Annals of the European Academy of Sciences, 2003, pp. 127-136

Feedback Algorithm for the Single-Facility Minisum Problem

Boris S. Verkhovsky[1] and Yuriy S. Polyakov
Computer Science Department, New Jersey Institute of Technology
Newark, NJ 07102, USA

Abstract

A new accelerated algorithm to solve the single-facility minisum location problem is developed. The acceleration is achieved using a feedback factor. The proposed algorithm converges faster than the accelerating procedures available in the literature. Being nearly as simple as the classical Weiszfeld procedure, the new method can easily be implemented in real applications. Practical subroutines dealing with special cases in the minisum problem are also provided.

Keywords: minisum, accelerator, Weiszfeld procedure

[1] Author to whom all the correspondence should be addressed

1. Introduction

The Fermat-Weber (continuous single-facility minisum) problem is thoroughly studied in the literature [1]. It deals with placing a new facility on the plane so as to minimize the sum of "weighted" distances from the facility to a set of fixed planar points. Finding an optimal position for a warehouse of a multi-store company or locating a data switch on a network with many users are among many tasks where the continuous single facility minisum problem can be successfully applied to. When the distances are given by the l_p norm, the problem is formulated as follows:

$$\min W(S) = \sum_{i=1}^{n} w_i l_p(S, P_i), \tag{1}$$

where *W(S)* is the total "weighted" distance; n is the number of fixed points (also referred to as customers); w_i is the "weight" (demand) of the i-th customer; $i = 1,...,n$; $P_i = (a_i, b_i)$ is the given location of the i-th customer (demand point); $S = (x, y)$ is the unknown location of the new facility.
The distance between any P_i and S is given by

$$l_p(S, P_i) = [|x - a_i|^p + |y - b_i|^p]^{1/p}, p \geq 1. \tag{2}$$

In practice, the "weight" is thought to be proportional to the demand of the i-th customer, and $w_i l_p(S, P_i)$ is related to the cost of service provided by the facility at S to meet the demand of the i-th customer.

Problem (1) has a long history of research [1]. The most common method for solving the problem is based on a one-point iterative algorithm, which was originally developed by Weiszfeld [2] in 1937 for Euclidian distances ($p = 2$). The method was rediscovered by Miehle [3] and Cooper [4] about 20 years later and by the first author of this paper about 25 years ago.

One can obtain expressions for the generalized Weiszfeld Iterative Algorithm (WIA) by writing the total weighted distance *W(x,y)*. Then the partial derivatives $\partial W(x, y)/\partial x$ and $\partial W(x, y)/\partial y$ are set to zero:

$$\sum_{i=1}^{n} w_i \operatorname{sign}(x - a_i) \frac{|x - a_i|^{p-1}}{\left[l_p(S, P_i)\right]^{p-1}} = 0; \sum_{i=1}^{n} w_i \operatorname{sign}(y - b_i) \frac{|y - b_i|^{p-1}}{\left[l_p(S, P_i)\right]^{p-1}} = 0.$$

After substituting $(x - a_i) = \operatorname{sign}(x - a_i)|x - a_i|$ and $(y - b_i) = \operatorname{sign}(y - b_i)|y - b_i|$ and isolating x and y, the obtained equations can be used iteratively to approach the optimal center location and the following formulas can be derived for the next (r+1) iteration [1, 10]:

$$x^{(r+1)} = \frac{\sum_{i=1}^{n} \frac{w_i a_i \left| x - a_i \right|^{p-2}}{l_p^{p-1}(S^{(r)}, P_i)}}{\sum_{i=1}^{n} \frac{w_i \left| x - a_i \right|^{p-2}}{l_p^{p-1}(S^{(r)}, P_i)}}; \qquad y^{(r+1)} = \frac{\sum_{i=1}^{n} \frac{w_i b_i \left| y - b_i \right|^{p-2}}{l_p^{p-1}(S^{(r)}, P_i)}}{\sum_{i=1}^{n} \frac{w_i \left| y - b_i \right|^{p-2}}{l_p^{p-1}(S^{(r)}, P_i)}} \tag{3}$$

where $S^{(r)} = (x^{(r)}, y^{(r)})$ is the point obtained in the previous iteration.

Although the single-facility minisum problem has been studied for many decades, the recent publications show that this problem is far from being exhausted [5-10]. Brimberg and Love [5, 6] recently proved the convergence of the generalized WIA with $1 \le p \le 2$. It was shown in [7] that the convergence in the case of $p > 2$ can be achieved by introducing a step size factor depending on *p*. The problem of singularities in the WIA is discussed for both Euclidian [8, 9] and l_p distances [10].

