Algorithms for the Data Placement Problem

Chaitanya Swamy*

1 Introduction

We consider the *data placement problem* introduced by Baev & Rajaraman [1]. We have a set of caches, $\mathcal{F}$, a set of data objects $\mathcal{O}$, and a set of clients $\mathcal{D}$. Each object $s \in \mathcal{O}$ has a *length* l_s and a cache $i \in \mathcal{F}$ has capacity u_i that limits the total length of data objects that may be stored in the cache. Further, each client $j \in \mathcal{D}$ has demand d_j for a specific data object $s(j) \in \mathcal{O}$ and has to be assigned to a cache that stores that object. Storing an object s in cache i incurs a *storage cost* of f_i^s and assigning client j to cache i incurs an *access cost* of $d_j l_{s(j)} c_{ij}$ proportional to the distance c_{ij} between i and j. The data placement problem seeks a placement of the data objects to caches that respects cache capacities and an assignment of clients to caches, so as to minimize the total storage and client access costs. More precisely, for each cache i in $\mathcal{F}$ we want to determine the set of objects $O(i) \subseteq \mathcal{O}$ it stores such that $\sum_{s \in O(i)} l_s \leq u_i$ and assign each client j to a cache $i(j)$ that stores object $s(j)$, i.e., $s(j) \in O(i)$, and we want to minimize $\sum_{i \in \mathcal{F}} \sum_{s \in O(i)} f_i^s + \sum_{j \in \mathcal{D}} d_j l_{s(j)} c_{i(j)j}$. We assume that the caches and clients are located in a common metric space, so the distances c_{ij} form a metric.

The data placement problem is a generalization of the metric *uncapacitated facility location* (UFL) problem and is *NP*-hard even when all object lengths are equal. We give a 10-approximation algorithm for the problem when all objects have the same length, improving upon the approximation guarantee of 20.5 given by Baev & Rajaraman [1]. Our improvement comes from an improved rounding procedure for a natural LP relaxation of the problem also considered in [1]. As in [1], we can modify the algorithm get a bicriteria approximation guarantee when objects have different lengths; the placement returned has cost at most 10 times the optimal, but the total length of objects stored in a cache may exceed the cache capacity by the maximum object length. We also extend the algorithm to the k-median variant, where there is a bound k_s imposed, for every object s, on the number of caches that may store object s.

As stated in [1], it is not hard to show via a reduction from the PARTITION problem, that with arbitrary object lengths, it is *NP*-complete to even decide if there is a feasible solution and hence no approximation ratio is achievable in polynomial-time unless *P=NP*. [1] showed that the problem is *MAXSNP*-hard even if all objects have the same length and there are no storage costs, by reducing metric UFL to the problem, and gave a 20.5-approximation algorithm. In their model, client j has demand d_{js} for every object $s \in \mathcal{O}$ and incurs a corresponding access cost; however this easily reduces to our model since for every object s, we can simply create a copy $j^{(s)}$ with demand d_{js} (for object s).

Related Work. Shmoys, Swamy & Levi [10] consider a closely related problem called *facility location with service installation costs*, where the caches are uncapacitated but one has to pay a location-dependent cache-setup cost to build/setup a cache at location i before any data object may be stored in the cache. They motivate the problem from a facility location perspective, where the caches correspond to facilities, and the objects correspond to different *services* that the clients require. To satisfy a client one has to assign it to an

*swamy@cs.cornell.edu. Dept. of Computer Science, Cornell University, Ithaca, NY 14853. Research supported partially by NSF grant CCR-9912422.

open facility (cache) on which the service (data object) requested by the client is installed (stored), and the cost incurred is the sum of the facility opening costs (cache setup costs), service installation costs (storage costs) and client assignment costs (access costs). Shmoys et al. [10] give a 6-approximation algorithm for this problem under a certain assumption on the service installation costs. Guha & Munagala [4] consider a generalization of the problem where a cache also has a client capacity limiting the number of clients that may be assigned to the cache. They however only obtain bicriteria results where both the client and object cache capacities may be violated. Their algorithm is based on rounding a natural LP relaxation that has an unbounded integrality gap for hard client cache capacities. Hence such an approach can only yield bicriteria results for client cache capacities.

The data placement problem is a generalization of UFL — if there is just one object then this is simply the uncapacitated facility location problem. There is a large body of literature that deals with designing approximation algorithms for metric UFL and we sample only a few results below; see [9] for a survey of this and earlier work. Shmoys, Tardos & Aardal [11] gave the first constant-factor approximation algorithm for this problem using the *filtering* technique of Lin & Vitter [7] to round the optimal solution of a linear program. Chudak & Shmoys [3] gave a rounding procedure that used information from the dual LP to give a $\left(1+\frac{2}{e}\right)$-approximation algorithm. The ratio was further improved by Sviridenko [12] to 1.58. Jain & Vazirani [6] gave a combinatorial *primal-dual* 3-approximation algorithm where the LP is used only in the analysis. The current best ratio for UFL is 1.52 [8] obtained by building upon a dual-fitting based greedy algorithm of [5]. For the closely related k-median problem, the first constant-factor approximation algorithm was given by Charikar, Guha, Tardos & Shmoys [2] using LP rounding. The techniques developed by them play an important role in our algorithm (and in the algorithm of [1]).

