Information Processing Letters 111 (2011) 545–549

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Approximation algorithms for the Fault-Tolerant Facility Placement problem

Li Yan *, Marek Chrobak

Department of Computer Science, University of California, Riverside, CA 92521, USA

ARTICLE INFO

Article history:
Received 5 August 2010
Received in revised form 2 March 2011
Accepted 5 March 2011
Available online 11 March 2011
Communicated by J. Torán

Keywords:
Approximation algorithms

ABSTRACT

In the *Fault-Tolerant Facility Placement* problem (FTFP) we are given a set of customers $\mathcal{C}$, a set of sites $\mathcal{F}$, and distances between customers and sites. We assume that the distances satisfy the triangle inequality. Each customer j has a demand r_j and each site may open an unbounded number of facilities. The objective is to minimize the total cost of opening facilities and connecting each customer j to r_j different open facilities. We present two LP-rounding algorithms for FTFP. The first algorithm achieves an approximation ratio of 4. The second algorithm uses the method of filtering to improve the ratio to 3.16.

Published by Elsevier B.V.

1. Introduction

The *Fault-Tolerant Facility Placement* problem (FTFP), that we study in this paper, is a generalization of the classical *Uncapacitated Facility Location* problem (UFL). In the UFL problem we wish to connect a collection of customers to facilities so as to minimize the total cost of connections and open facilities. The FTFP generalizes UFL by allowing customers to have arbitrary demands and allowing any number of facilities to be opened at each site, with the restriction that any two demands from any given customer have to be connected to different open facilities.

More specifically, in FTFP we are given a set of sites $\mathcal{F}$ and a set of customers $\mathcal{C}$. At each site $i \in \mathcal{F}$ we are allowed to open any number of facilities, each at cost $f_i \geqslant 0$. Each customer $j \in \mathcal{C}$ has an integral demand $r_j \geqslant 1$ which specifies the number of open facilities that j needs to be connected to. Some facilities that j is connected to could be located at the same site, as long as they are all different and open. For any pair $i \in \mathcal{F}$ and $j \in \mathcal{C}$ we are also given a distance $d_{ij} \geqslant 0$ between them that represents the cost to connect one unit of demand from customer j to a facility at site i. We assume the distance function to be metric, which means that it satisfies the following triangle inequality: $d_{ij} + d_{ij'} + d_{i'j'} \geqslant d_{i'j}$, for all $i, i' \in \mathcal{F}$ and $j, j' \in \mathcal{C}$. That is, connecting i' to j directly is cheaper than taking the detour via j' and i. Given such an instance, the objective is to minimize the total cost, defined as the sum of the facility opening costs and the connection costs. UFL is a special case of FTFP where $r_j = 1$ for each customer j. Another related problem is the *Fault-Tolerant Facility Location* problem (FTFL). Similar to FTFP, in FTFL each customer j has a demand r_j, but each site can open only one facility. One implication of this restriction is that every r_j is bounded by $|\mathcal{F}|$. In contrast, in FTFP the maximum r_j can be much larger than $|\mathcal{F}|$. Notice that FTFP can be reduced to FTFL by creating $R = \max_{j \in \mathcal{C}} r_j$ copies of each site. This reduction, however, creates an instance whose size depends linearly on R, so it does not lead directly to a polynomial-time approximation algorithm for FTFP.

The FTFP problem was introduced by Xu and Shen [18],[1] who gave a 1.861-approximation algorithm[2] based on the approach from [10]. Their algorithm, however, runs in time

* Corresponding author.
E-mail addresses: lyan@cs.ucr.edu (L. Yan), marek@cs.ucr.edu (M. Chrobak).

[1] In [18] this problem is called Fault-Tolerant Facility Allocation. We renamed it to avoid confusion with Fault-Tolerant Facility Location.

[2] The analysis in [18] is flawed, but the authors announced that it has been corrected in the new version of the paper that is forthcoming (private communication).

0020-0190/$ – see front matter Published by Elsevier B.V.
doi:10.1016/j.ipl.2011.03.005

proportional to R, so it is polynomial-time only in the restricted case when R is polynomial in $|\mathcal{F}|$.

Related work. The UFL problem has attracted great interest of researchers in both theoretical computer science and operations research. We now briefly review past work on UFL and related problems. Since we focus exclusively on the metric case, we omit the term "metric" in our discussion.

As shown by Guha and Khuller [6], UFL is MaxSNP-hard and it cannot be approximated to within ratio smaller than 1.463 unless the set cover problem can be approximated to $(1-\epsilon)\ln n$ for some constant $\epsilon > 0$, which, in turn, would imply $\mathsf{P} = \mathsf{NP}$ [12].

Several approximation algorithms for UFL are based on LP-rounding: solving the relaxation of the integer program for UFL and rounding the solution. The first such an algorithm with constant approximation ratio, of at most 3.16, was developed by Shmoys, Tardos and Aardal [14]. Using a different rounding approach, Chudak and Shmoys [5] gave an algorithm with ratio $1 + 2/e \approx 1.736$. This was further improved to 1.582 by Sviridenko [15]. Recently, Byrka [2] used the idea of scaling-up fractional solution and techniques from [11] to obtain another LP-rounding algorithm with currently best known ratio of 1.5.

