ABSTRACT

FEEDBACK ALGORITHM FOR SWITCH LOCATION: ANALYSIS OF COMPLEXITY AND APPLICATION TO NETWORK DESIGN

by
Yuriy S. Polyakov

An accelerated feedback algorithm to solve the single-facility minisum problem is studied with application to designing networks with the star topology. The algorithm, in which the acceleration with respect to the Weiszfeld procedure is achieved by multiplying the current Weiszfeld iterate by an accelerating feedback factor, is shown to converge faster than the accelerating procedures available in the literature. Singularities encountered in the algorithm are discussed in detail. A simple practical exception handling subroutine is developed. Several applications of the algorithm to designing computer networks with the star topology are demonstrated. Applications of the algorithm as a subroutine for multi-switch location problems are considered. Various engineering aspects involved in acquiring and processing coordinates for geographic locations are discussed. A complete algorithm in pseudocode along with the source code listing in Mathematica 4.1 is presented.

FEEDBACK ALGORITHM FOR SWITCH LOCATION: ANALYSIS OF COMPLEXITY AND APPLICATION TO NETWORK DESIGN

by
Yuriy S. Polyakov

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

Department of Computer Science

May 2003

APPROVAL PAGE

**FEEDBACK ALGORITHM FOR SWITCH LOCATION:
ANALYSIS OF COMPLEXITY AND APPLICATION TO NETWORK DESIGN**

Yuriy S. Polyakov

Dr. Boris S. Verkhovsky, Thesis Advisor Professor of Computer Science and Director of Cryptography and Telecommunications Laboratory, Computer Science Department, CCS-NJIT	Date
Dr. James M. Calvin, Committee Member Associate Professor, Computer Science Department, CCS-NJIT	Date
Dr. James A.M. McHugh, Committee Member Professor and Chairman of the Computer Science Department, CCS-NJIT	Date

BIOGRAPHICAL SKETCH

Author:	Yuriy S. Polyakov
Degree:	Master of Science
Date:	May 2003
Date of Birth:	September 3, 1980
Place of Birth:	Moscow, Russia

Undergraduate and Graduate Education:

- Master of Science in Computer Science
  New Jersey Institute of Technology, Newark, NJ, 2003

- Bachelor of Science in Computer Information Systems
  Excelsior College, Albany, NY, 2002

- Associate of Science in Computer Technology and Systems Software
  Wenatchee Valley College, WA, 2001

Major: Computer Science

Presentations and Publications:

Verkhovsky, B. S & Polyakov, Yu. S. (2003, Jul. – Aug.). Algorithms for Optimal Switch Location: Concave Cost Functions. 15th Int'l Conf. on Systems Research, Informatics and Cybernetics. Baden-Baden, Germany (accepted).

Verkhovsky, B. S & Polyakov, Yu. S. (2003, Jul. – Aug.). Highly Efficient Algorithm for Two-Switch Location Problem. 15th Int'l Conf. on Systems Research, Informatics and Cybernetics. Baden-Baden, Germany (accepted).

Verkhovsky, B. & Polyakov Yu. (2002). Feedback algorithm for the single-facility minisum problem. Annals of the European Academy of Sciences, 127-136.

Polyakov, Yu. S., Kazenin, D. A., Maksimov, E. D., & Polyakov, S. V. (2003). Kinetic model of depth filtration with reversible adsorption. Theoretical Foundations of Chemical Engineering, 37 (to be published in No. 5).

Polyakov, Yu. S., Maksimov, E. D., & Polyakov, V. S. (1999). On the design of microfilters. Theoretical Foundations of Chemical Engineering, 33, 64-71.

To my beloved family

ACKNOWLEDGEMENT

The author wishes to express his sincere gratitude to his thesis advisor, Professor Boris S. Verkhovsky, for arousing interest to network topology design, guidance, and friendship. Special thanks to Dr. James M. Calvin and Dr. James A.M. McHugh for serving as members of the committee.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Location Science and Fermat-Weber Problem

Finding the optimal location of a switch on a network with star topology can be considered as a special case of much more general problems in operations research, usually referred to as facilities location problems. The problems address the question of where to locate an object (or objects) called a facility (switch, in our case). The facility will interact with a group of other objects that have fixed locations, called existing facilities (users, in our case). A concept of distance between the facility to be located and the existing facilities will contribute to a performance measure. This will lead to an objective function that can be used to evaluate a trial location of the facility. The choice of locations may be restricted.

