ON-LINE COMPUTATION FOR THE EUCLIDEAN P-MEDIAN PROBLEMS

Zuo Dai *To-yat Cheung*

Department of Computer Science,
City University of Hong Kong, Hong Kong
Email: cscheung@cityu.edu.hk.

ABSTRACT

The Euclidean *p*-median problem is to locate *p* facilities and allocate *n* fixed demand points each to one and only one of these facilities so that the weighted sum of the distances between the facilities and the demand points is minimal. In the conventional version of this problem, the *p* facilities are to be located simultaneously. This problem is known to be NP-complete. An on-line version of this problem requires the solution to be spread over *p* steps. In each step, a new facility is added and is to be located without relocating the existing facilities whereas some demand points may have to be reallocated. In this paper, it is first shown that a greedy on-line algorithm for this problem has no finite competitive ratio. An on-line algorithm with competitive ratio 2*n* is then proposed.

1 INTRODUCTION

The Euclidean *p*-median problem is to locate a set of *p* facilities and allocate a set of *n* fixed demand points each to one and only one of these facilities. Each of the demand points has a weight of demand. The objective is to minimize the sum of the weighted distances (product of the weight and Euclidean distance) between the demand points and the nearest facilities they are assigned to. This problem has a wide realm of applications. For example, (facilities, demand points) may be the (servers, clients) in a computing environment, (schools, students' homes) in a district for a school board, (hospitals, households) in a community for medical services, (managers, programmers) in a software house, etc. Many exact and heuristic algorithms for solving this problem exist in the literature [2, 3, 4, 11, 12, 14, 16]. However, since the problem is the NP-hard, all the exact algorithms require exponential computation time. Although the heuristics cost much less time, none of them can guarantee a theoretical bound on the quality of the solution.

The on-line version of the Euclidean *p*-median problem has a similar objective but accommodates a different situation where the *p* facilities are to be allocated one after another. For some reasons (such as budget constraint, outcome of the sum obtained for the facilities already located, etc.), whether or not a new facility is to be added will be decided only after all the previous facilities have been located. In other words, the solution process is divided into *p* steps. In each step, one additional facility is to be located without relocating those facilities already located while possibly reallocating some of the demand points to the newly-added facility. Note that, in each step, no information (such as number of future facilities) about the future steps is available. The objective is to minimize the sum of the weighted distances of the demand points in every step.

In an on-line environment, the solution at each step is affected by the initial data, the data provided at the current step, and all the partial solutions obtained in the previous steps. In general, an on-line algorithm, if not carefully designed, may give very good results at some steps but extremely bad results the others. A good result at one step may adversely affect the results of many subsequent steps. Also, an on-line algorithm is supposedly designed for handling all possible values of the data given initially and at all steps, Therefore, while trying

to optimize the total cost at individual steps, the main overall strategy is to avoid outrageously unacceptable solutions at any step for all possible given data.

One frequently-used measure for the overall performance of on-line algorithms is a quantity called *competitive ratio.* Roughly, it is defined as an upper bound on the ratio of the optimal solution for current step over the non-on-line optimal solution for all the steps up to and including the current step.

Several related on-line problems have been studied in the literature. In the on-line assignment problem [1], the number and locations of the facilities are known in advance and the demand points appear one after another in steps. The problem is to assign each demand point to an appropriate facility immediately in a manner that will balance the load on the facilities. It has been proved that the general greedy algorithm has the best possible competitive ratio that can be achieved by any deterministic on-line algorithm. Another related on-line k-server problem [13] is to plan the motion of k mobile facilities such that the total distance moved by the facilities (the facilities must move to the demand point for providing the service) is minimized. Also, the demand points appear one after another and the service must be provided immediately.

In this paper, an algorithm is proposed for solving the on-line version of the Euclidean p-median problem. To the best of our knowledge, this is the first on-line algorithm for this problem. Our algorithm has a competitive ratio $2n$. In general, a competitive ratio of $O(n)$ cannot be considered as a good result. However, at this state of the arts, the best results of most of the on-line algorithms for many other problems are at this order of performance. For example, the best possible competitive ratio for the on-line directed Steiner tree problem is n [17]. Furthermore, in this paper, we will show that a greedy algorithm cannot even achieve a finite competitive ratio for this problem.