2 An LP relaxation

We can express the data placement problem as an integer program and relax the integrality constraints to get a linear program. Throughout we will use i to index the caches in $\mathcal{F}$, j to index the clients in $\mathcal{D}$ and s to index the objects in $\mathcal{O}$. We focus on the case when all objects have the same length and assume that $l_s = 1$ for every object s without loss of generality.

$$\begin{aligned}
\min \quad & \sum_i \sum_s f_i^s y_i^s + \sum_j \sum_i d_j c_{ij} x_{ij} && && \text{(P)}\\
\text{s.t.} \quad & \sum_i x_{ij} \geq 1 && \forall j && \\
& x_{ij} \leq y_i^{s(j)} && \forall i,j && \\
& \sum_s y_i^s \leq u_i && \forall i && (1)\\
& x_{ij}, y_i^s \geq 0 && \forall i,j,s. &&
\end{aligned}$$

Variable y_i^s indicates if object s is stored in cache i and x_{ij} indicates if client j is assigned to cache i. The first and second constraints say that each client must be assigned to a cache and if client j is assigned to cache i then object $s(j)$ must be stored in cache i. The third constraint states that the total length of items stored in any cache i is at most its capacity u_i. This LP is the same as the LP relaxation in [1]. An integer solution corresponds exactly to a solution to our problem. We let G_s denote the set of clients that demand object s, i.e., $G_s = \{j : s(j) = s\}$.

3 The rounding procedure

Let (x, y) denote the optimal solution to (P) and OPT be its value. We will round this to an integer solution losing a factor of at most 10. We use the terms access cost and assignment cost interchangeably.

3.1 Overview of the algorithm

We first give a high level description of the algorithm. Suppose for a moment that the optimal solution (x, y) satisfies the following property: for any cache i and object s, there is *at most one* client $j \in G_s$ such that $x_{ij} > 0$ $(*)$. We can then setup the following min-cost flow problem: create a bipartite graph with vertex set $\mathcal{D} \cup \mathcal{F}$ and edges (i, j) for every i, j such that $x_{ij} > 0$ with cost $d_j c_{ij} + f_i^{s(j)}$ and capacity 1; client j has a demand of 1 and cache i has capacity u_i. The LP solution translates to a feasible fractional flow in this graph of cost at most OPT. *Note that property* $(*)$ *is crucial for this*. Conversely an integer flow yields an integer solution to (P) of cost equal to the flow cost. Therefore by the integrality property of flows (given integer capacities) we can round (x, y) to an integer solution of no greater cost. Of course, the LP solution need not have property $(*)$ so our goal will be to transform (x, y) to a solution that has this property without increasing the cost by much.

Roughly speaking we want to do the following: for each object s, cluster the clients in G_s around certain 'centers' (also clients in G_s) such that (a) every client k is assigned to a "nearby" cluster center j whose LP assignment cost is less than that of k, and (b) the facilities serving the cluster centers in the fractional solution (x, y) are disjoint. So, the modified instance where the demand of a client is moved to the center of its cluster has a fractional solution, namely the solution induced by (x, y), that satisfies $(*)$ and has cost at most OPT. Furthermore, given a solution to the modified instance we can obtain a solution to the original instance losing a small additive factor. One option is to use the decomposition method of Shmoys et al. [11] that produces precisely such a clustering. The problem however is that [11] uses filtering which involves blowing up the x_{ij} values, thus violating the cache capacities. Chudak & Shmoys [3] use the same clustering idea but without filtering, using the dual solution to bound the cost. The difficulty here in bounding the cost using the dual solution is that there are terms with negative coefficients in the dual objective function that correspond to the primal constraints (1). Although [13] showed that it is possible to overcome this difficulty in certain cases, the situation here looks more complicated and it is not clear how to use their techniques.

Instead, we use the clustering technique of Charikar et al. [2] to cluster clients and first obtain a *half-integral solution* $(\hat{x}, \hat{y})$, that is, every $\hat{x}_{ij}, \hat{y}_i \in \left\{0, \frac{1}{2}, 1\right\}$, to the modified instance with cluster centers, losing a factor of 3. Further, any solution here will give a solution to the original instance while increasing the cost by at most $4 \cdot OPT$. Now we use the clustering method of [11] *without any filtering*, since the half-integral solution $(\hat{x}, \hat{y})$ is essentially already filtered; if client j is assigned to i and i' in $\hat{x}$, then $c_{ij}, c_{i'j} \leq 2(c_{ij}\hat{x}_{ij} + c_{i'j}\hat{x}_{i'j})$. This final step causes us to lose an additive factor equal to the cost of $(\hat{x}, \hat{y})$, so overall we get an approximation ratio of 4+3+3=10.