Location science has been developing in three main areas. The first (Love *et al.*, 1988) and the oldest category includes the continuous models which allow facility locations to be anywhere on the plane or a subset of the plane. The second type (Love *et al.*, 1988) has evolved out of modern practice and stems from the application of mathematical programming to solving location-allocation problems. These are known as discrete models; the possible locations are specified in advance and are finite in number. The third type (Handler *et al.*, 1979) of location models is based on graph and network theory. In this thesis, an algorithm of the first category will be developed and extensively studied.

The location problems are usually divided in two classes: single-facility and multi-facility problems. Single-facility problems are used to locate a single facility in some configuration, for example, one data switch on a network. Multi-facility problems are more complex and involve two or more facilities to be simultaneously located. A multi-switch network is a good example of a case when a multi-facility problem could be successfully applied. Single-facility problems are often used as subroutines in solving multi-facility location-allocation problems (Brimberg *et al.*, 2000; Levin & Ben-Israel, 2001).

Another important classification of facilities location problems is based on the optimality criterion. Locations problems are usually either minisum or minimax. The first type of location problems is used if it is necessary to minimize the sum of weighted distances from a facility(ies) to a set of existing facilities. Locating a data switch or locating a warehouse for a multi-store company are situations when one deals with minisum problems. Minimax location problems are used if it is necessary to find such a position for a facility(ies) that the farthest distance from it(them) to any existing facility is minimal. Locating an emergency service when the latest time of arrival must be minimized is a good example of a minimax problem.

The optimal location of a data switch on a network can be found by solving the continuous single-facility minisum (also referred to as Fermat-Weber) problem, which is thoroughly studied in the literature (Love *et al.*, 1988). For the data switch case, the problem is to place a switch on the plane so as to minimize the sum of "weighted" distances from the switch to a set of fixed planar points (users). If the distances are given by the l_p norm, the problem is formulated as follows:

$$\min W\left(S\right)=\sum_{i=1}^{n} w_i l_p\left(S,P_i\right), \tag{1.1}$$

where *W(S)* is the total "weighted" distance;

where *n* is the number of fixed demand points (users);

w_i is the "weight" (demand) of the *i-th* user; $i=1,...,n$;

$P_i=(a_{i1},a_{i2})$ is the given location of the *i-th* user (demand point);

$S=(x_1,x_2)$ is the unknown location of the data switch.

The distance between any $S,P_i \in R^2$ is given by

$$l_p(S,P_i)=[|x_1-a_{i1}|^p+|x_2-a_{i2}|^p]^{1/p}, p\geq 1. \tag{1.2}$$

The "weight" is thought to be proportional to the demand of user *i* and incorporates cost, bandwidth, network flow, and other factors. $w_i l_p\left(S,P_i\right)$ is related to the cost of servicing a request of user *i* by the switch.

1.2 History of Fermat-Weber Problem

Problem formulated in Equation (1.1) has a long history of research (Love *et al.*, 1988; Zacharias, 1931). The first recorded efforts at solving locations problems are associated with Fermat in the early 17th century. In his essay on maxima and minima Fermat wrote, "Let he who does not approve of my method attempt the solution of the following problem: Given three points in the plane find a fourth point such that the sum of its distances to the three given points is a minimum". Before 1640 Torricelli observed that the circles circumscribing the equilateral triangles constructed on the sides of the triangle, exterior to the triangle, intersect at the optimal point. In 1834 Heinen proved that the Torricelli property was not general; if the three points result in a triangle with one angle equal to or greater than 120°, the vertex of this angle is the minimizing point. In his *Doctrine and Application of Fluxations* (1750), Simpson generalized the problem to obtaining the point that minimizes the "weighed" sum of distances from the three given points. Weber (1909) incorporated this problem into location theory in his influential treatise on the theory of industrial location.

This early research could not anticipate the iterative mathematical perspective enabled by the advent of the electronic computer. This fact explains why most of the research related to facility location is relatively recent, starting in 1950s – 1960s. The only exception is the monumental paper by E. Weiszfeld, now known as Andrew Vaszonyi, published in 1937 that became the basis for a lot of research in location science. The fact that the paper was written in French, submitted from Prague, and published in a Japanese journal left the results of the paper virtually unknown for over 20 years.

1.3 Weiszfeld Iterative Algorithm

The most common method for solving the problem is based on a one-point iterative algorithm, which was originally developed in Weiszfeld (1937) for Euclidian distances ($p = 2$). The method was rediscovered by Miehle (1958), Kuhn & Kuenne (1962), Cooper (1963), and Professor Verkhovsky (approx. 1978) many years later and has become one of the most frequently used methods to solve the single-facility minisum location problem.

