IBM Research Report

Local Search Heuristic for budget constrained k-median Problem

Vinayaka Pandit
IBM Research Division
IBM India Research Lab
Block I, I.I.T. Campus, Hauz Khas
New Delhi - 110016. India.

Naveen Garg
Dept. of Computer Science and Engg
IIT, New Delhi - 110016. India

Rohit Khandekar
Dept. Of Computer Science and Engg
IIT, New Delhi

IBM Research Division
Almaden - Austin - Beijing - Delhi - Haifa - T.J. Watson - Tokyo - Zurich

Abstract

k-median, and k-center are well studied problems in operations research. In this paper, we present local search algorithm to obtain good solutions with respect to these measures. It is easy to show that both k-median, and k-center cannot be approximated within constants simultaneously. So we consider one of natural relaxations of the problem, namely minimizing the cost of k-median solution when a bound is placed on the k-center measure it induces. We call this problem as *budgeted k-median problem*. Our local search search technique produces a constant factor approximation of the median cost, while ensuring that the bound on the k-center measure is not violated by more than a constant factor. Our technique is based on local search technique presented in [1]. The key idea of our technique is the preprocessing of the input to satisfy the *min-max* criteria of the optimization problem. The best approximation we obtain provides a factor 5 approximation to the median cost.

1 Introduction

k-median, and k-center are well studied OR problems. The objective in k-median problem is to minimize a global sum, and the objective in k-center is a *min-max* criterion. Consider the problem of obtaining a simultaneous good solution for both k-median, and k-center problems. The contrasting goals of the objective functions can be exploited to construct examples where obtaining such a solution is not possible. For example, consider an input instance where there are two clusters of very small radius containing $n/2$ points each. Suppose these two clusters are separated by a distance 1. Suppose there is a $n+1$th point which is at a distance $\sqrt{n}$ from both the clusters. Let k be equal to two. The best k-median solution opens two facilities inside the clusters, and the cost of the solution is $\sqrt{n}$. The best k-center solution opens a facility inside one of the clusters, and one facility at the single point, and has a cost 1. It is easy to see that any solution which opens exactly k facilities exceeds one of the measures by a factor of $O(\sqrt{n})$. We consider a natural relaxation of the above problem. We are given a budget on the k-center solution, henceforth called *mean budget*, and the objective is to find an optimal k-median solution whose induced k-center cost is within the mean budget. We call this problem as *budget constrained k-median problem* or *budgeted k-median problem*. We assume that there exists atleast one solution which satisfies mean budget, and we assume that the set of facilities, F is same as set of clients, C. Thus:

- **Input** : A set of clients C in a metric space, an integer k, and a mean budget α.
- **Assumption** : There exists a k-median solution in which no client travels more than α.
- **Output**: Minimum cost k-median solution in which no client travels more than α.

Clearly, budgeted k-median problem is NP-Complete. We can reduce a k-median instance to a budgeted k-median instance by specifying the budget to be the largest distance between points in the set of clients. We propose a local search heuristic for the problem. We call any solution which opens k facilities such that all clients are within mean budget as a *mean budgeted solution*. Throughout our discussion, we denote the distance between two clients $c1$, and $c2$ by $dist(c1, c2)$. We denote the number of clients in the input by n.

The rest of this paper is organised as follows. In Section 2, we briefly discuss works related to our problem. In Section 3, we will present a scheme to preprocess the input points to guide us in opening facilities. In Section 4, we will present the local search algorithm for budgeted k-median problem. In Section 5, we will analyze the performance of our local search algorithm. In Section 6, we will show how to extend our technique to get trade-off on the two criteria. We conclude this paper in Section 7.

2 Related Work

We briefly discuss previous works on facility problems related to our problem, mainly the k-median problem, and k-center problem. Here I will add some more references, how some of the known k-median solutions cant be imported, how jain-vazirani technique can be imported. Also a very brief local search for facloc survey.

In [1], Arya *et al* proposed a simple local search heuristic for the k-median problem, and proved that the local search provides an approximation guarantee of $3(1+\epsilon)$. In this paper, we will propose a local search heuristic for the budgeted version of the k-median problem. The local search algorithm in [1] starts with an arbitrary choice of k facilities, and performs a series of local swaps. It does not place any restriction on the swaps except that the swap improve the cost of the solution. We have to be more careful in the budgeted version in picking our initial set of facilities. We will also have to be careful about the swaps considered because the swaps should be such that, all clients travel a distance within constant times α.