We now describe each of these steps in detail. Let $\bar{C}_j = \sum_i c_{ij} x_{ij}$ denote the cost incurred by the LP solution to assign one unit of demand of client j.

3.2 Obtaining a half-integral solution $(\hat{x}, \hat{y})$

Step I: Consolidating demands around centers. We first consider every object s separately, and consolidate (or cluster) the demand of clients in G_s at certain clients, that we call *cluster centers*. We do not modify the fractional solution (x, y) but only modify the demands so that for some clients j, the demand d_j is "moved" to a "nearby" center k. We assume every client has a non-zero demand (otherwise we can simply get rid of such clients).

Set $d'_j \leftarrow 0$ for every j. Consider the clients in G_s in increasing order of $\bar{C}_j$. For each client j, if there exists a client k such that $d'_k > 0$ and $c_{jk} < 4\max(\bar{C}_j, \bar{C}_k) = 4\bar{C}_j$, set $d'_k \leftarrow d'_k + d_j$, otherwise set $d'_j \leftarrow d_j$. We do this for every object s. Let $D_s = \{j \in G_s : d'_j > 0\}$ and $D = \bigcup_s D_s$. Each client in D is a cluster center. Let $OPT' = \sum_{i,s} f_i^s y_i^s + \sum_{j \in D,i} d'_j c_{ij} x_{ij}$ denote the cost of (x, y) for the modified instance consisting of the cluster centers.

Lemma 3.1 *The following hold: (i) if $j, k \in D_s$, then $c_{jk} \geq 4\max(\bar{C}_j, \bar{C}_k)$, (ii) $OPT' \leq OPT$, and (iii) any solution (x', y') to the modified instance can be converted to a solution to the original instance incurring an additional cost of at most $4 \cdot OPT$.*

Proof : Suppose j was considered after k. Then $d'_k > 0$ at this time, otherwise d'_k would remain at 0 and k would not be in D_s. So if $c_{jk} < 4\max(\bar{C}_j, \bar{C}_k)$ then d'_j would remain at 0, giving a contradiction. It is clear that if we move the demand of client j to client k, then $\bar{C}_j \leq \bar{C}_k$ and $c_{jk} \leq 4\bar{C}_k$. So the assignment cost for the new instance, $\sum_j d'_j C_j$, only decreases and the storage cost $\sum_{i,s} f_i^s y_i^s$ does not change, hence $OPT' \leq OPT$. Given a solution (x', y') to the modified instance, if the demand of k was moved to j the extra cost incurred in assigning k to the same facility(ies) as in x' is at most $d_k c_{jk} \leq 4 d_k \bar{C}_k$ by the triangle inequality, so the total extra cost is at most $4 \cdot OPT$. ■

From now on we will focus on the modified instance with client set D and modified demands d'_j. At the very end we will use the above Lemma to translate an integer solution to the modified instance to an integer solution to the original instance

Step II: Transforming to a half-integral solution. We define the cluster of a client $j \in D_s$ to consist of all clients $k \in G_s$ whose demand d_k was moved moved to j, and a set of facilities F_j. F_j consists of all facilities i to which j is fractionally assigned such that j, is the center in D_s closest to i, that is, $F_j = \{i : x_{ij} > 0 \text{ and } c_{ij} = \min_{k \in D_s} c_{ik}\}$, with ties broken arbitrarily. Let $F'_j \subseteq F_j = \{i \in F_j : c_{ij} \leq 2\bar{C}_j\}$. Define γ_j to be $\min_{i \notin F_j : x_{ij} > 0} c_{ij}$. Clearly the sets F_j for $j \in D_s$ are disjoint. By property (i) of Lemma 3.1, we have that F_j contains all the facilities i such that $x_{ij} > 0$ and $c_{ij} \leq 2\bar{C}_j$. So $\sum_{i \in F'_j} x_{ij} = \sum_{i : c_{ij} \leq 2\bar{C}_j} x_{ij} \geq \frac{1}{2}$ where the last inequality follows from Markov's inequality.