One can obtain expressions for the Weiszfeld Iterative Algorithm (WIA) by writing the total weighted distance *W(S)* for *p*=2, which corresponds to straight-line or Euclidian distances. Then the partial derivatives $\partial W(S) / \partial x_k$, where *k* = 1 or 2, are set to zero:

$$\sum_{i=1}^{n} \frac{w_i(x_k - a_{ik})}{l_2(S, a_{ik})} = 0;$$

After rewriting, 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) iterate (Love *et al.*, 1988; Weiszfeld, 1937):

$$x_k^{(r+1)} = \frac{\sum_{i=1}^{n} \frac{w_i a_{ik}}{l_2(S^{(r)}, P_i)}}{\sum_{i=1}^{n} \frac{w_i}{l_2(S^{(r)}, P_i)}}; \tag{1.3}$$

where $S^{(r)} = (x_1^{(r)}, x_2^{(r)})$ is the point obtained in the previous iteration.

The WIA is readily extended to l_p distances (Love *et al.*, 1988; Brimberg & Chen, 1998). Differentiating *W(S)* with respect to x_k and setting the partial derivatives to zero, one obtains

$$\sum_{i=1}^{n} w_i \operatorname{sign}\left(x_k - a_{it}\right) \frac{\left|x_k - a_{ik}\right|^{p-1}}{\left[l_p\left(S^{(r)}, P_i\right)\right]^{p-1}} = 0.$$

After substituting $\left(x_k - a_{ik}\right) = \operatorname{sign}\left(x_k - a_{ik}\right)\left|x_k - a_{ik}\right|$ and isolating x_k on the left-hand side, one has

$$x_k^{(r+1)} = \frac{\sum_{i=1}^{n} \frac{w_i a_{ik} \left|x_k - a_{ik}\right|^{p-2}}{\left[l_p(S^{(r)}, P_i)\right]^{p-1}}}{\sum_{i=1}^{n} \frac{w_i \left|x_k - a_{ik}\right|^{p-2}}{\left[l_p(S^{(r)}, P_i)\right]^{p-1}}}; \tag{1.4}$$

Equation (1.4) is usually referred to as the generalized Weiszfeld procedure (Brimberg & Chen, 1998). Sometimes, a hyperbolic approximation is used to avoid division by zero and thus Equation (1.4) has a slightly different look (Love *et al.*, 1988):

$$x_k^{(r+1)} = \frac{\displaystyle\sum_{i=1}^{n} \frac{w_i a_{ik}}{d'(S^{(r)}, P_i)\, d''(x_k^{(l)}, a_{ik})}}{\displaystyle\sum_{i=1}^{n} \frac{w_i}{d'(S^{(r)}, P_i)\, d''(x_k^{(l)}, a_{ik})}},$$

where

$$d'\left(S^{(r)}, P_i\right) = \left[\left\{\left(x_1 - a_{i1}\right)^2 + \delta\right\}^{p/2} + \left\{\left(x_2 - a_{i2}\right)^2 + \delta\right\}^{p/2}\right]^{1-1/p}, \text{ and}$$

$$d''\left(x_k^{(r)}, a_{ik}\right) = \left\{\left(x_k - a_{ik}\right)^2 + \delta\right\}^{1-p/2}, \text{ and}$$

δ is a small positive number.

A detailed study of this singularity and hyperbolic approximation is provided later in the thesis.

A lot of research has been conducted to study various aspects and improve the WIA. However, the recent publications show that this problem is far from being exhausted. Brimberg and Love (1992, 1993) recently proved the convergence of the generalized WIA with $1 \le p \le 2$. It was shown in Uster *et al.* (2000) 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 (Brimberg, 1995; Chandrasekaran *et al.*, 1989) and l_p distances (Brimberg & Chen, 1998).

1.4 Newton Bracketing Method

An alternative approach to solve the problem formulated in Equation (1.1) is the Newton Bracketing method (Levin & Ben-Israel, 2002b). In contrast to the Weiszfeld procedure that approximates the value of the optimal location, the Newton Bracketing approach, based on the directional Newton method (Levin & Ben-Israel, 2002a), works by narrowing the bounds on the minimum value of *W(S)*.

