

# Importance Sampling via Load-Balanced Facility Location

Aaron Archer and Shankar Krishnan

AT&T Labs—Research, 180 Park Avenue, Florham Park, NJ 07932.  
{aarcher, krishnas}@research.att.com

**Abstract.** In this paper, we consider the problem of “importance sampling” from a high dynamic range image, motivated by a computer graphics problem called *image-based lighting*. Image-based lighting is a method to light a scene by using real-world images as part of a 3D environment. Intuitively, the sampling problem reduces to finding representative points from the image such that they have higher density in regions of high intensity (or energy) and low density in regions of low intensity (or energy).

We formulate this task as a facility location problem where the facility costs are a function of the demand served. In particular, we aim to encourage load balance amongst the facilities by using V-shaped facility costs that achieve a minimum at the “ideal” level of demand. We call this the *load-balanced facility location* problem, and it is a generalization of the uncapacitated facility location problem with uniform facility costs. We develop a primal-dual approximation algorithm for this problem, and analyze its approximation ratio using dual fitting and factor-revealing linear programs. We also give some experimental results from applying our algorithm to instances derived from real high dynamic range images.

## 1 Introduction

This paper introduces a discrete optimization model to address a problem arising in computer graphics. The clustering problem we propose, a variant of facility location, is NP-hard, so we design an approximation algorithm to attack it. This approach is unusual for computer graphics, where most methods that are proposed are heuristics that do not come with any sort of provable guarantees on the quality of the solutions they produce.

Suppose that we wish to synthetically render a scene that is being illuminated by multiple directional light sources. The challenge here is to accurately represent the reflections of light off of the surfaces in the scene. The way that light scatters off of a surface depends on physical properties of the surface material and on the directions from which incident light rays are striking the surface. By knowing these two things, we can compute how much light would be scattered in the direction of the viewer’s eye, and hence how we should render the surface.

One typically assumes that all of the light sources are infinitely far away, so that the incident illumination is identical at every point in space throughout the scene, modulo occlusions and reflections from objects inside the scene. Therefore, to capture all relevant information about the lighting, it is enough to obtain a single *light probe image*,

a spherical map of the intensity of light coming from each direction. From this spherical map, we can then compute the light incident on any given point via ray tracing, accounting for other elements of the scene that might occlude the direct lighting from certain directions, and also for single or multiple reflections from surfaces in the scene. This lighting method is called *image-based lighting* in computer graphics—that is, a method to light a scene by using real-world imagery as part of a 3D environment. The resolution of the light probe image is much greater than the number of rays we can typically afford to trace, and therefore we desire to sample a much smaller number of rays and trace only those. It is this sampling problem that we address in this paper.

Intuitively, we would expect to obtain the most faithful rendering if we sample many rays in regions of high intensity (or energy), and allocate relatively few in regions of low intensity (or energy). In other words, the quality of approximating an image-based lighting environment as a finite number of point lights is increased if the light positions are chosen to follow the distribution of the incident illumination. This is called the *importance sampling* problem. Debevec [4] suggests the following guideline. Suppose that the total intensity summed over all pixels in the light probe image is  $I$ , and we have a budget of  $B$  light rays (i.e., pixels from the light probe image) to sample. According to Debevec, subdividing the image into  $B$  regions of equal intensity and choosing a single light source in each region (center or centroid) should be sufficient. While he suggests a particular heuristic for defining the regions and placing a light source in each, we can interpret his strategy as follows: we should try to sample our  $B$  light rays such that when we compute the Voronoi diagram of our chosen rays and sum the intensities of the pixels in each Voronoi region, the total intensity is split as evenly as possible amongst the regions, i.e., each sum is as close as possible to  $I/B$ .

We cast this ray sampling task as a type of facility location problem. Every pixel is a client with demand equal to its intensity, and every pixel is also a potential facility. We wish to open a set of facilities and assign each unit of demand to some open facility so as to minimize the sum of two types of costs.<sup>1</sup> The first is a connection cost for each unit of demand, proportional to the distance between that demand and the facility that serves it. We assume that these distances form a metric. The second cost is incurred by each open facility, and is a function of the amount of demand served by that facility. The facility cost will be a “V” shape, described by three parameters. The first is a *target capacity*  $U$ , which denotes the ideal amount of demand that the facility should serve. Second, there is a fixed cost  $F_0$  to open the facility. Third, there is a penalty  $\Lambda$  for each unit by which the total demand served by the facility deviates from the ideal capacity  $U$ . Thus, if the facility serves  $D$  units of demand, the facility cost would be  $F_0 + \Lambda|D - U|$ . If  $D > U$  we say that the facility is *overfull*; if  $D < U$  we say that it is *underfull*. We denote by  $d_{ij}$  the connection cost of routing one unit of demand at client  $j$  to facility  $i$ . We call this the *load-balanced facility location problem* (LBFL). Notice that when  $\Lambda = 0$ , our problem reduces to the uncapacitated facility location problem with uniform facility costs, which is NP-hard.

Let us examine how we could apply this model to obtain a good solution for our ray sampling problem. Here, the connection costs are given by the great circle distance on the sphere. These costs encourage each demand to be routed to its Voronoi center, while

---

<sup>1</sup> Note that we allow a single client to split its demand amongst multiple facilities.

the facility costs encourage the facilities to be placed such that each serves roughly  $U$  units of demand. Clearly, we should set  $U = I/B$ . This leaves us two knobs to turn,  $F_0$  and  $\Lambda$ , in order to achieve our twin goals of opening roughly  $B$  facilities and splitting demand evenly amongst them. Consider a demand point  $j$ , whose nearest open facility (i.e., Voronoi center) is  $i$ . This demand can be routed to some other facility  $i'$  only if  $i$  is overfull,  $i'$  is underfull, and  $d_{i'j} < d_{ij} + 2\Lambda$ , because in this case the increased connection cost is overwhelmed by saving a penalty of  $\Lambda$  on each of the two facility costs. Thus, small values of  $\Lambda$  force the clusters to look more like Voronoi cells, while large values of  $\Lambda$  push the cluster sizes towards  $U$ . Setting  $F_0$  to be sufficiently high will cause only one facility to be opened, which will result in high connection costs and a highly overfull facility. As we decrease  $F_0$  while keeping  $\Lambda$  fixed, the number of facilities will increase, in order to decrease both the connection cost and the penalty for overfullness. Thus, our choice of  $\Lambda$  expresses a tradeoff between how well our clusters should resemble Voronoi cells and how much the demand should be balanced among them. Fixing a value for  $\Lambda$ , we can adjust  $F_0$  so as to open roughly  $B$  facilities.