Formal Presentation of Problems:

The locations of a given a set D of n demand points with weights $\{w_1,\ldots, w_n\}$ are fixed in the Euclidean plane. The locations of a set of p facilities, where $p < n$, are to be determined. The off-line and on-line Euclidean k-median problems ($k \le p$) can be formally described as follows:

Problem OFF-LINE(k, D):

Determine simultaneously the locations of the k facilities and allocate each of the demand points to one and only one of these k facilities in such a way such that the total cost $opt(k) = \sum_{j=1}^{n} w_j l_j^k$ is minimum, where l_j^k is the Euclidean distance between w_j and its nearest facility.

Problem ON-LINE(k, D):

At step k, assume that problem ON-LINE(k - 1, D) has been solved. Determine the location of a new facility k and reallocate some of the demand points to facility k so that $c(k)$ is minimum, where $c(k) = \sum_{j=1}^{n} w_j l_j^k$ and l_j^k is the Euclidean distance between w_j and its nearest facility.

Notations (In the following, index k is for step, i is for facility and j is for demand point):

w_j: the weight of demand point j. When there is no confusion, we also refer w_j *as* demand point j.

l_{je}: fixed distance between demand point j and demand point e. Note that $l_{ej} = l_{je}$.

$opt(k)$: optimal solution value of OFF-LINE(k, D).

o_i^k: location of facility i in the optimal solution for OFF-LINE(k, D).

l_j^k: distance between demand point j and its closest facility in the optimal solution for OFF-LINE(k, D).

G_i^k: the group of demand points allocated to facility i in the optimal solution for OFF-LINE(k, D).

n_i^k: the number of demand points in group G_i^k.

$opt(G_i^k)$: cost of group G_i^k based on the optimal solution for OFF-LINE(k, D). Note that $opt(k) = opt(G_1^k)+opt(G_2^k)+\ldots+ opt(G_k^k)$.

$c(k)$: total cost for ON-LINE(k, D), i.e., $c(k) = \sum_{j=1}^{n} w_j l_j^k$.

$c(w_j)$: cost of w_j based on the solution for ON-LINE(k, D), i.e., the weighted distance between w_j and its closest facility in the solution for ON-LINE(k, D). Note that $c(w_j) \le w_j l_{ij}$ for any i.

$c(X)$: cost of the set of demand points X based on the solution for ON-LINE(k, D), i.e., $\sum_{w_j \in X} c(w_j)$.

Definition 1 The *competitive ratio* r of an on-line algorithm is an upper bound on the ratio between the on-line solution value and the optimal off-line solution value over all steps and for all possible weights and distributions of the demand points. That is, for any weights and distribution of the demand points and $1 \le k \le p$, $\frac{c(k)}{opt(k)} \le r$.

We emphasize the fact that the bound spreads over all steps of the solution process. The following lemma was proved in [6] and will be used later in this paper.

Lemma 1 *Let* $\{(G_1^k, o_1^k),\ldots, (G_k^k, o_k^k)\}$ *be the optimal solution for OFF-LINE*(k, D), *where* $D = G_1^k \cup \ldots \cup G_k^k$ *and the demand points in* G_i^k *are assigned to the facility located at* o_i^k *for* $i = 1,\ldots, k$. *For* any k', where $1 \le k' \le k$, let D' be the union of any k' *of the* k *groups. That is, without loss of generality,* $D' = G_1^k \cup \ldots \cup G_{k'}^k$. *Then,* $\{(G_1^k, o_1^k),\ldots, (G_{k'}^k, o_{k'}^k)\}$ *is an optimal solution for OFF-LINE*(k', D').

2 ON-LINE ALGORITHMS

To show that our algorithm is non-trivial, let us first solve ON-LINE(k, D) by adopting a greedy strategy widely used for solving many other problems). The main idea of a greedy algorithm is to reduce as much as possible the sum of weighted distances at each step from of the previous step.

Algorithm GREEDY
for solving problem ON-LINE(k, D)

Input: A set D of n demand points and k - 1 facilities which have been located already.
Output: The location of the new facility k and the reallocation of demand points.
Method: Locate the new facility k and reallocate the n demand points in such a manner that $c(k)$ is minimized. (Note: Any method (e.g., Drezner [7]) achieving this goal can be used.)

For many other on-line problems [1, 8, 9, 10, 15, 17], a greedy algorithm has the best possible competitive ratio. For ON-LINE(p, D), however, the following observation and example confirm that this is indeed not the case.

Observation: *Algorithm GREEDY for solving ON-LINE(p, D) has* no *finite competitive ratio*.