Accelerating the convergence of descent methods such as the WIA usually involves selecting alternate step sizes [11]. The first attempt was based on the Steffensen's iteration [12]. The method is not globally convergent, although it may be used to accelerate the local convergence for the WIA. At the same time, the acceleration effect achieved in [12] is reduced because the Steffensen's iteration involves additional complexity. Drezner [13] applies a variable factor λ to multiply the step size of the WIA for Euclidian distances ($p = 2$). In this case, the WIA is proven to converge only if $1 \le \lambda < 2$ [14]. At the same time, the value of λ, which is recalculated at each iteration, may exceed two [13]. It was also shown in [13] that the number of iterations produced by the variable λ method is insignificantly reduced, as compared to that yielded by the method with constant $\lambda = 1.8$. As recalculating λ at each iteration significantly increases the algorithm complexity, the overall acceleration, as compared to $\lambda = 1.8$, is questionable. Another approach [15] is based on the assumption that the differences between points obtained in two consecutive iterations form a geometric series, and the limit of this series is the next iterate. Despite some non-converging cases, which could be rectified by replacing a special parameter with the value corresponding to $\lambda = 1.8$, the procedure [15] yields a significant reduction in the number of iterations, as compared to $\lambda = 1.8$, for low values of *n*. At the same time, this approach almost doubles the complexity of each iteration, and so the overall acceleration is not clear. Acceleration of the generalized WIA for l_p distances was studied in [16, 17].

This paper is focused on developing an accelerated algorithm for the case of Euclidian distances ($p = 2$) and is a further development of the results obtained in [26]. The idea is to multiply the WIA iterate by a so-called feedback factor, which is the ratio of the current WIA iterate and the previous iterate value resulting from the accelerated algorithm. Our goal is to demonstrate that this method can reduce the number of iterations and be faster than the existing procedures for solving the minisum problem. As the complexity of this algorithm is very close to that of the WIA, it can easily be implemented in engineering practice. In addition, the proposed algorithm can be used in solving the Multi-Facility Location Problem (MFLP) since the single-facility minisum problem can be used as a subroutine to solve the MFLP. In this case, the single-facility

problem would be applied for each iteration and the overall complexity of the MFLP would strongly depend on the computational complexity of the single-facility problem.

2. Accelerated algorithm

To accelerate the WIA we multiply the Weiszfeld iterate by a feedback factor and, therefore, we will refer to this algorithm as Feedback Algorithm for Single Facility Location (FASFL). Feedback accelerators are thoroughly studied in the earlier publications of the first author [19-25]. We considered two kinds of approaches: first, when the factor is the same for both *x* and *y*; second, when the factors are different for *x* and *y*.

In the first case, the factor that gave the most acceleration was found to be:

$$\gamma^* = \left(\frac{f_w(x^{(r-1)}) + f_w(y^{(r-1)})}{x^{(r-1)} + y^{(r-1)}} \right)^t ,$$

where γ^* is the factor itself, $f_w(a)$ is the next WIA iterate for coordinate *a* determined by formula (3) for $p = 2$, *t* is some parameter, and *r* is the iteration number. Multiple calculations showed that there is no static *t* that always yields the least number of iterations. In fact, *t* varied from -1 to 4 or more. In addition, the acceleration increase was on the average less significant than for the approach described below.

In the second case, the factor that yielded the least number of iterations is expressed as

$$\gamma(X^{(r-1)}) = \left(\frac{f_w(X^{(r-1)})}{X^{(r-1)}} \right)^t ,$$

where $X^{(r-1)}$ is the coordinate (*x* or *y*) for the previous iterate. We experimentally found that *t* = 1 is the optimal value in all cases, except when the optimal center location coincides with one of the demand points (discussed later). Numerous computer experiments demonstrated that this method gives a significant acceleration. At each iteration, the WIA iterate is adjusted in the direction to the optimal location. The fact that factor γ has different values for the *x*- and *y*-coordinates allows the FASFL to follow a converging sequence that forms a curved trajectory. The accelerated next iterates are given by

$$x^*_{r+1} = \frac{\left(x^{(r+1)}\right)^2}{x^{(r)}}; y^*_{r+1} = \frac{\left(y^{(r+1)}\right)^2}{y^{(r)}} . \qquad (4)$$

Let us consider singularities and cases when the found location coincides with one of the demand points. Suppose we get an iterate equal to one of the demand points. Then division by zero takes place. Although this situation usually happens only for l_p norms and/or multi-facility problems, it may also occur, at least theoretically, in a single facility problem with Euclidian distances. There are several ways to deal with this singularity.