There are other ways that we could have modeled ray sampling as a discrete optimization problem in order to satisfy Debevec's criterion of roughly balanced Voronoi cells. The most obvious would be to explicitly request  $B$  samples such that the vector of demands contained in the  $B$  Voronoi regions they induce is as close as possible to the vector  $(I/B, \dots, I/B)$  under some norm. Another option would be to formulate it as a facility location problem with a hard constraint that the total demand served by each facility must fall inside some window surrounding  $I/B$ . We chose our formulation because it seemed to be the most tractable.

**Related Work.** Facility location problems have been a major focus of study in operations research since the 1960s, and in theoretical computer science since the 1990s. Many variants of this problem have been studied. The most popular one is the *uncapacitated* version (UFL), where each facility  $i$  pays a fixed cost for being opened, and can then serve an unlimited amount of demand for no extra cost. See [14, pp. 22-25] for an excellent summary of much of this work and a long list of references.

Since most variants of the problem are NP-hard, they have inspired a large body of work developing approximation algorithms for them, particularly for the uncapacitated version. An *r*-*approximation algorithm* is an algorithm that runs in polynomial time, and is guaranteed to return a solution of value at most  $r$  times the optimum;  $r$  is called the *approximation ratio*. The uncapacitated facility location problem has provided fertile ground for developing a large range of approximation techniques, including LP rounding, randomized rounding, pipage rounding, cost-scaling, greedy augmentation, local search, and primal-dual methods, including dual fitting and factor-revealing linear programs (FRLPs). Again, see [14] for references. The current best approximation factor, due to Mahdian, Ye and Zhang [16], is 1.52 and uses a combination of cost-scaling and dual fitting, analyzed using FRLPs. Regarding hardness, Guha and Khuller [7] proved that no 1.463-approximation algorithm exists unless  $NP \subseteq \text{DTIME}[n^{O(\log \log n)}]$ .

Several papers [9, 15, 5] have given approximation algorithms for facility location problems with facility costs that vary with the demand served, but all of these assume

that these costs are increasing with the demand. In contrast, our costs are V-shaped, so they first decrease then increase. This turns out to drastically change the structure of the problem. It also causes standard local search techniques such as those in [15, 5] to break. These algorithms bound the connection cost by considering a sequence of moves that add all the optimal facilities to the current solution and re-route all demand to them. In our problem, shedding demand from a facility can actually *increase* its cost, so this bounding technique breaks.

Svitkina [17] recently gave a  $(558 + \epsilon)$ -approximation algorithm for the *lower-bounded facility location problem*, which is identical to UFL, except that each facility has a hard lower bound on the amount of demand it must serve. This model was previously introduced by [8, 13]. It is not appropriate for our desired application, because it does not give the opportunity to penalize for overloading a facility, so the solutions are not load-balanced. For our graphics application, it might be reasonable to use a model that imposes both upper and lower bounds on the demands served. However, an optimal solution for this model would tend to push the demands toward the bounds, so we prefer our formulation since it encourages true even splitting.

We use the technique of dual fitting, which involves constructing a dual solution to an LP relaxation of our problem along with an integer primal solution in such a way that the dual exactly pays for the primal, but is infeasible and hence does not give a valid lower bound. However, it can be scaled down by some factor  $\gamma$  to give a valid lower bound, resulting in an approximation guarantee of  $\gamma$ . One uses an FRLP to find a value of  $\gamma$  that will work for all instances. This technique goes back to Chvatal’s analysis of the greedy algorithm for set cover [3], and was formalized by Jain et al. [11], who applied it to UFL. Mahdian, Ye and Zhang then used it to obtain the current best approximation factor for that problem [16], and it has been successfully applied to other combinatorial optimization problems as well [2, 10, 12]. FRLPs were first used explicitly by Goemans and Kleinberg [6] (although not in the dual fitting framework) to analyze their algorithm for the minimum latency problem. Our work extends the techniques and applicability of dual fitting and FRLPs.

**Our Results.** We develop an approximation algorithm for the LBFL problem where the approximation factor depends on a parameter  $\lambda = AU/F_0$ , but is a constant with respect to the size of the input. We analyze the approximation ratio using the technique of dual fitting and FRLPs. This means that the analysis of our approximation algorithm boils down to analyzing the optimal solutions of an infinite family of linear programs (LPs). This involves identifying patterns in the optimal dual solutions to these LPs, so that we can establish a valid upper bound on the optimal value of each LP in the infinite family.

Like the FRLPs previously used for UFL [11, 16], our FRLP has quadratically many “triangle inequality” constraints. Using a clever trick based on the max flow / min cut theorem, we reduce this to a linear number (Theorem 1). This is important for three reasons. First, the new, smaller LP and its dual are much simpler and help highlight the structure of their solutions. Second, the smaller size allows us to solve larger FRLPs faster, which is useful when exploring the solution structure. Both of these features are important because a detailed understanding of the structure of optimal solutions to the

FRLP is needed to prove upper bounds on the approximation ratio of our algorithm. Third, Theorem 1 can be applied to [11, 16] as well, allowing a simpler derivation of their results and possibly aiding any future work that uses their framework.

In Jain et al.'s analysis for UFL, the collection of FRLPs that requires analysis is only a single-parameter family, corresponding to the possible values of the total demand served by a single facility. Our collection is indexed also by  $U$ ,  $\lambda$  and another parameter  $k^*$ , so it is a 4-parameter family. This is a significantly more complex beast, but we are able to tame it. The approximation guarantee that we derive depends on  $\lambda$ , which is inherent in our approach because the LP we use to get lower bounds on  $OPT$  has an integrality gap of  $\Omega(\lambda)$ .

Finally, we applied our algorithm to instances arising from real images, and report the results.

## 2 The Model

The input to our problem is a set of *facilities*  $\mathcal{F}$ , a set of *clients* (or *demand points*)  $\mathcal{D}$ , a metric  $d$  on  $\mathcal{F} \cup \mathcal{D}$  where  $d_{ij}$  denotes the distance from facility  $i$  to demand point  $j$ , a fixed facility cost  $F_0$ , an ideal capacity  $U$  and a penalty parameter  $\Lambda$ . These last three parametrize a facility cost function  $F_{F_0, U, \Lambda}(D) = F_0 + \Lambda|D - U|$  for  $D \geq 0$ . We assume for simplicity of exposition that  $U$  is an integer and every client has unit demand, but everything can still be made to work without this assumption. While  $F_0$ ,  $U$  and  $\Lambda$  are the most natural parameters from a modeling perspective, for our analysis it will be more convenient to rescale the facility costs. Let  $\lambda = \Lambda U / F_0$ ,  $p = F_0 / U$ , and  $f_{U, \lambda}(D) = U + \lambda|D - U|$ . Then  $F_{F_0, U, \Lambda}(D) = p f_{U, \lambda}(D)$ . We will ordinarily omit the subscripts  $U$  and  $\lambda$ , and write the facility cost as  $pf(D)$ .