An iteration begins with an interval $\left[L^{(r)}, U^{(r)}\right]$, called a *bracket*, containing the minimum value of *W(S)*, which will be referred to as $W_{\min}$; that is, $L^{(r)} \le W_{\min} \le U^{(r)}$. If the bracket is sufficiently small, that is, $U^{(r)} - L^{(r)} \le \rho$ for an acceptable tolerance $\rho > 0$, then $S^{(r)}$ is declared optimal and the computation stops. Else, value $M^{(r)}$ inside the bracket is selected, such as

$$M^{(r)} = \sigma U^{(r)} + (1-\sigma) L^{(r)}, \text{ for some } 0 < \sigma < 1,$$

and one directional Newton iteration (Levin *et al.*, 2002b) is applied for solving

$$W(S) = M^{(r)}$$

to obtain

$$S^{(r+1)} = S^{(r)} - \frac{W\left(S^{(r)}\right) - M^{(r)}}{\left\|\nabla W\left(S^{(r)}\right)\right\|^2} \nabla W\left(S^{(r)}\right).$$

After that two cases are possible

1) If $W\left(S^{(r+1)}\right) < W\left(S^{(r)}\right)$, then $U^{(r+1)} := W\left(S^{(r+1)}\right)$, $L^{(r+1)} := L^{(r)}$;

2) If $W\left(S^{(r+1)}\right) \ge W\left(S^{(r)}\right)$, then $L^{(r+1)} := M^{(r)}$, $U^{(r+1)} := U^{(r)}$, $S^{(r+1)} := S^{(r)}$.

The initial bracket $\left[L^{(0)}, U^{(0)}\right]$ is given by:

$U^{(0)} = S^{(0)}$, where $S^{(0)}$ is the initial iterate, and

$L^{(0)} = l_2(P_i, P_j)\min\{w_i, w_j\}$, for any two demand points.

Information on when the Newton Bracketing method can be applied and other further details specific to this approach can be found in Levin & Ben-Israel (2002a, 2002b).

1.5 Acceleration of the Weiszfeld Iterative Algorithm

Accelerating the convergence of descent methods such as the WIA usually involves selecting alternate step sizes (Cohen, 1981). The first attempt was based on the Steffensen's iteration (Katz, 1974). The method is not globally convergent, though it may be used to accelerate the local convergence for the WIA. At the same time, the acceleration effect achieved in Katz (1974) is reduced because the Steffensen's iteration involves additional complexity.

Drezner (1992) 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 $1 \le \lambda < 2$ (Ostresh, 1978). At the same time, the value of λ, which is recalculated at each iteration, may exceed *two* (Drezner, 1992). It was also shown in Drezner (1992) 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 (Drezner, 1995) 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 (Drezner, 1995) 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 (Frenk *et al.*, 1994; Brimberg *et al.*, 1998).

1.6 Objectives of the Thesis

This thesis is focused on studying an accelerated algorithm for the case of Euclidian distances ($p = 2$) and is a further development of the ideas suggested by Professor Verkhovsky and the results obtained in Verkhovsky and Polyakov (2002a, 2002b). The idea is to multiply the current 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. The 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 iteration complexity of this algorithm is very close to that of the WIA, it can easily be used in practical applications, specifically in finding the optimal location of a data switch on a network with the star topology. Several possible computer-related applications of the studied algorithm are also discussed. In addition, the algorithm is applied to solving the Multi-Switch Location Problem (MSLP) since the single-switch location problem can be used as a subroutine to solve the MSLP. In this case, the single-switch problem is applied for each iteration and the overall complexity of the MSLP strongly depends on the computational complexity of the single-switch problem.

CHAPTER 2

FEEDBACK ALGORITHM FOR SINGLE-SWITCH LOCATION PROBLEM

2.1 Acceleration Technique

To accelerate the WIA, the Weiszfeld iterate is multiplied by a feedback factor and, therefore, this algorithm will be referred to as Feedback Algorithm for Switch Location (FASL). Feedback accelerators are thoroughly studied in Verkhovsky (1976a, 1976b, 1976c, 1976d, 1977, 1978) and Veroy (1985).

Two kinds of approaches are considered here (Verkhovsky & Polyakov, 2002a; 2002b): first, if the factor is the same for both x_1 and x_2; second, if the factors are different for x_1 and x_2.

