots, k_{2l} - 1\}$ and $\sum_{i \in \cup_{r=1}^{l-1} S_r} u_i + u_{k_{2l}} < D$.

Let q be the index such that $|F| \in S_q$ or $|F| \in B_q$. Then we say $\{S_l\}_{l=1}^{q}$ and $\{B_l\}_{l=1}^{q}$ (where B_q is allowed to be an empty set) are the *small sets* and *big sets* respectively for the problem instance $\mathrm{SL}(\{(f_i, c_i, u_i)\}_{i \in F}, D)$. We call the facilities contained in big sets and small sets *big facilities* and *small facilities* respectively.

The small facility sets $\{S_l\}_{l=1}^{q}$ and big facility sets $\{B_l\}_{l=1}^{q}$ allow us to define candidate solutions to the single-demand facility location problem as follows.

Definition 2. Given problem instance $\mathrm{SL}(\{(f_i, c_i, u_i)\}_{i \in F}, D)$ and its small and big facility sets $\{S_l\}_{l=1}^{q}$ and $\{B_l\}_{l=1}^{q}$, define a collection $\mathcal{C}_1$ of pairs (S, σ) for $S = \cup_{l=1}^{q'} S_l \cup \{r\}$ for all $q' = 1, \ldots, q$ and $r \in B_{q'}$; for each such S, define the corresponding σ such that $\sigma(i) = u_i$ for each small facility $i \in S$, and $\sigma(r)$ is the remaining demand, which is assigned to big facility r (that is, $\sigma(r) = D - \sum_{i \in S - \{r\}} u_i$).

To be specific, for $(S, \sigma) \in \mathcal{C}_1$, the set S consists of the first q' small sets $\cup_{l=1}^{q'} S_l$ and one big facility $r \in B_{q'}$ for some q'. By the definition of the small and big facility sets given above, the assignment σ given is always possible.

3.2. The greedy algorithm for single-demand facility location

By choosing the minimum-cost solution over all the candidate solutions in $\mathcal{C}_1$, we will obtain a 2-approximate solution for the single-demand facility location problem. For a given instance $\mathrm{SL}(\{(f_i, c_i, u_i)\}_{i \in F}, D)$, let us denote the algorithm that constructs the corresponding candidate set $\mathcal{C}_1$ and chooses the minimum-cost solution from $\mathcal{C}_1$ as Algorithm $G_1(\mathrm{SL}(\{(f_i, c_i, u_i)\}_{i \in F}, D))$.

Theorem 3. *The algorithm $G_1(SL(\{(f_i, c_i, u_i)\}_{i \in F}, D))$ gives a 2-approximate solution to $SL(\{(f_i, c_i, u_i)\}_{i \in F}, D)$. The algorithm runs in time $O(n \log n)$, where $n = |F|$.*

Proof. Let (S^*, σ^*) be an optimal solution to the problem instance $\mathrm{SL}(\{(f_i, c_i, u_i)\}_{i \in F}, D)$. Notice that there must be at least one big facility in S^* in order for it to be a feasible solution. Let p be the big facility of least index in S^*. Consider the solution $(\hat{S}, \hat{\sigma})$, where $\hat{S}$ consists of all small facilities of index less than p, plus the big facility p, and $\hat{\sigma}$ is such that $\hat{\sigma}(i) = u_i$ for all small facilities in $\hat{S}$, and $\hat{\sigma}(p) = D - \sum_{i \in S - \{p\}} u_i$ is all remaining demand. Clearly this solution $(\hat{S}, \hat{\sigma})$ is in the candidate set $\mathcal{C}_1$ considered by Algorithm G_1. We only need to show that this solution is within a factor of two in cost of (S^*, σ^*). In particular, we will prove that

$$c(\hat{S}, \hat{\sigma}) \leq c(S^*, \sigma^*) + f_p \leq 2 \cdot c(S^*, \sigma^*),$$

which will prove the result.

Recall that $\rho_i = \frac{f_i + c_i u_i}{u_i} = \frac{f_i}{u_i} + c_i$, and that facilities are indexed in order of nondecreasing ρ_i. Since p is the big facility of lowest index in S^*, and $(\hat{S}, \hat{\sigma})$ assigns all demand not assigned to p to small facilities of index smaller than p, a simple interchange argument shows that

$$\sum_{i \in \hat{S}} \rho_i \cdot \hat{\sigma}(i) \leq \sum_{i \in S^*} \rho_i \cdot \sigma^*(i). \tag{1}$$

Because for any small facility $i \in \hat{S}$, $\hat{\sigma}(i) = u_i$, we have that $\rho_i \cdot \hat{\sigma}(i) = f_i + c_i \cdot \hat{\sigma}(i)$. Also, for facility p, $c_p \cdot \hat{\sigma}(p) \leq \rho_p \cdot \hat{\sigma}(p)$.