The most common approach is to use a hyperbolic approximation [1, 7, 17]. For Euclidian distances, we can easily avoid this singularity. Suppose that $S^{(r)} = P_k$. Expression (3) for $x^{(r+1)}$ can be rewritten as:

$$x^{(r+1)} = \frac{\sum_{\substack{i=1 \\ \neq k}}^{n} \frac{w_i a_i}{l_2(S^{(r)}, P_i)} + \frac{w_k a_k}{l_2(S^{(r)}, P_k)}}{\sum_{\substack{i=1 \\ \neq k}}^{n} \frac{w_i}{l_2(S^{(r)}, P_i)} + \frac{w_k}{l_2(S^{(r)}, P_k)}}.$$

Now we multiply both the numerator and denominator by $l_2(S^{(r)}, P_k)$ to avoid division by zero:

$$x^{(r+1)} = \frac{l_2(S^{(r)}, P_k) \times \sum_{\substack{i=1 \\ i \neq k}}^{n} \frac{w_i a_i}{l_2(S^{(r)}, P_i)} + w_k a_k}{l_2(S^{(r)}, P_k) \times \sum_{\substack{i=1 \\ i \neq k}}^{n} \frac{w_i}{l_2(S^{(r)}, P_i)} + w_k}.$$

Since the two sums are multiplied by zero, we have $x^{(r+1)} = a_k$. Obviously, the same applies to the *y*-coordinate, and correspondingly, $y^{(r+1)} = b_k$. The same approach can be extended to l_p distances.

Another special case may occur when the found center location coincides with one of the demand points. In this case, a special test is commonly used to find out whether the demand point is optimal [1]. However, if the point is not optimal, the test does not provide any clue as to what direction the algorithm should take to look for the optimal location. To this end, we developed a special subroutine, the so-called kick-out procedure, the pseudo-code for which is given below:

1) If $S = P_i$ then remove P_i and rerun the procedure.
2) If $S = P_i$ again then
    { P_i is optimal. stop.}
else
    {Consider a neighborhood of P_i:
    $(a_i + z, b_i + z)$; $(a_i + z, b_i - z)$; $(a_i - z, b_i + z)$; $(a_i - z, b_i - z)$;
    if $W(P_i)$ is smaller than *W* in each of the neighboring points then
        { P_i is optimal. stop.}
else
    {Suppose $S_{\min} = (f, g)$ - point corresponding to the smallest *W*.
    if $2f - a_i > 0$ then $x := 2f - a_i$; else $x := z$;

if $2g - b_i > 0$ then $y := 2g - b_i$; else $y := z$;

Restart the procedure from the beginning with the new starting values of x and y.}

}

Most of the steps above are self-explanatory. The final step (last "if-else" clause) is run to determine if P_i gives the minimal cost compared to its neighbors. If not, the next starting point is taken in the direction of the point that gave the least sum. It is noteworthy that one can achieve a reduction in the complexity of the comparison $S = P_i$ by presorting the demand points lexicographically and then applying binary search to compare the values of the x-coordinate for the i-th demand point and the current iterate.

In addition, a bound or stopping rule is required to terminate the FASFL. For this purpose, there exist various approaches, such as: rectangular bounding method [18], acceptable deviation from a calculated bound on the optimal value of the objective function [1], difference between two successive values of the objective function, etc. For simplicity, we will use the distance between two consecutive points as the stopping parameter, which will be referred to as ε.

The FASFL converged in all our calculations. However, in cases when the optimal location coincided with one of the demand points, the convergence took much longer than the WIA. We noticed that in these cases the procedure starts to oscillate around the optimal value while the amplitude of this oscillation around the optimal location decreases extremely slowly. This results in a slow convergence. To remedy this situation we introduced the following statements:

$$\text{if} \quad (x^*_{r+1} - x^*_r) \times (x^*_{r+2} - x^*_{r+1}) < 0 \ \text{ then } \ x^*_{r+2} := \frac{x^*_{r+1} + x^*_{r+2}}{2} \ ; \tag{5}$$

$$\text{if} \quad (y^*_{r+1} - y^*_r) \times (y^*_{r+2} - y^*_{r+1}) < 0 \ \text{ then } \ y^*_{r+2} := \frac{y^*_{r+1} + y^*_{r+2}}{2} \ ;$$

When an iterative procedure oscillates around the optimal point approaching it slowly, taking the average may be helpful to come faster to the optimal point. Our calculations show that this correction fixes the slow convergence issue and gives approximately the same number of iterations as the uncorrected algorithm for all other cases.