In the first case, proposed by Professor Verkhovsky, the factor that gave the most acceleration was found to be

$$\gamma^* = \left(\frac{f_w(x_1^{(r-1)}) + f_w(x_2^{(r-1)})}{x_1^{r-1} + x_2^{r-1}} \right)^t ,$$

where γ^* is the factor itself,

$f_w(a)$ is the next WIA iterate for coordinate *a* determined by Equation (1.3),

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. One of the calculations clearly demonstrating absence of a static value for optimal t is presented in Table 2.1.

Table 2.1 Optimal Value of t When the Same Factor is Used for Both Coordinates

Number of demand points	1st run	2nd run	3rd run
10	0.75	>2	-0.25
15	1.25	1.5	0.75
20	1	>2	0
25	1.25	1.5	0.75

Intervals for random number generation: [0,200] for a_1, [0,100] for a_2 and (0,100] for w.

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_1 or x_2) for the previous iterate. It was experimentally found that t approximately equal to one (it was always in the range from 0.96 to 1.07) 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 FASL to follow a converging sequence that forms a curved trajectory. The accelerated next iterates are given by

$$x_k^* = \frac{\left(x_k^{(r+1)}\right)^2}{x_k^{(r+1)}}. \tag{2.1}$$

In all further calculations presented in this thesis Equation (2.1) will be used. The important feature of this acceleration method is its simplicity, which should make it attractive for various implementations that involve solving the single-facility minisum problem.

2.2 Special Cases in the FASL and Behavior of the Cost Function

Singularities and cases when the found location coincides with one of the demand points are now considered. Suppose an iterate equal to one of the demand points is obtained. Although this situation usually takes place only for 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 (Love at al., 1988; Uster & Love, 2000).

In this case, each absolute term $|q|$ in Equation (1.2) is replaced with $\left(q^2+\delta\right)^{1/2}$, where δ is a small positive number. The approximation is always larger than the original term, but approaches the original term as $\delta \to 0$. Approximated Equation (1.2) is described as follows:

$$l_p^*(S,P_i)=\left[\left((x_1-a_{i1})^2+\delta\right)^{p/2}+\left((x_2-a_{i2})^2+\delta\right)^{p/2}\right]^{1/p}, p\geq 1,$$

and the problem formulated in Equation (1.1) for *WH(S),* the approximated total "weighted" distance, can be formulated as

$$\min WH\left(S\right)=\sum_{i=1}^{n} w_i l_p^*\left(S,P_i\right)$$

Difference between *WH(S)* and *W(S)* satisfies the following inequality (Love *et al.*, 1988):

$$\max_X \{WH(X) - W(X)\} \le \Delta(\delta) = 2^{1/p}\delta^{1/2}\left(\sum_{i=1}^{n} w_i\right),$$

where X is a coordinate.

For Euclidian distances, this singularity can easily be avoided using the idea suggested by Professor Verkhovsky in Verkhovsky & Polyakov (2002b). Suppose that $S^{(r)} = P_m$. Equation (1.3) for $x_k^{(r+1)}$ can be rewritten as:

$$x_k^{(r+1)} = \frac{\sum\limits_{\substack{i=1\\ \neq m}}^{n} \frac{w_i a_{ik}}{l_2(S^{(r)}, P_i)} + \frac{w_m a_{mk}}{l_2(S^{(r)}, P_m)}}{\sum\limits_{\substack{i=1\\ \neq m}}^{n} \frac{w_i}{l_2(S^{(r)}, P_i)} + \frac{w_m}{l_2(S^{(r)}, P_m)}}.$$

This expression can then be represented in the following equivalent form:

$$x_k^{(r+1)} = \frac{l_2(S^{(r)}, P_m) \times \sum\limits_{\substack{i=1\\ \neq m}}^{n} \frac{w_i a_{ik}}{l_2(S^{(r)}, P_i)} + w_m a_{mk}}{l_2(S^{(r)}, P_m) \times \sum\limits_{\substack{i=1\\ \neq m}}^{n} \frac{w_i}{l_2(S^{(r)}, P_i)} + w_m}.$$

If $l_2(S^{(r)}, P_m)$ is equal to zero, one has $x_k^{(r+1)} = a_{mk}$. 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 (Love *et al.*, 1988).