Thus we have that

$$c(\hat{S}, \hat{\sigma}) \leq f_p + \sum_{i \in \hat{S}} \rho_i \cdot \hat{\sigma}(i). \tag{2}$$

Similarly, since for any facility $i \in S^*$, $\rho_i \cdot \sigma^*(i) \leq f_i + c_i \cdot \sigma^*(i)$, we have that

$$\sum_{i \in S^*} \rho_i \cdot \sigma^*(i) \leq c(S^*, \sigma^*). \tag{3}$$

Thus by combining Inequalities (2), (1), and (3), we get that

$$\begin{aligned} c(\hat{S}, \hat{\sigma}) &\leq f_p + \sum_{i \in \hat{S}} \rho_i \cdot \hat{\sigma}(i) \leq f_p + \sum_{i \in S^*} \rho_i \cdot \sigma^*(i) \\ &\leq c(S^*, \sigma^*) + f_p \leq 2 \cdot c(S^*, \sigma^*), \end{aligned} \tag{4}$$

as desired.

Regarding the time complexity of the algorithm, we need $O(n \log n)$ time to sort the facilities and then $O(n)$ time to define the big and small facilities. There are at most $O(n)$ big facilities, each of which defines a candidate solution in $\mathcal{C}_1$. For each candidate solution, it takes at most $O(n)$ time to compute its cost. Therefore in total we need $O(n^2)$ time to execute the algorithm. However, we can note that as we compute each candidate solution, we consider the index of each big facility in turn, and use all lower indexed small facilities to capacity. Thus we can compute the cost of the next candidate solution by checking the cost of filling the next big facility with the remaining demand, and we do not need to recalculate the cost of the small facilities that have been used thus far. Therefore, we only need $O(1)$ time to compute the cost of each candidate solution, for a total running time of $O(n \log n)$. □

4. Generalization to a polynomial-time approximation scheme

In this section we show how to generalize the algorithm above to a PTAS for the SDFL problem. We give a sequence of algorithms $\{G_k\}_{k \in \mathbb{N}}$, such that each G_k is a $\frac{k+1}{k}$-approximation algorithm running in time $O(n^k \log n)$; we use induction on k to prove the result. We formalize this idea in the following theorem.

Algorithm 1: Algorithm $G_k(\text{SL}(\{(f_i, c_i, u_i)\}_{i \in F}, D))$

if $k = 1$ **then**
  return $G_1(SL(\{(f_i, c_i, u_i)\}_{i \in F}, D))$
else
  $\mathcal{C} \leftarrow G_1(\text{SL}(\{(f_i, c_i, u_i)\}_{i \in F}, D))$;
  foreach *big facility* p *in* $SL(\{(f_i, c_i, u_i)\}_{i \in F}, D))$ **do**
    $(\tilde{S}, \tilde{\sigma}) \leftarrow G_{k-1}(\text{SL}(\{(f_i, c_i, u_i)\}_{i \in F \setminus \{p\} \cup \{\tilde{p}\}}, D))$, where $\tilde{p}$ is the facility $(0, c_p, u_p)$;
    if $\tilde{p} \in \tilde{S}$ **then**
      $S \leftarrow \tilde{S} - \{\tilde{p}\} \cup \{p\}$;
      $\tilde{\sigma}(p) = \tilde{\sigma}(\tilde{p})$
    else
      $(S, \sigma) \leftarrow (\tilde{S}, \tilde{\sigma})$;
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{(S, \sigma)\}$
  return *the solution* $(\hat{S}, \hat{\sigma})$ *in* $\mathcal{C}$ *of minimum cost.*

Theorem 4. *For each positive integer k, Algorithm 1 is a $\frac{k+1}{k}$-approximation algorithm G_k for the SDFL problem. The time complexity for algorithm G_k is $O(n^k \log n)$, where n is the size of the set of facilities.*

Proof. The theorem is true for $k = 1$ by Theorem 3. So assume $k > 1$. By induction we assume that a $\frac{k}{k-1}$-approximation algorithm G_{k-1} exists. Now we show that there is an algorithm G_k with approximation ratio $\frac{k+1}{k}$. Given a problem instance $\text{SL}(\{(f_i, c_i, u_i)\}_{i \in F}, D)$, let (S^*, σ^*) be an optimal solution. Let p be the big facility with least index in S^*, and let $\text{OPT} = c(S^*, \sigma^*)$. We now consider two cases based on the relationship of f_p and OPT.