The goal is to open a subset of facilities  $\mathcal{O} \subseteq \mathcal{F}$  and select an assignment  $\phi : \mathcal{D} \rightarrow \mathcal{O}$  of demands to open facilities so as to minimize the quantity

$$\sum_{j \in \mathcal{D}} d_{\phi(j)j} + \sum_{i \in \mathcal{O}} pf(|j : \phi(j) = i|),$$

which is the sum of the connection costs of each client to the facility that serves it, plus the sum of the facility costs for each facility to serve the amount of demand routed to it. Notice that for any fixed set of facilities, the optimal assignment of demands can be computed using min-cost flow, so (by integrality of flows) if we were to allow ourselves to split a client's unit demand over multiple facilities, it would not lower the cost.

In much of our analysis, we will have to distinguish between the two cases  $\lambda < 1$  and  $\lambda > 1$ , owing to a qualitative difference between the two. For a facility serving exactly the ideal demand  $U$ , the facility cost per unit demand is  $p$ , while the over- or underfullness penalty is  $p\lambda$ . Thus, for  $\lambda > 1$ , the cost per unit demand achieves a minimum when the facility is exactly full, whereas for  $\lambda < 1$ , the cost per unit demand continues to decrease even once the facility is overfull.

## 3 Our Algorithm and Analytic Framework

Let us define a *star*  $(i, S)$  to be a facility  $i \in \mathcal{F}$ , along with a collection of demand points  $S \subseteq \mathcal{D}$  that it might serve. Every solution to our facility location problem can be

viewed as a partition of the demand points by stars, where the center of the star is an open facility and the facility serves exactly the demand points in  $S$ . This problem can be modeled by an integer program whose linear relaxation is shown below, along with its LP dual. We use  $c_{(i,S)} = \sum_{j \in S} d_{ij} + pf(|S|)$  to denote the cost of the star  $(i, S)$ .

$$\begin{aligned}
\min \sum_{(i,S)} c_{(i,S)} x_{(i,S)} & & \max \sum_{j \in \mathcal{D}} \alpha_j - \sum_{i \in \mathcal{F}} \mu_i \\
\text{s.t. } \sum_{\substack{i \in \mathcal{F}, \\ S \ni j}} x_{(i,S)} = 1, \quad \forall j \in \mathcal{D} & & \text{s.t. } \sum_{j \in S} \alpha_j - \mu_i \leq c_{(i,S)}, \quad \forall (i, S) \\
\sum_S x_{(i,S)} \leq 1, \quad \forall i \in \mathcal{F} & & \mu_i \geq 0, \quad \forall i \in \mathcal{F}. \\
0 \leq x_{(i,S)} \leq 1, \quad \forall (i, S) & & 
\end{aligned} \tag{1}$$

We use a dual-fitting algorithm to construct a feasible primal solution along with an infeasible solution to the dual. Our algorithm will fix all  $\mu_i$  variables to zero and never change them. Our primal and dual solutions will have the same objective function value, and we will exhibit some  $\gamma$  such that multiplying our dual solution by  $\frac{1}{\gamma}$ , makes it feasible and hence provides a lower bound on  $OPT$ . Thus, our primal solution is a  $\gamma$ -approximation. The main thrust of our work will be to prove an upper bound on the maximum  $\gamma$  that could ever be necessary to make the scaled dual solution feasible.

Conceptually, our algorithm starts all clients  $j$  with  $\alpha_j = 0$ , and raises these dual variables uniformly until some star  $(i, S)$  becomes tight. At this point, we open  $i$ , connect all clients in  $S$  to it, and freeze their dual variables. We then proceed as before, except that we consider only unfrozen clients when computing the next star to open. For an already-open facility  $i$  that is currently underfull, if any client  $j$  achieves  $\alpha_j = d_{ij} - p\lambda$ , then we connect  $j$  to  $i$  and freeze it. If  $i$  is already full or overfull, then  $j$  must achieve  $\alpha_j = d_{ij} + p\lambda$  to connect this way. We always raise the unfrozen dual variables uniformly until we first hit one of these new star or direct connection events, at which point we process the event, then continue until all demands are frozen.

This is the most natural algorithm to consider, given that we are using LP (1) and aim to use dual fitting for the analysis; it is very similar to one in [11] for UFL. It can be implemented to run in  $O(m \log m)$  time, where  $m = |\mathcal{F}||\mathcal{D}|$ . The implementation is similar to that in [11], so we omit the details.

**Deriving the Factor-Revealing LP.** By construction, the  $\sum_j \alpha_j$  resulting from the algorithm exactly pays for the primal. Since the smallest acceptable value of the scale factor  $\gamma$  is

$$\max_{(i,S)} \frac{\sum_{j \in S} \alpha_j}{c_{(i,S)}}, \tag{2}$$

we need to find an upper bound on this quantity. Notice that the max is taken over *all possible stars*, not just the ones that the algorithm actually opens. Consider the star  $(i, S)$  that maximizes (2), and let  $k = |S|$ . We rename the clients in  $S$  as  $1, \dots, k$  such that  $\alpha_1 \leq \dots \leq \alpha_k$ , and abbreviate  $d_{ij}$  as  $d_j$ . Then  $c_{(i,S)} = \sum_{j \in S} d_j + pf(k)$ . Since

scaling down  $p$  and all  $d_{ij}$  uniformly causes the algorithm to behave the same but with the  $\alpha$  scaled, we can assume WLOG that  $c_{(i,S)} = 1$ . Thus, our goal reduces to finding the largest possible value that  $\sum_{j \in S} \alpha_j$  can be, given that the algorithm imposes the following constraints.