According to the test, *W(S)* is minimized at the s^{th} existing facility location $\left(a_{s1}, a_{s2}\right)$ if, and only if (Love *et al.*, 1988):

$$\left[\left(\sum_{\substack{i=1\\ \neq s}}^{n} \frac{w_i\left(a_{s1}-a_{i1}\right)}{l_2\left(a_s, P_i\right)}\right)^2 + \left(\sum_{\substack{i=1\\ \neq s}}^{n} \frac{w_i\left(a_{s2}-a_{i2}\right)}{l_2\left(a_s, P_i\right)}\right)^2\right]^{1/2} \leq w_s .$$

However, if the point is not optimal, this test does not provide any clue as to what direction the algorithm should take to look for the optimal location. To this end, Professor Verkhovsky (Verkhovsky & Polyakov, 2002b) 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 $P_i$:
        $(a_{i1}+z, a_{i2}+z)$; $(a_{i1}+z, a_{i2}-z)$; $(a_{i1}-z, a_{i2}+z)$; $(a_{i1}-z, a_{i2}-z)$;
        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 $2f - a_{i1} > 0$ then $x := 2f - a_{i1}$; else $x := z$;
                if $2g - a_{i2} > 0$ then $y := 2g - a_{i2}$; 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.

As demonstrated in (Brimberg & Chen, 1998), it is very unlikely for a current iterate to coincide with one of the demand points. This event becomes less and less probable with increasing the accuracy of calculations. To gain a better understanding of when and why singularities may occur, one can look at a three-dimensional plot of the total “weighted” cost *W(S)*. If $1 \le p \le 2$, l_p norm is a convex function and it can easily be shown that *W(S)* is also a convex function (Love *et al.*, 1988). As the result, it has one global minimum and thus looks like on the three-dimensional plot in Figure 1.1 (it is an actual plot for some randomly generated input data). The only time the WIA (and thus the FASL) does not converge (Love *et al.*, 1988; Weiszfeld, 1937) is when the iterate, on its way to the global minimum, coincides with one of the demand points and the gradient is undetermined. When one deals with real values, the probability of this singularity is related to the accuracy of computations: the higher the accuracy, the less is the probability. Therefore this event can be expected to occur very rarely, for example, if double precision arithmetic is used.

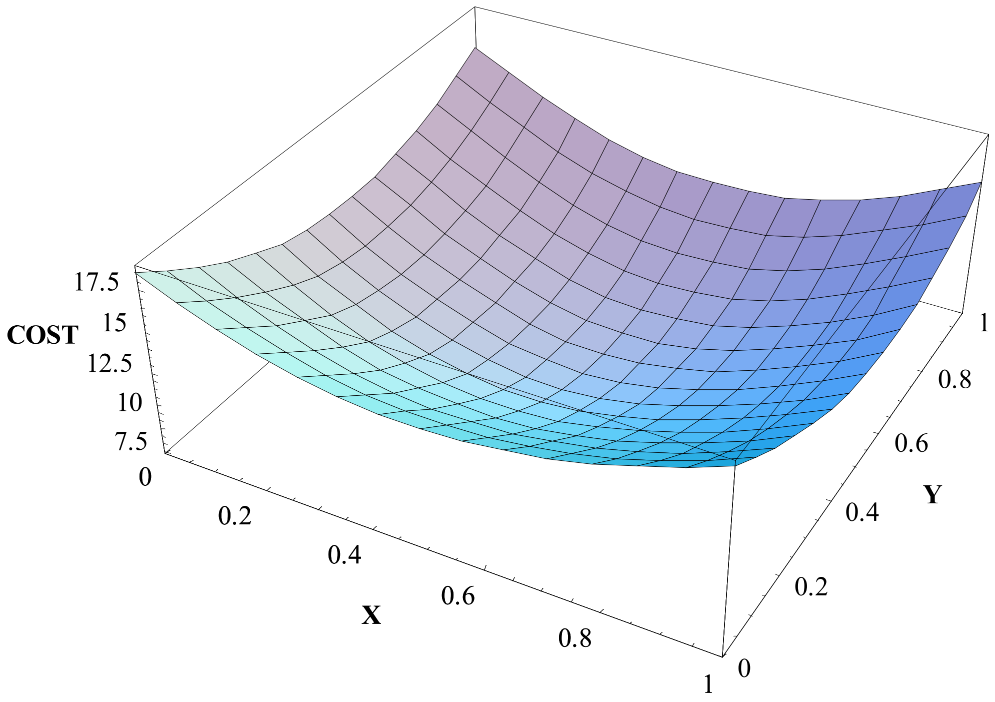

Figure 2.1 Total “weighted” cost function for a set of randomly generated points with associated random weights.