In the half-integral solution $(\hat{x}, \hat{y})$, we will store object s only at caches that lie in some set F_j for $j \in D_s$. To obtain $(\hat{x}, \hat{y})$, we setup a min-cost flow problem. We create a sink t and a node r_i for every cache i in $\bigcup_{j \in D} F_j$ with an outgoing edge (r_i, t) of capacity u_i and cost 0. For each client $j \in D$ we create four nodes v_j, w_j, a_j, and b_j. Node v_j has demand -1 (i.e., the net outgoing flow should be 1) and has edges (v_j, a_j) of capacity 1 and cost 0, and (v_j, w_j) of capacity $\frac{1}{2}$ and cost 0. Node w_j has edges (w_j, b_j) of capacity $\frac{1}{2}$ and cost 0, and (w_j, t) of capacity $\frac{1}{2}$ and cost $3d'_j\gamma_j$. Node a_j has edges (a_j, r_i) to every $i \in F'_j$ of capacity 1 and cost $d'_j c_{ij} + f_i^{s(j)}$. Node b_j has edges (b_j, r_i) to every $i \in F_j \setminus F'_j$ of capacity 1 and cost $d'_j c_{ij} + f_i^{s(j)}$. Since all edge capacities are $\frac{1}{2}$ or 1 the network has a half-integral min-cost flow.

Given such a flow we obtain $(\hat{x}, \hat{y})$ as follows. Consider object s. For every $j \in D_s$ and cache $i \in F'_j$, we set $\hat{y}_i^s = \hat{x}_{ij} =$ flow along edge (a_j, r_i). Similarly for every $i \in F_j \setminus F'_j$, we set $\hat{y}_i^s = \hat{x}_{ij}$ equal to the flow along edge (b_j, r_i). Observe that there is at least one cache $i \in F'_j$ such that $x_{ij} > 0$; we call the cache in F'_j closest to j with $x_{ij} > 0$ the *primary cache* of j. Note that since the sets F_j (and hence F'_j) for $j \in D_s$ are disjoint, every client in D_s has a unique primary cache i. If edge (w_j, t) carries positive flow (implying that edge (w_j, b_j) carries 0 flow, so no facility from $F_j \setminus F'_j$ is open), let i' be the cache nearest to j, other than its primary cache, with $y_{i'}^s > 0$; we set $\hat{x}_{i'j} =$ flow on $(w_j, t) = \frac{1}{2}$. We do this for every object s. All other $\hat{x}_{ij}, \hat{y}_i^s$ are set to 0. If a client j is assigned to a cache other than its primary cache, we call the other cache the *secondary cache* of j. It is easy to verify that $(\hat{x}, \hat{y})$ is a feasible solution to (P) where we only have clients in D. The following lemma shows that the cost of $(\hat{x}, \hat{y})$ is at most $3 \cdot OPT$.

Lemma 3.2 *The cost of* $(\hat{x}, \hat{y})$*, that is,* $\sum_{i,s} f_i^s \hat{y}_i^s + \sum_{j \in D, i} d'_j c_{ij} \hat{x}_{ij}$*, is at most* $3 \cdot OPT' \leq 3 \cdot OPT$.

Proof : First we show that (x, y) induces a flow of cost at most $3 \cdot OPT'$, so the cost of the min-cost flow is no greater. Then we show that the cost of $(\hat{x}, \hat{y})$ is bounded by the cost of the min-cost flow.

Consider the following flow: each edge (v_j, a_j) has flow $\sum_{i \in F'_j} x_{ij}$, (v_j, w_j) has flow $1 - \sum_{i \in F'_j} x_{ij}$; edge (w_j, b_j) has flow $\sum_{i \in F_j \setminus F'_j} x_{ij}$ and (w_j, t) has flow $1 - \sum_{i \in F_j} x_{ij}$; every edge (a_j, r_i) or (b_j, r_i) has flow x_{ij}; the flow on (r_i, t) is $\sum_s \sum_{j \in D_s : i \in F_j} x_{ij}$. This is a feasible flow since $\sum_{i \in F'_j} x_{ij} \geq \frac{1}{2}$, and for any i, s there is at most one $j \in D_s$ with $i \in F_j$, so $\sum_{j \in D_s : i \in F_j} x_{ij} \leq y_i^s$. The cost of this flow is, $\sum_{s, j \in D_s} \Big(\sum_{i \in F_j} (d'_j c_{ij} + f_i^s) x_{ij} + 3 d'_j \gamma_j (1 - \sum_{i \in F_j} x_{ij}) \Big)$ which is at most,

$$\sum_{i,s} f_i^s y_i^s + \sum_j d'_j \Big(\sum_{i \in F_j} c_{ij} x_{ij} + 3\gamma_j (1 - \sum_{i \in F_j} x_{ij}) \Big).$$

We have $OPT' = \sum_{i,s} f_i^s y_i^s + \sum_j d'_j \bar{C}_j$. For any $j \in D$, $\bar{C}_j = \sum_{i \in F_j} c_{ij} x_{ij} + \sum_{i \notin F_j} c_{ij} x_{ij} \geq \sum_{i \in F_j} c_{ij} x_{ij} + \gamma_j (1 - \sum_{i \in F_j} x_{ij})$ since γ_j was defined as $\min_{i \notin F_j : x_{ij} > 0} c_{ij}$. This shows that the cost of the constructed flow, and hence of the min-cost flow, is at most $3 \cdot OPT'$.