Facility  $i$  was opened during the time interval  $[\alpha_{k^*}, \alpha_{k^*+1})$ , for some index  $k^*$  (where  $k^* = 0$  if  $i$  was opened before time  $\alpha_1$ , and  $k^* = k$  if  $i$  was opened after  $\alpha_k$  or never). We claim that the following linear program, denoted  $P(k, U, \lambda, k^*)$ , provides an upper bound on the approximation ratio for this instance. The constraints are labeled by the names we will give to their corresponding variables in the dual of this LP.

$$\begin{aligned}
& \text{maximize} && \sum_{j=1}^k \alpha_j \\
& \text{subject to} && \sum_{j=1}^k d_j + pf(k) = 1, && (\nu) \\
& && \alpha_j \leq \alpha_l + d_l + d_j + 2\lambda p, && \forall 1 \leq l < j \leq k && (\text{Tri}_{lj}) \\
& && |S|\alpha_j \leq \sum_{g \in S} d_g + pf(|S|), && \forall j \leq k^*, S \subseteq [k] && (\text{Star}_S) \\
& && && \text{s.t. } \min S = j && \\
& && \alpha_j \leq d_j + \lambda p && \forall j > k^* && (\text{DC}_j) \\
& && \alpha_j \leq \alpha_{j+1} && \forall 1 \leq j < k && (\sigma_j) \\
& && \alpha, d, p \geq 0
\end{aligned}$$

We need to justify constraints  $(\text{Tri}_{lj})$ ,  $(\text{DC}_j)$ , and  $(\text{Star}_S)$ . For  $(\text{DC}_j)$ , recall that facility  $i$  is already open before time  $\alpha_j$ , so once  $j$  pays for its connection cost modulo the  $\pm\lambda p$  penalty/discount, it will connect to  $i$  as a singleton.

For  $(\text{Tri}_{lj})$ , consider the situation faced by client  $j$  just before it connects to some facility at time  $\alpha_j$ . Demand  $l < j$  connected to some facility  $i'$  at time  $\alpha_l$ . Since  $l$  received a discount of at most  $\lambda p$ , we know  $\alpha_l \geq d_{i'l} - \lambda p$ . To connect to  $i'$  later as a singleton,  $j$  would be charged at most a  $\lambda p$  penalty. Thus, we have

$$\alpha_j \leq d_{i'j} + \lambda p \leq d_{i'l} + d_{il} + d_{ij} + \lambda p \leq (\alpha_l + \lambda p) + d_l + d_j + \lambda p,$$

from the triangle inequality and the previous lower bound on  $\alpha_l$ . Thus,  $(\text{Tri}_{lj})$  is valid.

For  $(\text{Star}_S)$ , consider the point in time just before demand  $j$  connects to some facility. Each demand  $g \geq j$  is still active at this point, and at this time all active demands have dual variable equal to  $\alpha_j$ . Since  $i$  is not yet open at this point, one possible star is the one centered at  $i$  and serving exactly these demands. Thus, the sum of their dual variables cannot exceed the cost of opening this star, since then the star would have been opened strictly before time  $\alpha_j$ . Constraint  $(\text{Star}_S)$  expresses this.

Let us denote the optimal value of  $P(k, U, \lambda, k^*)$  by  $\rho(k, U, \lambda, k^*)$ . Then our goal is to compute the quantity  $\rho(\lambda) = \sup_{k, U, k^*} \rho(k, U, \lambda, k^*)$ , because this is an upper bound on the violation factor of any possible star in the dual solution generated by our algorithm, and is hence a valid approximation guarantee. Our approximation bound must depend on  $\lambda$  since a simple example shows that the integrality gap of (1) is  $\Omega(\lambda)$ .

## 4 Analysis of the Factor-Revealing LP

In this section, we study the structure of  $P(k, U, \lambda, k^*)$  in depth, so we can understand what optimal solutions to this LP look like. In particular, we would like to construct a near-optimal solution to its dual, which will give a good upper bound on  $\rho(\lambda)$ . We start by simplifying the LP itself. We then show that the worst case LP is when  $U = k^* = k$  and  $k \rightarrow \infty$ . Finally, we construct a dual solution for this worst case to establish an upper bound on  $\rho(\lambda)$ .

**Simplifying  $P(k, U, \lambda, k^*)$ .**  $P(k, U, \lambda, k^*)$  has  $\binom{k}{2}$   $(\text{Tri}_{l_j})$  constraints, and it turns out that many of these are redundant. By considering the feasibility system consisting of only these constraints, taking its dual, performing a transformation based on the max flow-min cut theorem and taking the dual again we can derive the following theorem (whose proof we defer to the full paper because of space limitations):

**Theorem 1** *The non-negative variables  $\alpha, d, p$  satisfy all the  $(\text{Tri}_{l_j})$  constraints iff there exist  $\omega_1, \dots, \omega_k \geq 0, t \in \mathbb{R}$  s.t.*

$$\alpha_j \geq t - d_j - \lambda p + \sum_{g:g>j} \omega_g, \quad \forall 1 \leq j < k \quad (\text{TG}_j)$$

$$\alpha_j \leq t + d_j + \lambda p + \sum_{g:g \geq j} \omega_g, \quad \forall 1 < j \leq k. \quad (\text{TL}_j)$$

Therefore, if we add the  $\omega$  variables to  $P(k, U, \lambda, k^*)$  and replace the  $(\text{Tri}_{l_j})$  constraints with the  $(\text{TG}_j)$  and  $(\text{TL}_j)$  constraints, we end up with an equivalent LP, which we denote  $\bar{P}(k, U, \lambda, k^*)$ . The dual of this LP is  $\bar{D}(k, U, \lambda, k^*)$ :

$$\begin{aligned} & \text{minimize} && \nu \\ & \text{subject to} && \\ & \left. \begin{aligned} & \text{TL}_j - \text{TG}_j + \text{DC}_j + \sigma_j - \sigma_{j-1} \\ & + \sum_{S:\min S=j} |S| \text{Star}_S \end{aligned} \right\} \geq 1 && \forall 1 \leq j \leq k \quad (\alpha_j) \\ & \nu \geq \text{TL}_j + \text{TG}_j + \sum_{S:j \in S} \text{Star}_S + \text{DC}_j && \forall 1 \leq j \leq k \quad (d_j) \\ & f(k)\nu \geq \sum_S f(|S|) \text{Star}_S + \lambda \sum_{j=1}^k (\text{TL}_j + \text{TG}_j + \text{DC}_j) && (p) \\ & \sum_{j=1}^k \text{TG}_j - \sum_{j=1}^k \text{TL}_j = 0 && (t) \\ & \sum_{l:l<j} \text{TG}_l \geq \sum_{l:l \leq j} \text{TL}_l && \forall 1 \leq j \leq k \quad (\omega_j) \\ & \text{DC}_j = 0 \ (j \leq k^*), \text{Star}_S = 0 \ (S \text{ s.t. } \min S > k^*) \\ & \sigma_0 = \text{TL}_1 = \text{TG}_k = 0, \text{TL}, \text{TG}, \text{Star}, \sigma \geq 0 \end{aligned}$$

**Reducing to  $\rho(k, k, \lambda, k), k \rightarrow \infty$ .** We next proceed to show that  $U = k^* = k$ , with  $k \rightarrow \infty$  is the worst case for this LP (i.e., the value of the optimal solution is highest for this case).