So the key ideas to formulate a local search technique to solve budgeted k-median problem are:

1. Open the initial set of k facilities such that every client is within a distance of constant times mean budget from an open facility.
2. Consider only those swaps which improve the cost, and maintain the invariant that every client is within constant times mean budget from an open facility.

Throughout our discussion, we assume that set of facilities is same as set of clients. If a client is chosen by an algorithm, then it is called a facility, otherwise it will continue to be a client.

3 Covering the clients

It is clear from the preceding discussion that, being able to ensure that no client travels a long distance is key for the local search algorithm. So we preprocess our input to form a set of logical regions which cover all clients, and help us to meet the objective. This preprocessing is called as *covering phase*. We form region in such a way that

1. The diameter of the region is within a constant times mean budget. Hence, if we open a facility inside a region, then all clients inside the region can be served while satisfying the mean budget constraint.
2. We form the regions such that, any mean budgeted solution is guaranteed to open a certain number of facilities inside it.

Initially, we try to cover as many clients as possible using *triplet region*s in which every k-budgeted solution is guaranteed to open atleast three facilities.

Definition 3.1 *A triplet region is defined by three points* $p1$, $p2$, *and* $p3$ *as shown in Figure 1. The points satisfy the following properties:*

1. *The distance between each two points is greater than* 2α.
2. *There exists a point out of these three, say* $p1$, *such that its distance from other two points is less than* 4α.

The union of balls of radius 2α around each points $p1$, $p2$, and $p3$ forms triplet region. We have ensured that the α balls around these points do not intersect, and all the clients inside the region are within constant factor of the from a point inside these balls. The union of balls of radius α around $p1$, $p2$, and $p3$, the shaded area in Figure 1 is called as *facility restriction area* of the triplet region. Note that any solution which satisfies mean budget has to open atleast 3 facilities in a triplet region. We construct the triplet regions as

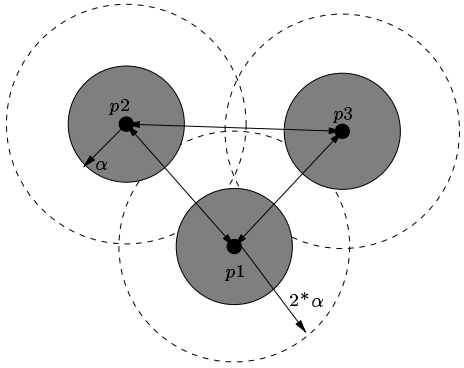

Figure 1: Illustration of a *triplet region*

follows:

Procedure Cover-By-TripletRegion

1.Mark all clients as *uncovered*.

2. While there exist uncovered clients $p1$, $p2$, and $p3$ which define a triplet region

A. Form the triplet region formed by $p1$, $p2$, and $p3$.

Mark all clients inside this region as *covered*.

End while

End Procedure

Note that, this procedure can be easily run in polynomial time. Even a naive implementation can form regions is $O(n^3)$ time. Note also that, the input may be such that we cannot form any triplet region. When the above procedure terminates, we are left with a set of uncovered clients. We will cover the uncovered clients by regions where any valid mean budgeted solution has to open atleast two facilities, or by regions where any valid mean budgeted solution has to open atleast one facility. Throughout our discussion we use the term *envelope of radius r around a region* to denote all points(including those which are not clients) which are within a distance r from the boundary of the region.

We define a *doublet region* by two points $p1$, and $p2$ such that (i) $2\alpha < dist(p1, p2) < 4\alpha$; and (ii) there does not exist an uncovered client $p3$ such that $p1$, $p2$, and $p3$ form a triplet region. The union of balls of radius 2α around points $p1$, and $p2$ forms a doublet region as shown in Figure 2A. When the procedure to cover client using triplet regions terminates there will be many doublet regions.