Now consider the solution $(\hat{x}, \hat{y})$ induced by the half-integral min-cost flow. By construction, the quantity $\sum_{i,s} f_i^s \hat{y}_i^s + \sum_{j \in D, i \in F_j} d'_j c_{ij} \hat{x}_{ij}$ is exactly equal to the total cost of the flow on edges (a_j, r_i) and (b_j, r_i). For any $j \in D$ the remaining cost $\sum_{i \notin F_j} d'_j c_{ij} \hat{x}_{ij}$ is equal to $d'_j c_{i'j}$(flow on (w_j, t)) where i' is the secondary cache of j. So it suffices to show that $c_{i'j} \leq 3\gamma_j$. Let $\gamma_j = c_{i''j}$ where $i'' \notin F_j$ and $x_{i''j} > 0$. Let k be the center in D_s nearest to i'' and let ℓ be the primary cache of k. Then, $c_{i'j} \leq c_{\ell j}$ and $4 \max(\bar{C}_j, \bar{C}_k) \leq c_{jk} \leq c_{i''j} + c_{i''k} \leq 2\gamma_j$. Also $c_{\ell k} \leq 2\bar{C}_k$ since $\ell \in F'_k$. Combining the inequalities we get that $c_{i'j} \leq 3\gamma_j$ which completes the proof of the lemma. ■

3.3 Converting $(\hat{x}, \hat{y})$ to an integer solution

Define $\hat{C}_j = \sum_i c_{ij} \hat{x}_{ij}$ for $j \in D$. Let $i_1(j)$ denote the primary cache of j. For convenience, we will say that every client $j \in D$ has both a primary cache $i_1(j)$ and a secondary cache i' with $\hat{x}_{i_1(j)j} = \hat{x}_{i'j} = \frac{1}{2}$, with the understanding that if j does not have a secondary cache then $i' = i_1(j)$ and $\hat{x}_{i_1(j)j} = 1$. We denote the secondary cache by $i_2(j)$. Then we have, $\hat{C}_j = \frac{1}{2}(c_{i_1(j)j} + c_{i_2(j)j})$, $c_{i_1(j)j} \leq \hat{C}_j$ and $c_{i_1(j)j} \leq c_{i_2(j)j} \leq 2\hat{C}_j$. Let $L_s = \{i : \hat{y}_i^s > 0\}$ and $L = \bigcup_s L_s$.

Step III: Clustering. First for every object s we cluster the clients in D_s as follows: pick $j \in D_s$ with smallest $\hat{C}_j$. Remove every client $k \in D_s$ such that both j and k are (fractionally) assigned to a cache $i \in L_s$, and recurse on the remaining set of clients until no client in D_s is left. Let D'_s be the set of clients picked for object s and $D' = \bigcup_s D'_s$ — these are the *new cluster centers*. It is clear that for any cache in L_s at most one client in D'_s is assigned to it. Observe that for every client $k \in D_s \setminus D'_s$ there is some $j \in D'_s$ such that $\hat{C}_j \leq \hat{C}_k$ and $\exists i \in L_s$ such that $\hat{x}_{ij}, \hat{x}_{ik} > 0$ which implies $c_{jk} \leq 4\hat{C}_k$. We call j the *center* of k and denote it by $\mathsf{ctr}(k)$.

So for every client $k \in D \setminus D'$ we can move the demand d'_k to $j = \mathsf{ctr}(k)$, set up a min-cost flow problem to get an integer solution to the instance with the new cluster centers and the new demands, and translate this to a solution to the instance with client set D losing an additive factor of at most $4 \sum_{k \in D} d'_k \hat{C}_k$. We will set up the min-cost flow network more carefully so that we only lose an additive factor of $\sum_{k \in D} d'_k \hat{C}_k$, that is, the cost of the integer solution on the client set D is at most twice the cost of $(\hat{x}, \hat{y})$. We want to capture the following observation: suppose the demand of $k \in D_s$ was moved to $j \in D'_s$. Let i, i' be the primary and secondary caches of k and suppose that i'', i are the primary, secondary caches of j. Object s

is stored in i'' or i in the integer solution (flow), so the per unit demand assignment cost of k is either c_{ik} or $c_{i''k}$. But the average of these two using the weights $\hat{x}_{ij} = \hat{x}_{i''j} = \frac{1}{2}$ assigned by the fractional solution (flow), is at most $\frac{1}{2}(c_{ik} + c_{i''k}) \leq c_{ik} + \hat{C}_j \leq 2\hat{C}_k$ since $c_{i''k} \leq c_{i''j} + c_{ij} + c_{ik}$, so the total cost is only $\sum_{i,s} f_i^s \hat{y}_i^s + 2\sum_{j \in D} d'_j \hat{C}_j$.