3. Computer experiments

To evaluate the performance of the FASFL, it was compared with the WIA and two acceleration procedures described below. Most of the existing accelerating approaches can be reduced to the following formula:

$$x' = x_0 + \lambda(x_1 - x_0) \tag{6}$$

where x' is the accelerated next iterate, λ is the acceleration factor, x_1 is the WIA next iterate, and x_0 is the previous iterate. It is easy to see that $\lambda = 1$ corresponds to the WIA approach. $\lambda = 1.8$ was found to be the optimal value when λ is constant [13]. Two accelerating procedures dealing with variable λ are reported in [13, 15]. The procedure

in [15] looks faster, simpler, and, therefore, more practical. So the FASFL will be compared with this procedure. It should be noted that every iteration in the procedure [15] requires calculation of two sequential iterates by formula (3). As a result, the complexity of every iteration is almost doubled. Consequently, to get the actual number of iterations, as compared to the FASFL, we have to multiply the number of iterations obtained in this approach by a factor of two. Mathematically, it will be expressed by introducing a parameter c related to complexity. Accordingly, for the procedure in [15] c is equal to 2 and 1 for all other algorithms, where the additional acceleration complexity can be neglected.

First, we considered a typical problem where random weights and coordinates were uniformly generated in a closed interval [0,1]. In all calculations, as well as for the uniform generation of random numbers, we used Mathematica 4.1. We took classical cases when n=5, 10, 50, 100, 500, and 1000 [13, 15]. For every n we ran one hundred problems. As the stopping parameter we used $\varepsilon = 10^{-5}$.

The results summarized in Table 1 show, that on the average, the FASFL provides better performance than any other procedure for any value of n. The acceleration is 5%-17% compared to the next fastest algorithm, where λ is constant and equal to 1.8. In the calculations, the FASFL converged to the same optimal points, as did the other procedures. Note that the FASFL acceleration compared to the WIA increases for higher n.

Table 1. Comparison of the accelerating procedures for points generated in [0,1]. Number of iterations.

Number of points	λ=1 (c=1)			λ=1.8 (c=1)			λ' (c=2)*			FASFL (c=1)		
	Min	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.
n=5	5	362	47.45	7	262	32.61	6	208	33.66	6	236	27.05
n=10	7	140	31.09	5	99	19.78	6	180	22.9	5	94	16.93
n=50	6	40	15.32	4	30	8.01	6	30	10.3	4	23	7.6
n=100	7	23	13.45	4	24	6.53	4	18	8.1	3	12	5.95
n=500	7	14	11.01	3	7	4.91	4	10	6.28	3	7	4.43
n=1000	7	15	10.25	3	7	4.66	4	8	5.9	3	7	3.96

* - c = 2 accounts for almost double complexity for each iteration

Then we considered another classical case, in which demand points are distributed randomly, using the uniform distribution function built in Mathematica 4.1, over a 100x100 square. The weights were randomly selected between 1 and 100. The same values of n and the number of problems for each n were taken. In this case, the stopping parameter $\varepsilon = 10^{-3}$ was used. The acceleration is 5%-22% compared to the next fastest algorithm, where λ is constant and equal to 1.8. The results summarized in Table 2 show exactly the same behavior as in Table 1: the FASFL is the fastest.

Table 2. Comparison of the accelerating procedures for the 100x100 square case. Number of iterations.

Number of points	λ=1 (c=1)			λ=1.8 (c=1)			λ' (c=2)*			FASFL (c=1)		
	Min	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.
n=5	6	288	40.9	6	184	29.53	6	338	30.5	6	171	23.14
n=10	7	111	26.1	5	71	16.36	6	118	18.96	4	64	13.79
n=50	10	29	15.5	4	22	7.91	6	32	10.18	5	21	7.54
n=100	9	24	13.4	4	25	6.7	6	28	8.58	3	20	6.29
n=500	8	15	11	3	6	4.89	4	12	6.18	3	7	4.46
n=1000	7	13	10.4	3	7	4.7	4	10	6	3	7	4.14

* - c = 2 accounts for almost double complexity for each iteration

4. Conclusions

The FASFL in which the current WIA iterate is multiplied by the feedback factor shows the better performance than the existing accelerating procedures for the single-facility minisum problem. Along with the subroutines developed to handle special cases, the algorithm is simple and can be used for a wide range of practical location problems: communication networks design, location of various service providing centers, location of emergency services, etc. In addition, the FASFL can be used as an efficient subroutine to solve the multi-facility location problem.