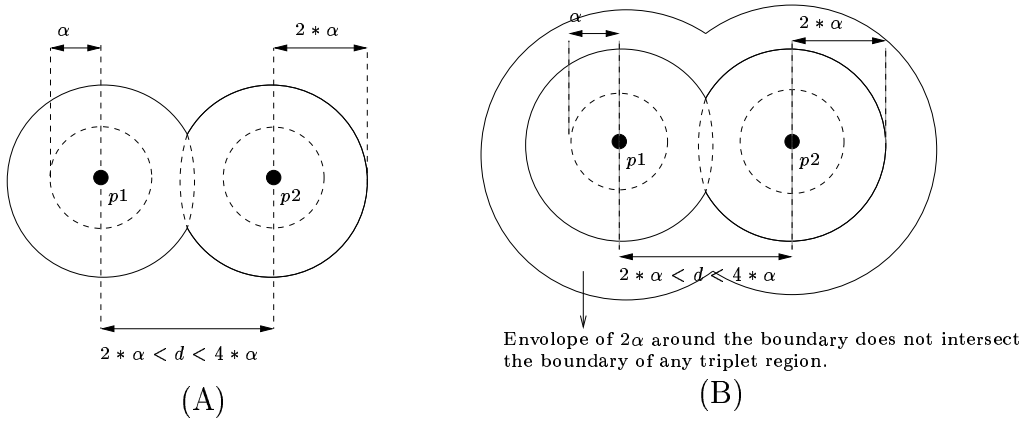

Figure 2: Illustration of a *doublet region*

Lemma 3.1 *After the covering phase by triplet region, for every doublet region there does not exist an uncovered client in the envelope of radius 2α around it.*

Proof. Suppose an uncovered client $p3$ exists in the envelope of radius 2αaround the doublet region defined by $p1$, and $p2$. Then, the points $p1, p2$, and $p3$ define a triplet region, and the covering phase by triplet regions would not have terminated. ∎

Note that some of the doublet regions could be close to an already formed triplet region. Specifically, we consider a doublet region to be close to a triplet region if the envelope of radius 2α around the doublet region intersects with the boundary of the triplet region. In such a case we mark all the clients inside such a doublet region as covered, and count it with any one of the triplet regions close to it. A doublet region whose envelope of radius 2α does not intersect the boundary of any triplet region is called as *isolated doublet region* (See Figure 2B). We mark all clients inside an isolated doublet region as marked, and as belonging to it. So far, we have covered client by marking them as (i) belonging to a triplet region directly or being inside close doublet region, OR (ii) belonging to an isolated doublet region.

Clearly, we may not be able to cover all clients as above. There will still be many points such that their distance from every uncovered client is either less than 2α, or greater than 4α. Each set of uncovered points which are at a distance of less than 2α from each other is called a *singlet region* as shown in Figure 3.

Lemma 3.2 *For every singlet region, its envelope of radius 2α does not intersect the boundary of a doublet region.*

Proof. Similar to the proof of 3.1. ∎

A singleton region is close to a triplet region if its envelope of radius 2α intersects the boundary of the triplet region. We mark all clients inside such a singlet region as marked, and belonging to one of the triplet

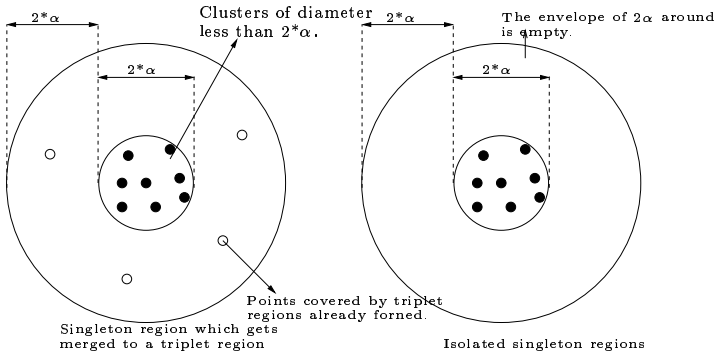

Figure 3: Illustration of a *singlet region*

regions close to it. If the envelope of radius 2α around a singlet region does not intersect the boundary of any triplet region, then we call such a region as *isolated singlet region.* We mark all clients inside an isolated singlet region as belonging to it. Thus we have covered all clients using triplet regions, isolated doublet regions, and isolated singlet regions.

We make some simple observations:

1. The total number of regions triplet regions, isolated doublet regions, and isolated singlet regions to cover all clients is less than or equal to k.
2. The distance between clients belonging to same region is within a constant times of mean budget.
3. Any mean budgeted solution has to open atleast three facilities inside a triplet region, atleast two facilities inside an isolated doublet region, and atleast one facility inside an isolated singlet region.