**Theorem 2** For all  $k, U, \lambda, k^*$ , we have  $\rho(k, U, \lambda, k^*) \leq \rho(k, k, \lambda, k^*)$ .

**Proof:** Fix any solution to  $\bar{D}(k, k, \lambda, k^*)$ . We show that this solution is also feasible for  $\bar{D}(k, U, \lambda, k^*)$  for all  $U$ , and the result follows. The only constraint in  $\bar{D}(k, U, \lambda, k^*)$  that depends on  $U$  is (p), since the functions  $f(\cdot)$  depend implicitly on  $U$ . Thus, we only need to show that these constraints are still satisfied when we change  $U$ .

Let  $s_l = \sum_{S:|S|=l} \text{Star}_S$ , and  $g(U) = f_U(k)\nu - \sum_l f_U(l)s_l - \lambda \sum_j (\text{TL}_j + \text{TG}_j + \text{DC}_j)$  (i.e.,  $g(U)$  is the slack in the (p) constraint). We know  $g(k) \geq 0$  and wish to show that  $g(U) \geq 0$  for all  $U$ . Recall that

$$f_U(l) = U + \lambda|l - U| = \begin{cases} (1 - \lambda)U + \lambda l, & \text{if } U \leq l \\ (1 + \lambda)U - \lambda l, & \text{if } U \geq l \end{cases} \quad (3)$$

Thus,  $\frac{\partial}{\partial U} f_U(l) = 1 + \lambda$  for all  $l \leq k \leq U$ . Moreover, for each fixed  $l$ ,  $f_U(l)$  is a convex function of  $U$ , so  $g(U)$  is concave on  $[0, k]$ .

First consider the case  $U \geq k$ . Differentiating gives  $g'(U) = (1 + \lambda)(\nu - \sum_l s_l)$ . But since  $f_k(k)\nu \geq \sum_l f_k(l)s_l + \lambda \sum_j (\text{TL}_j + \text{TG}_j + \text{DC}_j)$  and  $f_k(l)$  achieves its min at  $l = k$ , we have  $\nu \geq \sum_l s_l$ . Thus,  $g'(U) \geq 0$  for all  $U \geq k$ , so  $g(U) \geq 0$  as well.

Now consider the case  $U \leq k$ . Since  $g$  is concave on  $[0, k]$  and  $g(k) \geq 0$ , it is enough to show that  $g(0) \geq 0$ , which simplifies to  $k\nu \geq \sum_l l s_l + \sum_j (\text{TL}_j + \text{TG}_j + \text{DC}_j)$ . Summing all the  $(d_j)$  constraints gives precisely this. ■

**Theorem 3** For all  $k, \lambda, k^*$ , and every  $M \in \mathbb{N}$ ,  $\rho(k, k, \lambda, k^*) \leq \rho(Mk, Mk, \lambda, Mk^*)$ .

**Proof:** For simplicity, we prove the theorem for  $M = 2$ . The proof for general  $M$  is analogous. Given any feasible solution to  $\bar{P}(k, k, \lambda, k^*)$ , we will “double” it to obtain a feasible solution to  $\bar{P}(2k, 2k, \lambda, 2k^*)$  of equal value, and the result follows.

Let  $(\alpha, d, p, t, \omega)$  be our solution to  $\bar{P}(k, k, \lambda, k^*)$ . We propose a solution  $(\bar{\alpha}, \bar{d}, \bar{p}, \bar{t}, \bar{\omega})$  to  $\bar{P}(2k, 2k, \lambda, k^*)$ . Let  $\bar{p} = p/2$  and, for each  $j \in [k]$ , let  $\bar{\alpha}_{2j-1} = \bar{\alpha}_{2j} = \alpha_j/2$ ,  $\bar{d}_{2j-1} = \bar{d}_{2j} = d_j/2$ . By construction, the objective value is still the same, and constraints  $(\nu)$ ,  $(\text{DC}_j)$ ,  $(\sigma_j)$ , are still satisfied. By Theorem 1,  $(\alpha, d, p)$  satisfies the  $(\text{Tri}_j)$  constraints, so  $(\bar{\alpha}, \bar{d}, \bar{p})$  does too. By Theorem 1, there exist  $\bar{\omega}$  and  $\bar{t}$  such that the new solution satisfies  $(\text{TG}_j)$  and  $(\text{TL}_j)$ . The  $(\text{Star}_{\bar{S}})$  constraints are a bit more subtle.

Given any  $\bar{S} \subseteq [2k]$ , there exist sets  $O, E \subseteq [k]$  s.t.  $\bar{S} = \{2j - 1 : j \in O\} \cup \{2j : j \in E\}$ . Summing constraints  $(\text{Star}_O)$  and  $(\text{Star}_E)$  for the original solution gives

$$\begin{aligned} |O|\alpha_{\min O} + |E|\alpha_{\min E} &\leq \sum_{g \in O} d_g + \sum_{g \in E} d_g + pf_k(|O|) + pf_k(|E|) \\ (|O| + |E|)\alpha_{\min(O \cup E)} &\leq 2 \sum_{g \in \bar{S}} \bar{d}_g + 2\bar{p}(k + \lambda(k - |O|) + k + \lambda(k - |E|)) \quad (4) \\ 2|\bar{S}|\alpha_{\min \bar{S}} &\leq 2 \sum_{g \in \bar{S}} \bar{d}_g + 2\bar{p}f_{2k}(|\bar{S}|), \end{aligned}$$

where (4) follows from  $(\sigma_j)$ . This last inequality is simply twice  $(\text{Star}_{\bar{S}})$ . ■

**Theorem 4** For all  $k, \lambda, k^*$ , we have  $\rho(k, k, \lambda, k^*) \leq \rho(k, k, \lambda, k)$ .

**Proof sketch:** In  $\bar{D}(k, k, \lambda, k^*)$ , when we change  $k^*$  from  $k$  to something smaller, we lose some  $(\text{Star}_S)$  constraints but gain some  $(\text{DC}_j)$  constraints. Our goal is to show that this makes the LP tighter. Combining the  $(\text{DC}_j)$  and  $(\sigma_j)$  constraints yields  $\alpha_j \leq \min_{g \geq j} d_g + p\lambda$ , for  $j > k^*$ . This constraint implies most of the  $(\text{Star}_S)$  constraints. Fixing  $S \subseteq \{k^* + 1, \dots, k\}$ , with  $j = \min S$ , we have

$$|S|\alpha_j \leq |S| \min_{g \geq j} d_g + |S|p\lambda \leq \sum_{g \in S} d_g + |S|p\lambda$$