Step IV: The min-cost flow network. Fix $s \in \mathcal{O}$ and consider a client $j \in D'_s$. We will maintain two sets A_j and B_j for j. Let $i = i_1(j)$ and $i' = i_2(j)$ be the primary and secondary caches of j. We define $A_j = \{k \in D_s : \mathsf{ctr}(k) = j\}$, and $B_j = \{k \in D_s : \mathsf{ctr}(k) \neq j \text{ and } i' = i_1(k)\}$. We also have a set B_i^s for every cache $i \in L_s$ such that $\hat{x}_{ij} = 0$ *for every* $j \in D'_s$. Define $B_i^s = \{k \in D_s : i = i_1(k)\}$. Note that all the sets A_j, B_j and B_i^s are subsets of $D_s \setminus D'_s$.

We create a sink t and a node r_i for every $i \in L$ such that $\exists j \in D', \hat{x}_{ij} > 0$ or $\exists s, B_i^s \neq \phi$. We have an edge (r_i, t) of capacity u_i and cost 0. For every client $j \in D'$ we create a node v_j. Further, for every $i \in L, s \in \mathcal{O}$ with $B_i^s \neq \phi$ we create a node w_i^s. The nodes v_j and w_i^s all have demand -1. For every node v_j we have edges (v_j, r_i) to every i with $\hat{x}_{ij} > 0$, and we have edges (w_i^s, r_i), (w_i^s, t) for every node w_i^s. All these edges have capacity 1. The cost of these edges is set as follows. Consider a node v_j and let $i = i_1(j), i' = i_2(j)$. We set the cost of (v_j, r_i) to $f_i^{s(j)} + d'_j c_{ij} + \sum_{k \in A_j} d'_k c_{ik}$ and the cost of $(v_j, r_{i'})$ to $f_{i'}^{s(j)} + d'_j c_{i'j} + \sum_{k \in A_j} d'_k c_{i'k} + \sum_{k \in B_j} d'_k (c_{i'k} - c_{i_2(k)k})$. Note that the quantity $c_{i'k} - c_{i_2(k)k}$ is actually negative since $i' = i_1(k)$ for $k \in B_j$. We set the cost of (w_i^s, r_i) to $f_i^s + \sum_{k \in B_i^s} d'_k (c_{ik} - c_{i_2(k)k})$ and the cost of (w_i^s, t) to 0.

Since the capacities are all integer, there is an integer min-cost flow. We map this to an integer solution $(\tilde{x}, \tilde{y})$ to the instance with client set D. Set $\tilde{x}_{ij}, \tilde{y}_i \leftarrow 0$ for all i, j. Consider object s. First, for every $j \in D'_s$ we set $\tilde{x}_{i_1(j)j} = 1$ if edge $(v_j, r_{i_1(j)})$ carries non-zero flow, and $\tilde{x}_{i_2(j)j} = 1$ otherwise. For every client $k \in B_j$ we set $\tilde{x}_{i_2(j)k} = \tilde{x}_{i_2(j)j}$, and for every $k \in B_i^s$ we set $\tilde{x}_{ik} =$ flow on(w_i^s, r_i). Next, for every $j \in D'_s$ and every $k \in A_j$ that has not yet been assigned, i.e., $\sum_i \tilde{x}_{ik} = 0$, we set $\tilde{x}_{i_1(j)k} = 1$ if edge $(v_j, r_{i_1(j)})$ carries non-zero flow, and $\tilde{x}_{i_2(j)k} = 1$ otherwise. Finally, set $\tilde{y}_i^s = \max_{j \in D_s} \tilde{x}_{ij}$. We do this for every s. $(\tilde{x}, \tilde{y})$ is feasible since for every i, s such that $\tilde{y}_i^s = 1$ we send a unit of flow along the edge (r_i, t). So constraints (1) are satisfied and the other constraints are clearly satisfied. We now bound the cost of $(\tilde{x}, \tilde{y})$.

Lemma 3.3 *The cost of the min. cost flow in the network is at most twice the cost of* $(\hat{x}, \hat{y})$.

Proof : We exhibit a fractional flow of cost at most the claimed cost.

The fractional flow is obtained by simply setting the flow on every edge (v_j, r_i) to $\hat{x}_{ij}$ and the flow on (w_i^s, t) and (w_i^s, r_i) to $\max_{k \in B_i^s} \hat{x}_{ik} = \frac{1}{2}$ where the equality follows since every $k \in B_i^s$ is assigned to an extent of $\frac{1}{2}$ to $i_2(k) \neq i$. The flow on the edges (r_i, t) is set accordingly to $\sum_s \left(\sum_{j \in D'_s} \hat{x}_{ij} + \max_{k \in B_i^s} \hat{x}_{ik}\right)$. This is a feasible flow since for every i, s, either $B_i^s = \phi$ and there is exactly one $j \in D'_s$ such that $\hat{x}_{ij} > 0$, or $B_i^s \neq \phi$ and $\hat{x}_{ij} = 0$ for every $j \in D'_s$. So $\sum_{j \in D'_s} \hat{x}_{ij} + \max_{k \in B_i^s} \hat{x}_{ik}$ is at most y_i^s.