4 Local Search Algorithm

After covering all the clients as mentioned above, we run a local search algorithm proposed in [1] with some restrictions. We define a *valid solution* as a solution such that

- It opens atleast one facility inside the facility restriction area of each triplet region.
- It opens atleast one facility inside every isolated doublet region.
- It opens atleast one facility inside every isolated singlet region.

We define a *valid single swap* with respect to a valid solution as closing an open facility, and opening a new facility such that the new solution is also a valid solution. We formulate our local search as:

Begin Local Search

1. Cover all the points as described in the previous section.
2. Pick an initial valid solution.
3. While there exists a valid single swap which improves the cost
   3A. Perform the valid single swap.
4. Output the locally optimum valid solution.

Lemma 4.1 *In a valid solution, every client is within a distance 21α from an open facility.*

Proof. The distance between any two clients inside an isolated singleton region is within 2α, and distance between any two clients inside an isolated doublet region is within 8α. Since we open atleast one facility

inside facility restriction of every triplet region, the largest distance of such a facility from a client in that region is given by a client at one end of a close double region, and the points defining the triplet region being on a straight line. This is bounded by 21α. ∎

Lemma 4.2 *If the covering phase does not form any triplet region, then all the clients are covered by isolated doublet regions, and isolated singlet regions.*

Proof. Follows from 3.1, and 3.2. ∎

Let A denote any mean budgeted solution, or a valid solution. For every A, and an open facility in A, say a, we define *neighborhood* of a, denoted by $N_S(s)$ to be the set of clients served by a in A. We denote the locally optimal valid solution output by our algorithm by S. Let OPT be any optimal mean budgeted solution. For every facility o in OPT, we partition $N_O(o)$ into $N_s^o = N_O(o) \cap N_S(s)$ for each $s \in S$. We say that $s \in S$ *captures* an optimal facility $o \in O$ if $|N_s^o > |N_O(o)|$. We call a facility in our solution as *bad* if it captures atleast one optimal facility. Furthermore, we call a facility in S as *t-bad* if it captures exactly t optimal facilities. We call a facility in S as *(t+)-bad* captures atleast t optimal facilities. If a facility in our solution does not capture any facility in O then it is called *good* facility.Consider a step of swapping an optimal facility o in, and swapping a facility $s \in S$ out. We have to be be able to reroute all clients which were served by s in our solution. To help us in bounding the reroute cost, we will define a 1:1, and onto function $\pi : N_O(o) \rightarrow N_O(o)$ which satisfies the following important property.

Property 4.1 *For all $s \in S$ and $o \in O$ such that,*
$|N_s^o| \leq \frac{1}{2}|N_O(o)|$, we have, $\pi(N_s^o) \cap N_s^o = \emptyset$.

It is easy to see that such a mapping exist, and can be computed efficiently. For more details on all the notions the reader is referred to [1].

5 Analysis

In this section, we compare the cost of the locally optimal valid solution output by our algorithm with any optimal solution. We know that any valid swap does not improve the cost of our solution. We will consider k valid swaps, and write equations corresponding to each swap. By simple adding these equations, we obtain a ration of the cost of our solution to any optimal solution. In [1] this technique of analysis has been presented in greater detail. We use many of those results. The main ideas can be recapped as follows:

1. Consider the swap of a facility in our solution with that of OPT, say $\langle s, o \rangle$. To bound the change in cost, we route all client in $N_S(s) \cap N_O(o)$ to o. We route all other clients in $N_S(s)$, say c to $\pi(c)$ via o, and then to the facility which serves c in our solution.
2. We consider every facility in OPT in exactly one swap. Let us call it as *swap-out constraint* for swaps.
3. We do not consider any *(2+)-bad* facility in a swap. Let us call it as *two-bad constraint* for swaps.
4. A *1-bad* facility is considered in atmost one swap, and even when swapped it is swapped with the OPT facility that it captures. Let us call it as *one-bad constraint* for swaps.
5. It is possible to satisfy all the above conditions by swapping in every *good* facility atmost twice.

This was sufficient to prove that, the locally optimum solution is within 5 times the optimal. For the rest of our discussion, we will call all of swap-out constraints, two-bad constraints, one-bad constraints as *sawp constraints*. The key *idea is to prove that it is possible to satisfy swap constraints while swapping out any good facility atmost a constant number of times.* Suppose this constant is t, then it implies a $2t + 1$ approximation for the median optimal. For more details, the reader is referred to [1].