While the subroutines discussed above may be more useful in situations when the singularity is fairly common, for example, in a multi-facility problem, for this case a simpler idea can be used. The idea is to trap division by zero; and if that happens, to restart the FASL with new values for the initial coordinates. Adding some positive constants to both the *x*- and *y*- coordinates should avoid the singularity in the next run. This idea can be applied over and over until starting points that avoid the singularity are found. Since this event is very unlikely, the normal FASL would run in a great majority of the cases. Therefore, almost no extra complexity to handle these special cases will be introduced.

In addition, a bound or stopping rule is required to terminate the FASL. For this purpose, there exist various approaches, such as: rectangular bounding method (Uster & Love, 2002), acceptable deviation from a calculated bound on the optimal value of the objective function (Love *et al.*, 1988), difference between two successive values of the objective function etc. For simplicity, the distance between two consecutive points will be used as the stopping parameter, which will be referred to as ε.

The studied FASL converged in all the calculations that were run to experimentally check the convergence of the accelerated algorithm. However, in cases when the optimal location coincided with one of the demand points, the convergence took much longer than the WIA. It was 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, the following statements were introduced:

$$\textbf{if } (x_k^{(r-1)} - x_k^{(r-2)}) \times (x_k^{(r)} - x_k^{(r-1)}) < 0 \textbf{ then } x_k^{(r)} := \frac{x_k^{(r-1)} + x_k^{(r)}}{2} . \tag{2.2}$$

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. Calculations showed that this correction fixes the slow convergence issue and gives approximately the same number of iterations as the uncorrected algorithm for all other cases. In addition, many calculations were run where the optimal location for a specific configuration was calculated using both the corrected FASL and the built-in Mathematica 4.1 function **FindMinimum,** an alternative (non-WIA) procedure to find local minimums, which uses various methods due to Brent: the conjugate gradient in one dimension, and a modification of Powell's method in several dimensions. No noticeable deviations in the obtained results were found.

2.3 Complete Version of the FASL

The complete algorithm for the FASL listed in Appendix A is now studied. It is a common practice to start the Weiszfeld-like procedures at the center of gravity for a particular system (Love *et al.*, 1988). Thus steps 1 and 2 are used to initialize the coordinates using this idea. In step 3 the values for these coordinates are copied into some local variables x and y. Step 4 is used to set the iteration counter to zero.

Next goes the code that is run for every iteration. The first part of every iteration is the *for* loop that is used to calculate the next WIA iterate for a given point. This step exactly follows the Equation (1.3) broken into simple parts. However, inside that loop the exception handling, discussed in detail in the previous section, is also run. The idea is to add some small numbers δ_1 and δ_2 to the starting values x_1 and x_2 and rerun the procedure if division by zero is encountered. In step 14 the values for the coordinates obtained in the iteration $V - 2$ are saved. These values are then later used to handle the situations when the iterate oscillates around a demand point (discussed above). Step 15 is used to save the values obtained in the iteration $V - 1$. In step 16 the acceleration technique described in Equation (2.1) is applied. As the stopping criterion, the Euclidian difference between the current iterate and the previous one was used. If that difference is less than the specified accuracy z, at step 17 the algorithm stops and the optimal location value is displayed. And the final step in every iteration is accelerating the convergence of the

FASL in cases when the iterate oscillates around the demand point. Equation (2.2) is applied in this case.

2.4 Source Code for the FASL Implementation in Mathematica 4.1

The full source code for the FASL and initialization of the variables is provided in Appendix B. In this program the output is written to both the monitor and a text file. In the calculations the built-in **N** function, which gives the real value of an expression using a machine precision, was used. The machine precision is usually 16 digits (double precision).

The program starts with opening the output file. Then the FASL itself is defined. The code looks almost exactly the same as the algorithm discussed in the previous section. The only differences are related to specifics of the Mathematica programming language. To catch the division by zero exception, built-in **Check** function with **Power::infy** as the handled exception is used. If the exception occurs, a special message is displayed saying that a division by zero occurred and that the procedure is about to restart with a new starting point. After that, a variable **iDone**, which controls whether the algorithm has found the optimal point, is set to 0. If no exception occurs, the FASL continues in normal mode. It is noteworthy that the built-in Mathematica **Module** function is used to declare local variables.

After the FASL definition, the program runs the initialization part. For anybody interested in running more experiments, several lines that are used to randomly generate a set of two-dimensional points and weights associated with them are included. At the end of the initialization portion, the coordinates for the center of gravity are calculated using the built-in function **Sum**.

As seen at the end of the source code, the FASL is run in a **While** loop, and is executed only once if no division by zero occurs. However, if the exception occurs, the **iDone** variable gets reset to 0 (see above) and the FASL is rerun starting with a new initial point.