References

1. Love, R. F., Morris, J. G., & Wesolowsky, G. O. (1988). Facilities location, models and methods, North-Holland. New York.
2. Weiszfeld, E. (1937). "Sur la point pour lequel la somme de distances de n points donnés est minimum", Tôhoku Math. J., 43, 355-386.
3. Miehle, W. (1958). "Link-length minimization in networks", Operations Research, 6, 232-243.
4. Cooper, L. (1963). "Location-allocation problems", Operations Research, 11, 37-52.
5. Brimberg, J., & Love, R.F. (1992). "Local convergence in a generalized Fermat-Weber problem", Ann. Oper. Res., 40, 33-66.
6. Brimberg, J., & Love, R.F. (1993). "Global convergence in a generalized iterative procedure for the minisum location problem with l_p distances", Operations Research, 41, 1153-1163.
7. Uster, H., & Love, R.F. (2000). "The convergence of the Weiszfeld algorithm", Comput. Mathem. App., 40, 443-451.
8. Brimberg, J. (1995). "The Fermat-Weber location problem revisited", Math. Programming, 71, 71-76.
9. Chandrasekaran, R., & Tamir, A. (1989). "Open questions concerning Weiszfeld's algorithm for the Fermat-Weber location problem", Math. Programming, 44, 293-295.
10. Brimberg, J., & Chen, R. (1998). "A note on convergence in the single facility minisum location problem", Comput. Math. Appl., 35, 25-31.
11. Cohen, A. I. (1981). "Step size analysis for descent methods", J. Optim. Theory Appl., 33, 187-205.
12. Katz, I.N. (1974). "Local convergence in Fermat's problem", Math. Programming, 6, 89-104.
13. Drezner, Z. (1992). "A note on the Weber location problem", Ann. Oper. Res., 40, 153-161.
14. Ostresh, L. M. (1978). "On the convergence of a class of iterative methods for solving the Weber problem", Operations Research, 26, 597-609.
15. Drezner, Z. (1995). "A note on accelerating the Weiszfeld procedure", Location Science, 3, 275-279.
16. Frenk, J.B.G., Melo, M.T., & Zhang, S. (1994). "A Weiszfeld method for a generalized l_p distance minisum location model in continuous space", Location Science, 2, 111-127.

17. Brimberg, J., Chen, R., & Chen, D. (1998). "Accelerating convergence in the Fermat-Weber location problem", Oper. Res. Letters, 22, 151-157.
18. Uster, H., & Love, R. F. (2002). "A generalization of the rectangular bounding method for continuous location models", Comput. & Mathem. with Applic., 44, 181-191.
19. Verkhovsky, B. (1976). "Feedback algorithm for system of equations", IBM Technical Disclosure Bulletin, Vol.18, No.10.
20. Verkhovsky, B. (1976). "Algorithm for controlled feedback for system of equations with stochastic matrix". IBM Technical Disclosure Bulletin, Vol.18, No.10.
21. Verkhovsky, B. (1976). "Algorithm for system of equations with stochastic matrix", IBM Technical Disclosure Bulletin, Vol.18, No.10.
22. Verkhovsky, B. (1977). "Smoothing systems design and parametric markovian programming", Markov Decision Theory, Ed. H.C. Tijms and J. Wessels, Mathematische Centrum, Amsterdam, 105-117.
23. Verkhovsky, B. (1978). "Delinearization algorithms for elliptic partial differential equations", Research Report, 78-WR-3, Princeton University, 1978, 1-12.
24. Verkhovsky, B. (1976). "Algorithm with nonlinear acceleration for a system of linear equations", Research Report, No.76-WR-1, Princeton University.
25. Veroy (now Verkhovsky), B. (November 1985). "Projection algorithm for a stochastic dynamic programming problem", Proc. NEACP Technology & Science Conference, Newton, Massachusets.
26. Verkhovsky, B., and Polyakov, Y. (2002). "Accelerated algorithm for the single-facility minisum problem", Research Report, CS-02-05, New Jersey Institute of Technology.