This implies the  $(\text{Star}_S)$  constraint as long as  $|S|\lambda \leq f(|S|) = k + \lambda(k - |S|)$ , that is,  $|S| \leq \frac{(1+\lambda)k}{2\lambda}$ . This is always the case as long as either  $\lambda \leq 1$ , or  $\lambda \geq 1$  and  $k^* \geq \frac{\lambda-1}{2\lambda}k$ . The proof of the remaining case (i.e.,  $\lambda \geq 1, k^* < \frac{\lambda-1}{2\lambda}k$ ) is much more involved, so we defer it to the full version of paper. ■

Putting Theorems 2, 3 and 4 together yields:

**Corollary 5**  $\rho(\lambda) = \lim_{k \rightarrow \infty} \rho(k, k, \lambda, k)$ .

**Bounding  $\rho(k, k, \lambda, k)$  with a Canonical Dual.** Now that we have reduced our problem to analyzing  $\lim_{k \rightarrow \infty} \rho(k, k, \lambda, k)$ , we can use an LP solver to help us further explore the structure of the optimal solutions to  $\bar{D}(k, k, \lambda, k)$  and  $\bar{P}(k, k, \lambda, k)$ . Theorem 3 suggests the optimal solutions will be roughly scale-invariant, meaning that if we express all the indices as fractions of  $k$  and scale down the primal variables and the dual  $\text{Star}_S$  variables by a factor of  $k$ , the solutions for large  $k$  should look very similar as  $k$  varies. We solved a bunch of these LPs with CPLEX, which confirmed this intuition.

**Table 1.**  $\rho(k, k, \lambda, k)$  for different values of  $k$  and  $\lambda$  (generated by CPLEX), along with the best upper bound on  $\rho(\lambda)$  that we can prove analytically.

$k$	$\lambda = 0.0$	$\lambda = 0.25$	$\lambda = 0.5$	$\lambda = 0.75$	$\lambda = 1.0$	$\lambda = 1.5$	$\lambda = 2.0$
50	1.7979	1.9499	2.0948	2.2310	2.4684	3.0212	3.5694
100	1.8064	1.9588	2.1039	2.2400	2.4844	3.0427	3.5967
200	1.8107	1.9632	2.1084	2.2445	2.4925	3.0535	3.6106
400	1.8128	1.9655	2.1107	2.2467	2.4966	3.0590	3.6175
proof	1.8608	2.0396	2.1719	2.2500	2.5007	3.0645	3.62438

Table 1 shows the computed value of  $\rho(k, k, \lambda, k)$  for a range of  $\lambda$  and  $k$  values, along with the best upper bound we can prove for  $\rho(\lambda)$  using the method outlined below. For fixed  $\lambda$ , we see that the approximation bound  $\rho(k, k, \lambda, k)$  appears to be converging fairly rapidly. Since the full details of proving the bounds are quite hairy, we defer them to the full version of the paper, but here we sketch the steps involved.

Our overall goal is to create a “canonical” solution to  $\bar{D}(k, k, \lambda, k)$  which can be “projected” onto a feasible dual solution for any given value of  $k$ , thereby yielding an

upper bound on  $\rho(k, k, \lambda, k)$ . Because of Corollary 5, we care about the quality of this bound only for large  $k$ .

We first establish which variables in  $\bar{P}(k, k, \lambda, k)$  and  $\bar{D}(k, k, \lambda, k)$  are non-zero. We can readily deduce some of the patterns just by looking at the LPs. For instance, since we expect  $p$  to be non-zero in an optimal solution, there is no way for both  $(TG_j)$  and  $(TL_j)$  to be tight for the same  $j$ . Thus, by complementary slackness,  $TG_j$  and  $TL_j$  should never both be non-zero. Inspection of the optimal solutions output by CPLEX reveals further structure. For instance, when  $\lambda \geq 0.8022$ , the solutions adhere to the following pattern. All  $\alpha_j$  are positive, and all  $\sigma_j$  and  $\omega_j$  are zero. The only  $\text{Star}_S$  variables that are non-zero are for tail stars, i.e., ones of the form  $S_j = \{j, \dots, k\}$ . There exists a partition of the indices  $\{1, \dots, k\}$  into three intervals,  $I_1 = \{1, \dots, b_1k\}$ ,  $I_2 = \{b_1k + 1, \dots, b_2k\}$ ,  $I_3 = \{b_2k + 1, \dots, k\}$ , where  $0 \leq b_1 \leq b_2 \leq 1$  are roughly constant with respect to  $k$ , but depend on  $\lambda$ . These intervals determine the non-zero pattern of the remaining variables, as follows. For  $j \in I_1$ ,  $d_j$ ,  $TG_j$ , and  $\text{Star}_{S_j}$  are non-zero, while  $\text{Star}_{S_j}$  is also non-zero on  $I_2$ , and  $TL_j$  is non-zero on  $I_3$ . All other primal and dual variables are zero.

Once we have identified the pattern of non-zeros in the primal and dual, we use the complementary slackness conditions to determine which dual constraints should be tight. We then solve this system of linear equations to determine the non-zeros in the dual solution, up to a dependence on a small handful of parameters. Doing this for the pattern described above yields the following solution in terms of the parameters  $b_1, b_2, c$ :  $\text{Star}_{S_j} = \frac{c}{k}$  on  $I_1$  and  $\frac{1}{|S_j|}$  on  $I_2$ ;  $TG_j = |S_j| \frac{c}{k} - 1$  on  $I_1$ ;  $TL_j = 1$  on  $I_3$ . Any choice of these parameters defines a canonical dual solution  $(TG, TL, \text{Star}_S, \sigma_j, \nu)$ . Since the  $(d_j)$  and  $(p)$  constraints are just lower bounds on  $\nu$ , as long as  $(\alpha_j)$ ,  $(t)$ ,  $(\omega_j)$  and the non-negativity constraints are satisfied, we can just set  $\nu$  to be the max of these lower bounds. Thus, any choice of the parameters that causes the dual to satisfy these constraints is valid. Treating the parameters as variables, we can solve a non-linear program to optimize them such that  $\nu$  is as small as possible. This whole process results in a proof of our upper bound on  $\rho(\lambda)$ , for any particular  $\lambda$ . When  $\lambda = 1$  for example, we get  $(b_1, b_2, c) = (0.2068, 0.5577, 3.5007)$ , yielding  $\nu = 2.5007$ . Fortunately, the behavior of  $\bar{D}(k, k, \lambda, k)$  is qualitatively the same over intervals of  $\lambda$  (e.g.,  $\lambda \geq 0.8022$ ), so we have to derive only a few of these ‘‘parameter-revealing’’ non-linear programs. However, to obtain the actual bound on  $\rho(\lambda)$ , we need to solve one of these non-linear programs for that particular  $\lambda$ . This is why we do not give an explicit formula for our approximation bound as a function of  $\lambda$ , although we can say that it is asymptotic to  $\lambda$ . Since a trivial example gives an integrality gap of  $\lambda/3$  for LP (1), our algorithm is within a constant of the best one can do if one uses LP (1) as the lower bound on  $OPT$ .