Example: Figure 1 illustrates our observation. Consider 7 demand points whose weights satisfy: $w_1 = w_1^* = w_2 = w_2^* = w_3 = w_3^* >> w_4 = e$. The distance between w_1 *and* w_1^*, w_2 *and* w_2^* *and* w_3 *and* w_3^* are all e. : w_1, w_2, w_3 are at the vertices of an equilateral triangle and w_4 is at its center. Obviously, Algorithm *GREEDY* will locate facility 1 at w_4 since this is the optimal location for OFF-LINE(l, D). Then, no matter where the facilities are located in the next two steps, the total cost $c(3)$ at step 3 is at least $2w_1 l_1$. However, the optimal solution for OFF-LINE(3, D) is to locate the 3 facilities at the three vertices separately and the optimal value

should be around $l_1 e + 3e$. Hence, $\frac{c(3)}{opt(3)} = \frac{2w_1 l_1}{(l_1 + 3)e} \to \infty$ as $e \to 0$.

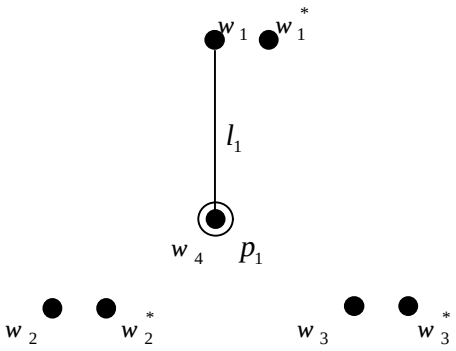


Figure 1. For explaining Observation.

The above example shows that, if we place a facility with the aim of minimizing the sum of the weighted distances of the demand points at a certain step, we run the risk of providing a very bad result at some future steps. The main strategy of on-line algorithms is to avoid such a possibility.

Our algorithm Algorithm MWD follows a similar strategy. The main idea is, at each step, to locate the new facility at a demand point which has the biggest weighted distance to its closest existing facility.

Computationally, Algorithm MWD is relatively simple. The difficulty lies in proving that it has a finite performance ratio as stated in the Theorem 2.

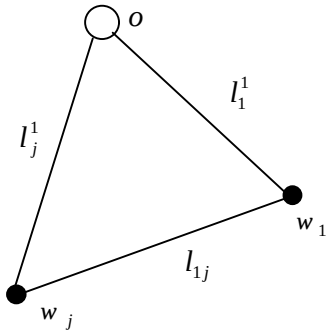


Figure 2. For proving Theorem 2 in the case p=1**.**

Algorithm *MWD* **(Maximum Weighted Distance) for problem ON-LINE(*p*, *D*)**

Input: A set *D* of fixed demand points with weights $\{w_1,...,w_n\}$.

Output: The locations of *k* facilities and the allocation of the *n* demand points.

Method:

1. For $k = 1$, locate the first facility at the demand point with maximum weight, say w_1. Allocate all demand points to this facility.
2. For $1 < k \le p$, without loss of generality, suppose the first $k - 1$ facilities are located at the demand points $w_1,...,w_{k-1}$ when solving **ON-LINE**(*i, D*), $i = 1, 2,..., k - 1$.
   (a) For $j = k,..., n$, let $d(w_j)$ be the weighted distance between w_j and its closest facility located at one of the points in the set $\{w_1,...,w_{i-1}\}$, *i.e.*, $d(w_j) = w_j \cdot \min_{1 \le e \le k-1}(l_{je})$.
   (b) Locate the new facility k at the demand point w_j *whose* $d(w_j)$ is maximum over $k \le j$ £ n. Without loss of generality, let this demand point be w_k.
   (c) Reallocate those demand points to w_k if w_k is the facility closest to them.

Theorem 2 *Let c(k) be the solution value obtained by Algorithm MWD for ON-LINE(k, D)* and *opt*(*k*) *be the optimal solution value for OFF-LINE(k, D). Then,* $\frac{c(k)}{opt(k)} \le 2n$ *for* $1 \le k \le$ *p and any possible weights and distribution of the demand points.*

Proof. We will apply mathematical induction on *p*. Figure 2 illustrates the case $p = 1$**.** Without loss of generality, let *o* be the optimal location of the facility for OFF-LINE(l, *D*) *and* w_1 be the demand point with maximum weight. Algorithm *MWD* locates the first facility at w_1. The solution value *c*(1) is