Lemma 5.1 *Suppose we do not form any triplet region in the covering phase, then no facility in our local optimum solution captures an OPT facility outside its region.*

Proof. Follows from the definition of isolated doublet, and isolated singlet region, and lemma 4.2. ∎

From the above lemma, it follows that a 1-bad facility captures an OPT facility inside its own region. So swapping a 1-bad facility with the facility it captures is also a valid swap. Also note that a good facility is not sitting alone in its region(otherwise, it should have captured atleast one facility by lemma 5.1, and hence can be considered in any swap with OPT facility is valid.

Lemma 5.2 *If no triplet region is formed in the covering phase, then cost of out locally optimum solution is within 5 times the cost of any optimal budgeted solution.*

Proof. From the above discussions, it is clear that, we can satisfy all swap constraints by swapping good facilities out atmost twice, and thus yielding a factor 5 approximation. ∎

We will assume that atleast one triplet region is formed, otherwise from lemma 5.2, we already have achieved our objective. So we will now be concerned with the how to consider swaps with the OPT facilities when triplet regions are formed. As observed earlier the facilities in our solution inside an isolated doublet region, and isolated singlet region cannot capture any OPT facility outside their regions. But we will face some problems in considering all OPT facilities in a swap, each time maintaining validity of our solution.

Let us consider what limitations could we face while considering only valid swaps to be able to satisfy *sawp constraints*. Suppose that, our local search algorithm has opened only one facility,say c, inside the facility restriction area of triplet region. Further, suppose c is a *1-bad facility* capturing an OPT facility outside the facility restriction area of the triplet region. Then, we call c as *constrained 1-bad facility*. We denote the number of constrained one bad facilities by $b1$. We could also face restrictions to swap a good facility. Suppose c is the only facility we have opened inside the facility restriction area of a triplet region. Suppose the facility opened by OPT in the three α balls are captured by our 1-bad facilities sitting outside the facility restriction area, then we cannot consider this good facility in any valid swap with an OPT facility outside. We call such a good facility as *constrained good facility*. We denote the number of good facilities by $g1$. We call all other good facilities as unconstrained, and denote their number by $g2$. We denote the number of (2+)-bad facilities by b_{2+}. We denote the number of unconstrained 1-bad facilities by l.

We will now specify the swaps considered to satisfy the sawp constraints:

Swaps Considered:

1. Suppose s is an unconstrained 1-bad facility capturing o, then we consider the swap $\langle s, o\rangle$, and write the corresponding equations.
2. Now the remaining OPT facilities are considered in swaps with only unconstrained good facilities.

We claim that,

Claim 5.1 *After considering swaps of unconstrained 1-bad facilities, the remaining OPT facilities can be considered in swaps with only unconstrained good facilities such that every unconstrained good facility is swapped out only constant number of times.*

After we consider the swaps of all unconstrained 1-bad facilities, we are left with $k1 = k - l$ OPT facilities to be considered for swaps. Clearly, the number of constrained 1-bad facilities, and constrained good facilities is bounded by the maximum number of triplet regions required to cover $k1$ facilities.

$$g1 + b1 \leq k1/3 \tag{1}$$

We are also left with exactly $k1$ facilities in our solution which have not been considered so far:

$$g1 + g2 + b1 + b_{2+} = k1 \tag{2}$$

We know that the number of facilities captured by *bad* facilities cannot exceed $k1$.

$$b1 + 2b_{2+} \leq k1 \tag{3}$$

Simple manipulations show that:

$$\begin{aligned} 2g1 + 2g2 + 2b1 + 2b_{2+} &= 2k1\{\text{multiplying 2 by 2 }\} \\ 2g1 + 2g2 + b1 + (b1 + 2 * b_{2+}) &= 2k1 \\ 2g1 + 2g2 + b1 >&= k1\{\text{by substituting 3 }\} \\ g1 + 2g2 \geq 2k1/3&\{\text{ follows from 1 }\} \\ 2g2 \geq k1/3&\{\text{by substituting 1 }\} \\ 6g2 \geq k1& \end{aligned} \tag{4}$$

Lemma 5.3 *OPT facilities not considered in swaps of unconstrained 1-bad facilities can be considered in single swaps with unconstrained good facilities such that, each remaining OPT facility is swapped-in once, and an unconstrained good facility is swapped-out atmost 6 times.*

Proof. Follows from Equation 4.