**Notes on Solving  $\bar{D}(k, U, \lambda, k^*)$ .** As we have shown, actually solving  $\bar{P}(k, U, \lambda, k^*)$  and  $\bar{D}(k, U, \lambda, k^*)$  computationally and inspecting the results was invaluable in deriving our approximation guarantees. It was also quite useful in guiding us to discover the theorems leading to Corollary 5. Thus, it is worth noting the multiple ways in which Theorem 1 helps us to solve these LPs efficiently and analyze them effectively, even though doing so is necessary only for the analysis of our algorithm, not for running the algorithm itself.

The transformation from  $P(k, U, \lambda, k^*)$  and  $D(k, U, \lambda, k^*)$  to  $\bar{P}(k, U, \lambda, k^*)$  and its dual  $\bar{D}(k, U, \lambda, k^*)$  allowed by Theorem 1 helps enormously in deciphering the form of the canonical dual solution.  $D(k, U, \lambda, k^*)$  has  $\binom{k}{2}$  Tri variables, which are highly redundant and thus cause a lot of degeneracy in the optimal solution of  $D(k, U, \lambda, k^*)$ , which makes it very difficult to pick out the relevant patterns in the Tri variables. Switching to  $\bar{D}(k, U, \lambda, k^*)$  greatly simplifies our task by getting rid of the degeneracy, and also by giving us many fewer variables to understand (just  $2k - 2$  TL and TG variables).

Since there are exponentially many  $\text{Star}_S$  variables, we solve  $\bar{P}(k, U, \lambda, k^*)$  using cutting planes (or equivalently, solve  $\bar{D}(k, U, \lambda, k^*)$  using column generation). It is easy to find the most violated  $\text{Star}_S$  constraint amongst all  $S$  s.t.  $\min S = j$ . We just sort clients  $g = j + 1, \dots, k$  by increasing  $d_g$ , and take all  $g$  for which  $\alpha_j - d_g + \lambda p > 0$ , provided there are at most  $U - 1$  of them. If there are more, we take the first  $U - 1$ , and in addition we take any  $g$  such that  $\alpha_j - d_g - \lambda p > 0$ . The most violated  $S$  consists of the clients we just identified, plus  $j$ .

We initialize the cutting plane method with the  $k$  tail stars (i.e.,  $S_1, \dots, S_k$ ). In our experience, the number of additional cuts generated was always less than  $3k$ . Thus, in practice, the LPs actually considered by the solver always have  $\Theta(k)$  variables and constraints. This highlights the next reason that Theorem 1 is so helpful: it reduces the size of our LPs from quadratic to linear in  $k$ . When trying to divine the asymptotic behavior of  $\rho(k, k, \lambda, k)$ , we want to push  $k$  as high as we can while still being able to solve  $\bar{D}(k, k, \lambda, k)$  reasonably quickly. Thus, the size and solution time of the LP really becomes an issue, even though we are using the LPs only to help us prove theorems.

Note that when  $U \geq k$ , we do not have to worry about stars of size larger than  $U$ . Thus, the separation oracle becomes easier since we need not sort the  $d_g$ . This also means that we could get rid of the  $(\text{Star}_S)$  constraints entirely by defining new variables  $C_{jt} \geq 0$  and adding the constraints  $C_{jt} \geq \alpha_t - d_j + \lambda p$  ( $\forall j \geq t$ ) and  $\sum_{j:j \geq t} C_{jt} \leq pU(1 + \lambda)$  ( $\forall t$ ). However, this is a bad idea because it again blows up the size of the LP from linear to quadratic in  $k$ .

## 5 Experimental Results

In this section, we describe some experimental results obtained from applying our algorithm to sample High Dynamic Range (HDR) images used in computer graphics for lighting scenes. We implemented our algorithm in C++. Its theoretical worst-case running time is  $O(m \log m)$  where  $m$  is the product of the number of demand points and the number of potential facilities. In this setting, the demand and facility sets are both identified with the pixels in the image. Hence  $m = (\# \text{ pixels})^2$ , which is prohibitively high even for moderate-sized images. However, in practice our implementation manages to avoid explicitly considering most (client, facility) pairs and hence runs much faster than  $O(m \log m)$ .

In order to further speed up the running time for our experiments, we need to reduce the cardinality of the facility set. We do so by generating some set of  $K \gg B$  pixels that we would expect to be good candidates for our final samples, and use these as  $\mathcal{F}$ . We then run our load-balanced facility location algorithm on this input to identify

roughly  $B$  of these  $K$  pixels as our final solution. Since we want our initial facility set  $\mathcal{F}$  to lie in regions of high intensity, a good solution to the  $K$ -median problem seems to be a natural attractive choice for  $\mathcal{F}$ .<sup>2</sup> We generate such an  $\mathcal{F}$  using an algorithm proposed by Arthur and Vassilvitskii [1]. Their algorithm chooses  $K$  random facilities in succession, according to a distribution that depends on the facilities already chosen. It gives an  $O(\log K)$ -approximate solution for  $K$ -median (in expectation). The algorithm, in addition to its theoretical guarantees, is very fast and simple to implement. For a budget of  $B$  samples, we set  $K$  to be either  $16B$  or  $32B$ .

Our algorithm has two parameters  $F_0$  (the facility cost at  $U$ ) and  $\lambda$  (the penalty per unit demand for deviating from ideal capacity  $U$ ).  $\lambda$  allows us to trade off Voronoi-ness against load balance. If  $\lambda = 0$ , we get Voronoi cells. If  $\lambda = \infty$ , we get perfect load balance. Given a fixed  $\lambda$ ,  $F_0$  allows us to control how many facilities are opened. If  $F_0 = \infty$ , only one facility will be opened. When  $F_0 = 0$ , more will be opened. Adjusting these values gives us a way to control the number of open facilities as well as the load balance. We use the following strategy to set  $F_0$  and  $\lambda$  so that we get roughly  $B$  approximately load-balanced facilities. We start with  $F_0 = 0$  and  $\lambda = 0$ . This setting opens all facilities. We increase  $\lambda$  until we open about  $2B$  facilities. We then increase  $F_0$  until the number of open facilities is close to  $B$ .