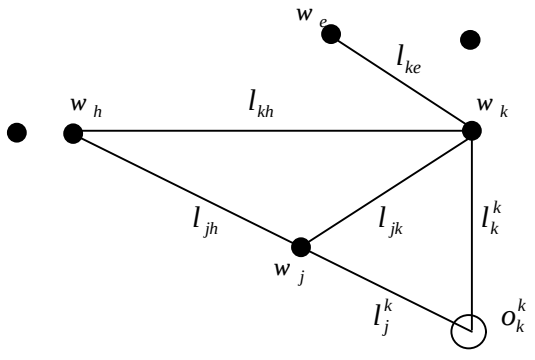
$$c(1) = \sum_{j=2}^{n} (w_j \cdot l_{1j}) \le \sum_{j=2}^{n} w_j (l_1^1 + l_j^1)$$
$$\le (\sum_{j=1}^{n} w_j l_j^1) + (n-2)\ w_1 l_1^1 \qquad (1)$$
$$= opt(1) + (n-2) w_1 l_1^1 \le 2n \cdot opt(1)$$

Assume that Theorem 2 is true for $p = k - 1$. That is, Theorem 2 is true for any problem with at most $k - 1$ facilities and any weights and distribution and number of demand points. We shall prove Theorem 2 for $p = k$. Let $\Psi = \{w_1,...,w_{k-1},w_k\}$ be the sequence of demand points where the first k facilities are located in the first k steps. Without loss of generality, let $w_k \in G_k^k$. Theorem 2 will be proved in two cases: $|G_k^k \cap \Psi| = 1$ and $|G_k^k \cap \Psi| \ge 2$.

Case 1, where $|G_k^k \cap Y| = 1$:

Consider the new p-median problem over the set D' of demand points, where $D' = D \backslash G_k^k = G_1^k \cup G_2^k \cup ... \cup G_{k-1}^k$. Let $c'(k - 1)$ be the solution value obtained by Algorithm *MWD* for ON-LINE($k - 1$, D') and $opt'(k - 1)$ be the optimal solution value for OFF-LINE($k - 1$, D'). Then, since Theorem 2 is true for $p = k - 1$, we have

$$c'(k-1) \le 2(n - n_k^k) \cdot \text{opt}'(k-1) \qquad (2)$$



Case where $|G_k^k \cap \Psi| = 1$

Figure 3.a. Illustration of Theorem 2 when $p = k$.

By Lemma 1, if $\{(G_1^k, o_1^k),..., (G_k^k, o_k^k)\}$ is an optimal solution for OFF-LINE(k, D), then $\{(G_1^k, o_1^k),...,(G_{k-1}^k, o_{k-1}^k)\}$ is an optimal solution for OFF-LINE(k-1, D'). That is, $\sum_{i=1}^{k-1} opt'(G_i^k) = \sum_{i=1}^{k-1} opt(G_i^k)$. This is the same as

$$opt'(k-1) = opt(k) - opt(G_k^k) \qquad (3)$$

Since $G_k^k \cap \Psi = \{w_k\}$, w_k is the only element in Ψ removed together with G_k^k. Next, we are going to show that $\Psi' = \Psi \backslash \{w_k\} = \{w_1,...,w_{k-1}\}$ is the sequence of locations obtained by Algorithm *M* for the first $k - 1$ facilities for ON-LINE($k - 1$, D'). According to Algorithm *MWD*, since w_1 is maximum over D and $D' \subset D$, w_1 is also maximum over D'. That is, the first facility will be located at w_1 when solving ON-LINE($k - 1$, D'). Again, according to Algorithm *MWD*, the second facility will also be located at a demand point with maximum weighted distance to w_1 over D'. Since w_2 has the maximum weighted distance to w_1 over D (note that the second facility for ON-LINE(k, D) is located at w_2) and $D' \subset D$, obviously the second facility for ON-LINE($k - 1$, D') will also be located at w_2. By similar argument, we can prove that, for ON-LINE($k - 1$, D'), Algorithm *MWD* locates the facilities at the sequence Ψ'.