The cost of an edge (v_j, r_i) or (w_i^s, r_i) consists of a storage component ($f_i^{s(j)}$ or f_i^s) and an assignment component that can be attributed to various clients. We call the contribution of the storage components to the flow cost the *flow storage cost*, and the contribution of the assignment components the *flow assignment cost*. The flow storage cost is $\sum_{i,s} f_i^s \left(\sum_{j \in D'_s} \hat{x}_{ij} + \max_{k \in B_i^s} \hat{x}_{ik}\right) \leq \sum_{i,s} f_i^s y_i^s$ by the above reasoning. To evaluate the flow assignment cost we consider the contribution of each client to the assignment components separately. Fix an object s. First consider $j \in D'_s$ with $i = i_1(j),\ i' = i_2(j)$. Client j only figures in the assignment component of (v_j, r_i) and $(v_j, r_{i'})$ and its contribution is $d'_j(c_{ij}\hat{x}_{ij} + c_{i'j}\hat{x}_{i'j}) = d'_j \hat{C}_j$. A client $k \in D_s \setminus D'_s$ is in exactly one set A_j where $j = \mathsf{ctr}(k)$ and may possibly also lie in one of the sets $B_{j'}$ or $B_{i''}^s$. Let $i = i_1(j)$ and $i' = i_2(j)$.

1. If k does not lie in any set $B_{j'}$ or $B^s_{i''}$, then it must be that $\hat{x}_{i_1(k)j} > 0$. This implies that $i_1(k) = i'$ since $i_1(k) \neq i$, as the primary caches of clients in D_s are distinct. Client k contributes only to the assignment component of edges (v_j, r_i) and $(v_j, r_{i'})$ and this contribution is $d'_k(c_{ik}\hat{x}_{ij} + c_{i'k}\hat{x}_{i'j}) \leq d'_k(c_{i'k} + \hat{C}_j) \leq 2d'_k\hat{C}_k$ since $\hat{x}_{ij} = \hat{x}_{i'j} = \frac{1}{2}$ and $c_{ik} \leq c_{i'k} + c_{i'j} + c_{ij}$.
2. Now suppose k is also in one of the sets $B_{j'}$ or $B^s_{i''}$, so it also contributes to the assignment component of an edge $(v_{j'}, r_{i_1(k)})$ or an edge $(w^s_{i''}, r_{i''})$. The contribution in both cases is $\frac{d'_k}{2}(c_{i_1(k)k} - c_{i_2(k)k})$ since we must have $x_{i_1(k)j'} = \frac{1}{2} = x_{i''k}$. Adding the contributions to edges (v_j, r_i) and $(v_j, r_{i'})$, the total contribution is $\frac{d'_k}{2}(c_{ik} + c_{i'k} + c_{i_1(k)k} - c_{i_2(k)k}) \leq d'_k(\hat{C}_k + \hat{C}_j) \leq 2d'_k\hat{C}_k$ since $c_{ik} + c_{i'k} \leq 2c_{i_2(k)k} + c_{ij} + c_{i'j}$.

So the flow assignment cost is at most $2\sum_{j\in D} d'_j\hat{C}_j$. Thus the total flow cost is at most $\sum_{i,s} f^s_i\hat{y}^s_i + 2\sum_{j\in D} d'_j\hat{C}_j$ which is at most twice the cost of $(\hat{x}, \hat{y})$. ■

Lemma 3.4 *The cost of the integer solution $(\tilde{x}, \tilde{y})$ is at most the cost of the min-cost integer flow.*

Proof : Observe that for any s, $\sum_i f^s_i\tilde{y}^s_i = \sum_{j\in D'_s,i} f^s_i\tilde{x}_{ij} + \sum_{\text{nodes } w^s_i} f^s_i(\text{flow on } (w^s_i, r_i))$. So the total storage cost is $\sum_{e=(v_j,r_i)} f^{s(j)}_i(\text{flow on } e) + \sum_{e=(w^s_i,r_i)} f^s_i(\text{flow on } e)$ which is just the flow storage cost.