**Data Sets** We use 5 high dynamic range images of various sizes to test our algorithm. They are named *Memorial*, *Rosette*, *Nave*, *Lamps* and *Tree*. Table 2 shows the statistics for the different input images.

**Table 2.** Statistics about the input images.

Name	Image Size	# Fac. $ \mathcal{F} $	Budget $B$	Tot. Intensity $I$	$U$
Memorial	$512 \times 512$	2048	128	50889.8	397.6
Rosette	$256 \times 256$	4096	256	98443.0	384.5
Nave	$256 \times 256$	2048	64	191955.2	2999.3
Lamps	$256 \times 256$	1024	64	3104.0	48.5
Tree	$256 \times 256$	1024	64	11033.6	172.4

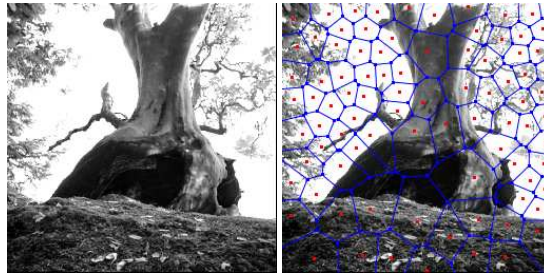
**Table 3.** Results from our algorithm.

Name	$F_0$	$\lambda$	$\lambda$	# Open Fac.	Total Fac. Cost	Total Conn. Cost	Approx. Ratio
Memorial	45.0	1.3	11.4	134	6030.0	81529.9	5.15
Rosette	35.0	0.9	9.88	252	8820.0	116180.0	2.15
Nave	60.0	0.54	27.0	66	3960.0	197038.0	6.23
Lamps	90.0	0.3	0.16	72	6480.0	14471.8	1.32
Tree	122.5	0.37	0.52	88	10780.0	53787.2	1.38

<sup>2</sup> The  $K$ -median problem is the same as UFL, except that there are no facility costs (only connection costs), and the number of open facilities is restricted to be  $K$ .

**Table 4.** Evaluating our algorithm’s output for load balance and Voronoi-ness.

Name	# Fac. Open	Conn. Cost Ratio (ours/Voronoi)	Our Under/Over-full Demand (% of I)	Voronoi Under/Over-full Demand (% of I)
Memorial	134	1.0503	10.2655 / 5.5779	27.1197 / 22.4322
Rosette	252	1.0705	6.0096 / 7.5720	11.6977 / 13.2601
Nave	66	1.0547	11.8984 / 8.7734	17.9927 / 14.8676
Lamps	72	1.0626	41.3946 / 28.8821	39.1269 / 26.6140
Tree	88	1.0093	38.7969 / 1.2788	38.5549 / 1.0368



**Fig. 1.** Results of our algorithm on the *Tree* data set. On the left is the original image; on the right, we have superimposed our samples and their Voronoi cells.

Table 3 shows the results of our algorithm on the input images. The first three columns indicate our settings for  $F_0$ ,  $\Lambda$  and  $\lambda = \Lambda U / F_0$ . The next four columns show the number of facilities opened by our algorithm, the total facility cost, the total connection cost and an upper bound on the approximation ratio for this particular instance. We derived our upper bound on the approximation ratio by computing (2), i.e., identifying the most violated star in the dual solution  $\alpha$  generated by our algorithm. The actual approximation ratio of this instance could be substantially better, since the lower bound generated by scaling down  $\alpha$  uniformly is likely to be fairly weak, compared to the bound one would obtain by actually solving LP (1). It may be feasible to solve this LP via column generation; we leave this to future work.

Finally, to test the results with our original objective of sampling such that each open facility roughly serves equal demand and that most clients connect to their closest open facility, we compare the total connection cost of our output with that of the Voronoi cells for the for each facility. If these costs are close, that implies that most of the demand is met by the closest facility. We also list the total demand served by overfull and underfull facilities. If there is perfect load balance and we open exactly  $B$  (the budget) facilities, then they would be zero. Table 4 shows the details for each of the data sets. Figure 1 depicts the sampling produced by our algorithm on the *Tree* instance, along with the corresponding Voronoi cells.

## Acknowledgments

We thank David Applegate for sharing his expertise with AMPL.

## References

1. D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. *SODA*, 1027–1035, 2007.
2. Nikhil Bansal, Lisa Fleischer, Tracy Kimbrel, Mohammad Mahdian, Baruch Schieber, and Maxim Sviridenko. Further improvements in competitive guarantees for QoS buffering. *ICALP*, 196–207, 2004.
3. Vasek Chvatal. A greedy heuristic for the set covering problem. *Math. Oper. Res.*, 4:233–235, 1979.
4. Paul Debevec. Image-based lighting. *IEEE Comput. Graph.*, 22(2):26–34, 2002.
5. Naveen Garg, Rohit Khandekar, and Vinayaka Pandit. Improved approximation for universal facility location. *SODA*, 959–960, 2005.
6. Michel X. Goemans and Jon M. Kleinberg. An improved approximation ratio for the minimum latency problem. *Math. Program.*, 82:111–124, 1998.
7. Sudipto Guha and Samir Khuller. Greedy strikes back: improved facility location algorithms. *J. Algorithm*, 31(1):228–248, 1999.
8. Sudipto Guha, Adam Meyerson, and Kamesh Munagala. Hierarchical placement and network design problems. *FOCS*, 603–612, 2000.
9. Mohammad Taghi Hajiaghayi, Mohammad Mahdian, and Vahab S. Mirrokni. The facility location problem with general cost functions. *Networks*, 42(6):42–47, 2003.
10. Nicole Immorlica, Mohammad Mahdian, and Vahab S. Mirrokni. Cycle cover with short cycles. *STACS*, 641–653, 2005.
11. Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *J. ACM*, 50(6):795–824, 2003.
12. Mohammad R. Salavatipour Kamal Jain, Mohammad Mahdian. Packing Steiner trees. *SODA*, 266–274, 2003.
13. David R. Karger and Maria Minkoff. Building Steiner trees with incomplete global knowledge. *FOCS*, 613–623, 2000.
14. Mohammad Mahdian. *Facility Location and the Analysis of Algorithms through Factor-Revealing Programs*. PhD thesis, MIT, Cambridge, MA, June 2004.
15. Mohammad Mahdian and Martin Pál. Universal facility location. *ESA*, 409–421, 2003.
16. Mohammad Mahdian, Yinyu Ye, and Jiawei Zhang. Improved approximation algorithms for metric facility location problems. *APPROX*, 229–242, 2002.
17. Zoya Svitkina. Lower-bounded facility location. *SODA*, 1154–1163, 2008.