Next, we consider the total allocation cost for the demand points in D' for two cases. In the first case, Ψ' is the locations of the first k-1 facilities for ON-LINE($k - 1$, D') with total cost $c'(k - 1)$. In the second case, Ψ is the locations for the first k facilities for ON-LINE(k, D') (the total cost is denoted as $c(k, D')$). Since Ψ contains one more facility than Ψ' for allocation purpose, we have $c(k, D') \le c'(k - 1)$. This is the same as

$$\sum_{i=1}^{k-1} c(G_i^k) \le c'(k-1) \qquad (4)$$

By Inequalities 2, 3 and 4, we have

$$\sum_{i=1}^{k-1} c(G_i^k) \le 2(n - n_k^k)(opt(k) - opt(G_k^k)) \qquad (5)$$

Next, we try to prove $c(G_k^k) \le 2n_k^k opt(G_k^k)$.

According to Algorithm *MWD*, we have

$$w_j l_{jh} \le w_k l_{ke} \tag{6}$$

where $l_{ke} = \min_{i \le k-1} l_{ki}$ *and* $l_{jh} = \min_{i \le k-1} l_{ji}$

Consider those $w_j \in G_k^k$. (Note that $w_k \in G_k^k$). If $w_j \le w_k$, then (Figure 3.a)

$$\begin{aligned} c(w_j) &\le w_j l_{jk} \le w_j (l_j^k + l_k^k) \\ &\le w_j l_j^k + w_j l_k^k \le opt(G_k^k) \end{aligned} \tag{7}$$

If $w_k \le w_j$, we consider c(w_j) for two subcases. In the subcase where $l_{jh} \le l_{jk}$, we have $l_{kh} \le l_{jk} + l_{jh} \le 2l_{jk}$. By Inequality 6, we get

$$\begin{aligned} c(w_j) &\le w_j l_{jh} \le w_k l_{ke} \le w_k l_{kh} \le 2w_k l_{jk} \\ &\le 2w_k (l_j^k + l_k^k) \le 2w_j l_j^k + 2w_k l_k^k \\ &\le 2opt(G_k^k) \end{aligned} \tag{8}$$

In the subcase where $l_{jk} \le l_{jh}$, we have $l_{kh} \le 2l_{jh}$. By Inequality 6, we have

$$w_j l_{jh} \le w_k l_{kh} \text{ and } w_j \le 2w_k .$$

For similar reasons as with Inequality 7, we have

$$c(w_j) \le 2opt(G_k^k). \tag{9}$$

By Inequalities 7, 8 and 9, we have

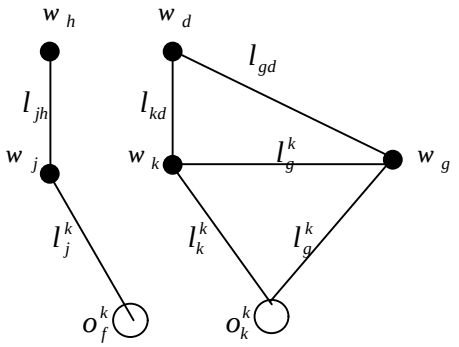
$$c(G_k^k) = \sum_{w_j \in G_k^k} c(w_j) \le 2n_k^k opt(G_k^k).$$

Adding this to Inequality 5, we have

$$c(k) \le 2n \cdot opt(k).$$

Case where $\left|G_k^k \cap \Psi\right| \ge 2$*:*

Without loss of generality, let $G_k^k \cap \Psi = \{w_g, ..., w_k\}$, where $g < k$. Since there are k groups, at least one of the groups, say G_f^k, does not contain any element of Ψ, i.e., $G_f^k \cap \Psi = f$.



Case where $|G_k^k \cap \Psi| \ge 2$

Figure 3.b. Illustration of Theorem 2 when $p = k$.

By Inequality 2, Lemma 1 and similar argument as with Inequality 5, we have

$$\sum_{i=1, i \ne f}^{k} c(G_i^k) \le 2(n - n_f^k)(opt(k) - opt(G_f^k))$$

Next, we try to prove the inequality $c(G_f^k) \le 2n_f^k opt(G_k^k)$. In Figure 3, let $l_{jh} = \min_{i<k} l_{ji}$, $l_{ke} = \min_{i<k} l_{ki}$ and $l_{kd} = \min_{i<g} l_{ki}$. Obviously, $d < g$. Consider those $w_j \in G_f^k$. In case $w_k \le w_g$, we have

$$\begin{aligned} c(w_j) &= w_i l_{jh} \le w_k l_{ke} \le w_k l_{kg} \\ &\le w_k (l_k^k + l_g^k) \le w_k l_k^k + w_g l_g^k \\ &\le opt(G_k^k) \end{aligned} \tag{10}$$