We will bound the assignment cost of a client by the contribution it makes to the flow assignment cost. Fix object s. Consider $j \in D'_s$. Let $i = i_1(j)$ and $i' = i_2(j)$. At most one of the edges (v_j, r_i), $(v_j, r_{i'})$ carries non-zero flow and we set $\tilde{x}_{ij}, \tilde{x}_{i'j}$ equal to the flow on the corresponding edge. So the assignment cost of j is $d'_j\big(c_{ij}(\text{flow on } (v_j, r_i)) + c_{i'j}(\text{flow on } (v_j, r_{i'}))\big)$, which is also the contribution of j to the assignment flow cost. The same argument holds for $k \in A_j$ if k is assigned to one of i or i'. The remaining case is when $k \in A_j$, and k is not assigned to i or i', but it is assigned to $i'' = i_1(k)$ either because $k \in B_{j'}$ where $i'' = i_2(j)$ and $(v_{j'}, r_{i''})$ has non-zero flow, or because $k \in B^s_{i''}$ and $(w^s_{i''}, r_{i''})$ carries non-zero flow. The assignment cost of k is $d'_k c_{i''k}$. The contribution of k to the assignment flow cost is at least $d'_k(c_{i''k} - c_{i_2(k)k}) + d'_k\min(c_{ik}, c_{i'k})$ since $k \in A_j$. This is at least $d'_k c_{i''k}$ since both $c_{ik}, c_{i'k}$ are at least $c_{i_2(k)}$. So the assignment cost of $(\tilde{x}, \tilde{y})$ is bounded by the flow assignment cost. This completes the proof. ■

Theorem 3.5 *The integer solution $(\tilde{x}, \tilde{y})$ translates to an integer solution to the original instance of cost at most $10 \cdot OPT$.*

Proof : By the previous two lemmas, the cost of $(\tilde{x}, \tilde{y})$ is at most twice the cost of $(\hat{x}, \hat{y})$ and hence at most $6 \cdot OPT$ by Lemma 3.2. We lose an additive factor of at most $4 \cdot OPT$ by Lemma 3.1 to translate $(\tilde{x}, \tilde{y})$ to a solution to the original instance with client set $\mathcal{D}$ and demands d_j, so the theorem follows. ■

4 Extension to the k-median variant

We can use the techniques in the previous section to handle an extension of the problem where in addition a bound of k_s is imposed on the number of caches that can store object s, for every object s. This adds the constraints $\sum_i y^s_i \leq k_s \ \ \forall s$ to the LP relaxation (P).

We need to modify the min-cost flow network construction slightly in Step II and Step IV of the previous section. In Step II, we remove the edges (w_j, t). Instead for every object s, we add a node p_s with demand $|D_s| - k_s$ and edges (w_j, p_s) for $j \in D_s$ of capacity $\frac{1}{2}$ and cost $3\gamma_j$. This limits the total flow on edges (v_j, a_j) and (w_j, b_j) where $j \in D_s$ to k_s. We also add edge (p_s, t) with capacity k_s and cost 0. The half-integral solution $(\hat{x}, \hat{y})$ is obtained as before with p_s playing the role of t. It is easy to see that $(\hat{x}, \hat{y})$ is feasible and Lemma 3.2 still holds.

In Step IV, we remove the edges (w_i^s, t). For every s, we add a node p_s with demand $|\{i : B_i^s \neq \phi\}| - (k_s - |D_s'|)$, add edges (w_i^s, p_s) of capacity 1 and cost 0, and add edge (p_s, t) with capacity $k_s - |D_s'|$ and cost 0. $(\tilde{x}, \tilde{y})$ is obtained as before and Lemmas 3.3 and 3.4 still hold. So we get the following theorem.

Theorem 4.1 *There is a 10-approximation algorithm for the data placement problem with a priori bounds on the number of caches that may store an object.*

References

[1] I. Baev and R. Rajaraman. Approximation algorithms for data placement in arbitrary networks. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 661–670, 2001.

[2] M. Charikar, S. Guha, É. Tardos, and D. B. Shmoys. A constant-factor approximation algorithm for the k-median problem. *Journal of Computer and System Sciences*, 65(1):129–149, 2002.

[3] F. Chudak and D. Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. *SIAM Journal on Computing*. To appear.

[4] S. Guha and K. Munagala. Improved algorithms for the data placement problem. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 106–107, 2002.

[5] K. Jain, M. Mahdian, E. Markakis, A. Saberi, and V. Vazirani. Greedy facility location algorithms analyzed using dual-fitting with factor-revealing LP. *Journal of the ACM*. To appear.

[6] K. Jain and V.V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation. *Journal of the ACM*, 48:274–296, 2001.

[7] J. H. Lin and J. S. Vitter. ϵ-approximations with minimum packing constraint violation. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 771–782, 1992.

[8] M. Mahdian, Y. Ye, and J. Zhang. Improved approximation algorithms for metric facility location. In *Proceedings of 5th APPROX*, pages 229–242, 2002.

[9] D. B. Shmoys. Approximation algorithms for facility location problems. In *Proceedings of 3rd APPROX*, pages 27–33, 2000.

[10] D. B. Shmoys, C. Swamy, and R. Levi. Facility location with service installation costs. To appear in *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2004.

[11] D. B. Shmoys, É. Tardos, and K. I. Aardal. Approximation algorithms for facility location problems. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 265–274, 1997.

[12] M. Sviridenko. An improved approximation algorithm for the metric uncapacitated facility location problem. In *Proceedings of 9th IPCO*, pages 240–257, 2002.

[13] C. Swamy and D. B. Shmoys. Fault-tolerant facility location. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 735–736, 2003.