1. If $f_p \leq \frac{1}{k}\text{OPT}$, then $G_1(\text{SL}(\{(f_i, c_i, u_i)\}_{i \in F}, D))$ returns a $\frac{k+1}{k}$-approximate solution since by Inequality (4) in the proof of Theorem 3 we have shown that G_1 returns a solution of cost at most
$$\text{OPT} + f_p \leq \left(1 + \frac{1}{k}\right) \text{OPT}.$$
2. If $f_p > \frac{1}{k}\text{OPT}$, then $\text{OPT} - f_p < \frac{k-1}{k}\text{OPT}$. Let $\tilde{p}$ denote the facility p with no opening cost, i.e. $\tilde{p} = (0, c_p, u_p)$. Let OPT' be the cost of an optimal solution to the modified instance $\text{SL}(\{(f_i, c_i, u_i)\}_{i \in (F \setminus \{p\}) \cup \{\tilde{p}\}}, D)$; because (S^*, σ^*) is feasible for the modified instance, and costs f_p less in the modified instance (since we changed the cost of facility p to 0), we have that $\text{OPT}' \leq \text{OPT} - f_p$. By induction, Algorithm $G_{k-1}(\text{SL}(\{(f_i, c_i, u_i)\}_{i \in (F \setminus \{p\}) \cup \{\tilde{p}\}}, D))$ returns a solution $(\tilde{S}, \tilde{\sigma})$ such that $c(\tilde{S}, \tilde{\sigma}) \leq \frac{k}{k-1}\text{OPT}' \leq \frac{k}{k-1}\text{OPT} - \frac{k}{k-1} f_p$. The algorithm converts $(\tilde{S}, \tilde{\sigma})$ into a solution (S, σ) to the original instance $\text{SL}(\{(f_i, c_i, u_i)\}_{i \in F}, D)$ by replacing $\tilde{p}$ with p and setting $\sigma(p) = \tilde{\sigma}(\tilde{p})$ if $\tilde{p} \in \tilde{S}$; this replacement incurs an additional cost of f_p. Therefore
$$\begin{aligned} c(S, \sigma) \leq c(\tilde{S}, \tilde{\sigma}) + f_p &\leq \frac{k}{k-1}\left(\text{OPT} - f_p\right) + f_p \\ &= \text{OPT} + \frac{1}{k-1}(\text{OPT} - f_p) \\ &< \text{OPT} + \frac{1}{k}\text{OPT} = \frac{k+1}{k}\text{OPT}, \end{aligned}$$
where the final inequality uses that $\text{OPT} - f_p < \frac{k-1}{k}\text{OPT}$ as noted at the beginning of this case.

Since we must be in one of the two cases when p is the big facility of least index in S^*, and the algorithm finds solutions for both cases and returns the minimum-cost solution from the two cases, it must return a solution of cost at most $\frac{k+1}{k}\text{OPT}$.

Recall that G_1 has time complexity $O(n \log n)$. If G_{k-1} has time complexity $O(n^{k-1} \log n)$, then G_k has time complexity of $n \cdot O(n^{k-1} \log n)$ which is $O(n^k \log n)$. □

Acknowledgments

This research was sponsored in part by NSF grant CCF-1115256. We thank the referee for substantial comments that improved the presentation of the paper significantly.

References

[1] D. Bienstock, B. McClosky, Tightening simple mixed–integer sets with guaranteed bounds, Math. Program. 133 (1–2) (2012) 337–363. http://dx.doi.org/10.1007/s10107-010-0435-x.
[2] T. Carnes, D.B. Shmoys, Primal–dual schema for capacitated covering problems, Math. Program. 153 (2) (2015) 289–308. http://dx.doi.org/10.1007/s10107-014-0803-z.
[3] R.D. Carr, L.K. Fleischer, V.J. Leung, C.A. Phillips, Strengthening integrality gaps for capacitated network design and covering problems, in: Proceedings of the Eleventh Annual ACM–SIAM Symposium on Discrete Algorithms, in: SODA '00, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000, pp. 106–115. http://dl.acm.org/citation.cfm?id=338219.338241.
[4] J. Csirik, J.B.G. Frenk, M. Labbé, S. Zhang, Heuristics for the 0–1 min-knapsack problem, Acta Cybern. 10 (1–2) (1991) 15–20. http://dl.acm.org/citation.cfm?id=150385.150387.

[5] M. Florian, J.K. Lenstra, A.H.G. Rinnooy Kan, Deterministic production planning: Algorithms and complexity, Manage. Sci. 26 (1980) 669–679. http://pubsonline.informs.org/doi/pdf/10.1287/mnsc.26.7.669.
[6] N. Garg, J. Könemann, Faster and simpler algorithms for multicommodity flow and other fractional packing problems, SIAM J. Comput. 37 (2007) 630–652. http://dx.doi.org/10.1137/S0097539704446232.
[7] G.V. Gens, E.V. Levner, Computational complexity of approximation algorithms for combinatorial problems, in: Mathematical Foundations of Computer Science 1979, Springer, 1979, pp. 292–300.
[8] O.H. Ibarra, C.E. Kim, Fast approximation algorithms for the knapsack and sum of subset problems, J. ACM 22 (1975) 463–468. http://dx.doi.org/10.1145/321906.321909.
[9] R. Karp, Reducibility among combinatorial problems, in: R. Miller, J. Thatcher (Eds.), Complexity of Computer Computations, Plenum Press, 1972, pp. 85–103.
[10] C. Van Hoesel, A.P. Wagelmans, Fully polynomial approximation schemes for single–item capacitated economic lot–sizing problems, Math. Oper. Res. 26 (2) (2001) 339–357. http://dx.doi.org/10.1287/moor.26.2.339.10552.