There is also a number of approximation algorithms for UFL based on techniques that do not require solving the LP, including primal–dual methods [9,17], dual fitting [10], local search [1], or facility cost scaling and greedy augmentation [6,4].

The Fault-Tolerant Facility Location problem (FTFL) was introduced by Jain and Vazirani [8] who gave a primal–dual algorithm with approximation ratio $O(\log R)$. Their algorithm consists of R phases, with each phase completing in polynomial time, yielding overall polynomial time due to the $R \leqslant |\mathcal{F}|$ bound. Guha, Meyerson and Munagala [7] developed the first algorithm with constant approximation ratio, bounded by 2.41. This ratio was then improved to 2.076 by Swamy and Shmoys [16] and later by Byrka, Srinivasan and Swamy [3], who achieved the currently best known ratio of 1.7245 [3].

Our contribution. We present two polynomial time algorithms for FTFP based on LP-rounding. The first algorithm gives an approximation ratio of 4. Our approach generalizes the now classical LP-rounding algorithm for UFL, by Chudak and Shmoys [5,13]. The proof for bounding connection cost is obtained using a natural extension of the argument for UFL. Bounding facility cost is more challenging and it requires additional insights into the structure of the FTFP problem because, unlike in UFL, the algorithm does not produce a partition of the instance into disjoint clusters. The second algorithm uses an extra filtering step to improve the ratio to $3/(1-e^{-3}) \approx 3.16$, following the ideas from [14].

2. The LP-rounding algorithm

A natural integer program formulation of FTFP uses variables y_i to denote the number of facilities opened at site i, x_{ij} to denote the number of connections between site i and customer j, and expresses the constraints and the objective function using those variables. The two constraints are: (i) the number of connections from a customer to any given site cannot exceed the number of facilities opened at that site; and (ii) the total number of connections from each customer cannot be smaller than his demand. By relaxing the integral variables to take fractional values, we obtain the following linear program for FTFP.

$$\begin{aligned} \text{minimize} \quad & \sum_{i\in\mathcal{F}} f_i y_i + \sum_{i\in\mathcal{F}, j\in\mathcal{C}} d_{ij} x_{ij} \qquad (1)\\ \text{subject to} \quad & y_i - x_{ij} \geqslant 0 \quad \forall i \in \mathcal{F},\ j \in \mathcal{C} \\ & \sum_{i\in\mathcal{F}} x_{ij} \geqslant r_j \quad \forall j \in \mathcal{C} \\ & x_{ij} \geqslant 0,\ y_i \geqslant 0 \quad \forall i \in \mathcal{F},\ j \in \mathcal{C}. \end{aligned}$$

The dual program is:

$$\begin{aligned} \text{maximize} \quad & \sum_{j\in\mathcal{C}} r_j \alpha_j \qquad (2)\\ \text{subject to} \quad & \sum_{j\in\mathcal{C}} \beta_{ij} \leqslant f_i \quad \forall i \in \mathcal{F} \\ & \alpha_j - \beta_{ij} \leqslant d_{ij} \quad \forall i \in \mathcal{F},\ j \in \mathcal{C} \\ & \alpha_j \geqslant 0,\ \beta_{ij} \geqslant 0 \quad \forall i \in \mathcal{F},\ j \in \mathcal{C}. \end{aligned}$$

Denoting the optimal fractional solutions to (1) and (2) as $(\mathbf{x}^*, \mathbf{y}^*)$ and $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$, respectively, we have the following fact that follows from complementary slackness conditions.

Fact 2.1. *$\alpha_j^* \geqslant d_{ij}$ for all $i \in \mathcal{F}$, $j \in \mathcal{C}$ such that $x_{ij}^* > 0$.*

2.1. Algorithm LPR

The algorithm works by first solving the LP's (1) and (2) to obtain optimal primal and dual fractional solutions $(\mathbf{x}^*, \mathbf{y}^*)$ and $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$. The algorithm starts with the empty (infeasible) solution, with no facilities open and no connections, and then proceeds in rounds, gradually opening facilities and creating new connections. In each round the algorithm identifies a customer l with minimum α_l^* among all not-fully-connected customers. For every customer j, let s_j be the yet unsatisfied (residual) demand of j at the beginning of this round, and define its *neighborhood* to be $N(j) = \{i \in \mathcal{F}\colon\ x_{ij}^* > 0\}$. The algorithm then picks a site k with minimum opening cost out of the neighborhood $N(l)$, and opens s_l facilities at site k. The s_l demands of customer l are then fully served by these new facilities. For all other not-fully-connected customers whose neighborhoods intersect $N(l)$, their residual demands are served up to s_l by the new facilities opened at site k (Fig. 1).

2.2. Analysis

We now show that Algorithm LPR is a 4-approximation algorithm for FTFP. It is easy to see that the solution $(\mathbf{x}, \mathbf{y})$ computed by the algorithm is feasible. To bound the cost,