In case $w_g \le w_k$, we consider $c(w_j)$ for two subcases. In the subcase where $l_{kg} \le l_{kd}$, since $w_k l_{kd} \le w_g l_{gd}$ and $\frac{w_k}{w_g} = \frac{l_{gd}}{l_{kd}} \le 2$, we have $w_k \le 2w_g$. By similar argument as for Inequality 10, we have $c(w_j) \le 2opt(G_k^k)$. In

the subcase where $l_{kd} \le l_{kg}$, we have $l_{gd} \le 2l_{kg}$. Hence, by Algorithm *MWD*, we get

$$\begin{aligned} c(w_j) &= w_j l_{jh} \le w_k l_{ke} \le w_k l_{kd} \le w_g l_{gd} \\ &\le 2w_g l_{kg} \le 2w_g (l_k^k + l_k^g) \\ &\le 2w_g l_k^k + 2w_g l_g^k \le 2opt(G_k^k) \end{aligned}$$

For similar reasons as in the case $\left|G_k^k \cap \Psi\right| = 1$, we get $c(k) \le 2n \cdot opt(k)$.

3. Conclusion and Future Research

We have proposed an on-line algorithm with competitive ration 2*n* for solving the p-medium problem where the facilities are provided one after another and the demand points are fixed. Though, as far as we know, it is the only available on-line algorithm for solving this problem and the competitive ratio is at the same order as the on-line algorithms for many other problems, we believe that it can be further improved.

There is the need for further research for several cases, such as the case where new facilities can be added and existing facilities can be deleted, the case where the demand points can be increased or decreased at various steps, etc.

Acknowledgement - We would like to thank Dr. Xiaotie Deng for useful discussion.

Research supported financially by Research Grants Council of Hong Kong under Grant RGC CityU 1133/99E.

REFERENCES

[1] Y. Azar, J. Naor and R. Rom, The competitiveness of on-line assignments, *Journal of Algorithms* 10, 221-237, 1995.

[2] I.Bongartz, P.H. Calamai and A.R. Conn, A projection method for l_p norm location-allocation problems. *Mathematical Programming* 66, 283-312, 1994.

[3] L. Cooper, Location-allocation problems. *Operations Research* 11, 331-343, 1963.

[4] R. Chen, Solution of minisum and minimax location-allocation problems with Euclidean distances. *Naval Research Logistic* 30, 449-459, 1983.

[5] P. Crescenzi and V. Kann, A compendium of NP optimization problems, *Manuscript* 1995.

[6] Z. Dai and T.Y. Cheung, A new heuristic approach for the Euclidean p-median problem. *Journal of the Operational Research Society* 48, 950-960, 1997.

[7] Z. Drezner, On the conditional p-median problem, *Computer and Operations Research* 22, 525-530, 1995.

[8] J.A. Garay, I.S. Gopal, S. Kutten, Y. Mansour and M. Yung, Efficient on-line call control algorithms, Journal *of* Algorithms, 23, 180-194, 1997.

[9] M. Imase and B. M. Waxman, Dynamic Steiner tree problems, *SIAM Journal on Discrete Mathematics* 3, 369-384, 1991.

[10] B. Kalyaiiasundaram and K. Pruhs, On-line weighted matching, *Journal of Algorithms* 14, 478-488, 1993.

[11] R. F. Love and H. Juel, Properties and solution methods for large location-allocation problems. *Journal of Operational Research Society* 33, 443-452, 1982.

[12] C. Liu, R. Kao and A. Wang, Solving location-allocation problems with rectilinear distances by simulated annealing. *Journal of the Operational Research Society* 45, 1304-1315, 1994.

[13] M. S. Manasse, L. A. McGeoch and D.D. Sleator, Competitive algorithms for server problems, *Journal of Algorithms* 11, 208-230, 1990.

[14] F. Maffioli and G. Righini, An annealing approach to multi-facility location problems in Euclidean space. *Location Science* 2, 205-222, 1994.

[15] R. Motwani, V. Saraswat and E. Torng, On-line scheduling with look ahead: multipass assembly lines, *Technical Report of Stanford University*, 1997.

[16] K. E. Rosing, An optimal method for solving the (generalized) multi-weber problem. *European Journal of Operational Research* 58, 414-426, 1992.

[17] J. Westbrook and D. Yan, Linear bounds for on-line Steiner problems, *Information Processing Letters* 55, 59-63, 1995.