Theorem 5.4 *The cost of locally optimum valid solution output by our algorithm is atmost 13 times the cost of any optimal solution, thus yielding a (21,13) approximation for the budgeted k-median problem.*

Proof. Follows from 4.1, 5.3, and preceding discussion of results from [1]. ∎

6 Extentions

In this section we show how to extend our idea to obtain a bi-criteria approximation for the budgeted k-median problem. We achieve this by generalising the idea of triplet regions. The idea is to form regions such that a larger number of facilities are guaranteed to be open inside them by any mean budgeted solution.

Specifically, we form *X-regions* for any constant X. Similar to triplet region, an X-region is defined X clients such that, (i) the distance between all of them is greater than 2α, and (ii) the diameter of this set of clients is less than $4(X-1)$. Note that, a triplet region is obtained by letting X to be 3. The X-region is defined to be the union of balls of radius 2α around the X points. So the diameter of clients within an X-region is $4(x-1)+2)\alpha$. The *facility restriction area of an X-region* is defined to be the union of balls of radius α around the X points defining it. We now try to cover all points uxing X-regions or isolated $(X-1)$-regions, and so on till isolated singlet regions. Note that, a naive implementation can cover all clients in time $O(n^X)$ time. It is easy to obtain the following theorem:

Theorem 6.1 *The local search algorithm with X-regions can be used to obtain a $(8X-3, \frac{5X-2}{X-2})$ approximation for the budgeted k-median problem.*

7 Conclusions and Open Problems

In this paper, we have provided an analysis for the bicriteria approximation of k-median, and k-means. We note that, (constant,constant) is possible by extending techniques proposed by Jain, and Vazirani. We have analysed a simple local search heuristic for the same problem. Although the results provided by us are inferior compared to Jain, and Vazirani, it gives some intuition about the structure of local minimas.

References

[1] V. Arya, N. Garg, R. Khandekar, V. Pandit, A. Meyerson, and K. Munagala. Local search heuristics for k-median, and facility location problems. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, July 2001.

[2] M. Charikar and S. Guha. Improved combinatorial algorithms for the facility location and k-median problems. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, October 1999.

[3] F.A. Chudak. Improved approximation algorithms for uncapacitated facility location problem. In *Proceedings of the 6th Conference on Integer Programming and Combinatorial Optimization*, June 1998.

[4] F.A. Chudak and D.B. Shmoys. Improved approximation algorithms for capacitated facility location problem. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, january 1999.

[5] F.A. Chudak and D.P. Williamson. Improved approximation algorithms for capacitated facility location problems. In *Proceedings of the 7th Conference on Integer Programming and Combinatorial Optimization*, June 1999.

[6] S. Guha and S. Khuller. Greedy strikes back: Improved facility location algorithms. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, January 1998.

[7] D. Hauchbaum and D. Shmoys. A unified approach to approximation algorithms to bottleneck problems. *Journal of ACM*, 33(3), july 1986.

[8] K. Jain and V.V. Vazirani. Primal-dual approximation algorithms for metric facility location and k-median problems. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, October 1999.

[9] T. Kanungo, D. M. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Y. Wu. A local search approximation algorithm for k-means clustering. In *Proceedings of the 18th Annual ACM Symp. on Computational Geometry*, 2002.

[10] J.-H. Lin and J. S. Vitter. Approximation algorithms for geometric median problems. *Information Processing Letters*, 44, 1992.

[11] S. Lin and B. W. Kernigham. An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, 21, 1973.

[12] G. Cornuejols, G.L. Nemhauser, and L.A. Wolsey. The uncapacitated facility location problem. In P. Mirchandani and R. Francis, editors, *Discrete Location Theory*. John Wiley and Sons, 1990.

[13] M. Korupolu, C. Plaxton, and R. Rajaraman. Analysis of a local search heuristic for facility location problems. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, January 1998.

[14] M. Korupolu, C. Plaxton, and R. Rajaraman. Analysis of a local search heuristic for facility location problems. Technical Report 98-30, DIMACS, June 1998.

[15] D. B. Shmoys, E. Tardos, and K. Aardal. Approximation algorithms for facility location problems. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, May 1997.

[16] M. Charikar, S. Guha, E. Tardos, and D.B. Shmoys. A constant-factor approximation algorithm for the k-median problem. In *Proceedings of the 31th Annual ACM Symposium on Theory of Computing*, May 1999.