CHAPTER 3

COMPLEXITY AND CONVERGENCE ANALISYS

3.1 Other Acceleration Methods

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

$$x'_k = x_k^{(r-1)} + \lambda(f_w(x_k^{(r-1)}) - x_k^{(r-1)}) \tag{3.1}$$

where x'_k is the accelerated next iterate, λ is the acceleration factor, $x_k^{(r-1)}$ is the previous iterate, and $f_w(x_k^{(r-1)})$ is the WIA next 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 (Drezner, 1992). Two accelerating procedures dealing with variable λ are reported in (Drezner, 1992; 1995). The procedure in Drezner (1995) looks faster, simpler, and, therefore, more practical. Therefore the FASL will be compared with this procedure. Before comparing the results of the calculation series with the FASL, the procedure (Drezner, 1995) is briefly described.

This acceleration method is based on Aitken's Δ^2 Process (Burden *et al.*, 1981) and an accelerated next iterate $x_k^{(r^*)}$ is described using the following formula:

$$x_k^{(r^*)} = x_k^{(r-1)} - \frac{(x_k^{(r)} - x_k^{(r-1)})^2}{x_k^{(r+1)} + x_k^{(r-1)} - 2x_k^{(r)}}, \tag{3.2}$$

where

$x_k^{(r-1)}$ is the previous iteration;

$x_k^{(r)}$ is the next iterate calculated using the WIA, = $f_w(x_k^{(r-1)})$;

$x_k^{(r+1)}$ is the next iterate after $x_k^{(r)}$ calculated using the WIA, and is equal to $f_w(x_k^{(r)})$;

For consistency, it is better to write Equation (3.2) the following way

$$x_k^{(r*)} = x_k^{(r-1)} + \frac{1}{1-\theta}(x_k^{(r)} - x_k^{(r-1)}),$$

where $\theta = \left(x_k^{(r+1)} - x_k^{(r)}\right)/\left(x_k^{(r)} - x_k^{(r-1)}\right)$, which brings it to the standard look of Equation (3.1).

It should be noted that every iteration in the procedure described in Equation (3.2) requires calculation of two sequential iterates by Equation (1.3). As the result, the complexity of every iteration is almost doubled. Consequently, to get the actual number of iterations, as compared to the FASL, the number of iterations obtained in this approach has to be multiplied by a factor of *two*. Mathematically, it will be expressed by introducing a parameter c related to complexity. Accordingly, for the procedure formulated in Equation (3.2) c is equal to 2 and 1 for all other algorithms, where the additional acceleration complexity can be neglected.

3.2 Computer Experiments

First, a typical problem where weights and coordinates were uniformly generated in a closed interval [0,1] was considered. In all calculations, as well as for the uniform generation of random numbers, Mathematica 4.1 was used. Classical cases when n=5, 10, 50, 100, 500, and 1000 (Drezner, 1992; 1995) were taken. For every n one hundred problems was run, which should be enough to analyze the overall behavior and trends. As the stopping parameter $\varepsilon = 10^{-5}$ was used.

Table 3.1 Comparison of the Accelerating Procedures for Points Generated in [0,1] in Terms of the Number of Iterations

Number of points	λ=1 (c=1)			λ=1.8 (c=1)			λ' (c=2)*			FASL (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.

The results summarized in Table 3.1 show that, on the average, the FASL 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. It is noteworthy that the acceleration of the FASL in Table 3.1 compared to the classical Weiszfeld procedure, that is $\lambda = 1$, increases for higher n. Thus at this point it can be suggested that the more is the number of demand points, the better is the acceleration the FASL provides. In all the calculations, the FASL converged to the same optimal points as did the other procedures.

Then another classical case was considered, in which demand points were 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.

Table 3.2 Comparison of the Accelerating Procedures for the 100x100 Square Case in Terms of the Number of Iterations

Number of points	λ=1 (c=1)			λ=1.8 (c=1)			λ' (c=2)*			FASL (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

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 3.2 show exactly the same behavior as in Table 3.1: the FASL is the fastest. In addition, the same trend, the more is the number of demand points, the better is the acceleration provided by the FASL, is also true. The magnitude of acceleration compared to other methods is approximately the same as in Table 3.1. Subsequently there are enough grounds to say that the behavior demonstrated in Tables 3.1 and 3.2 should hold true for most of the cases. Again, in all the calculations, the FASL converged to the same optimal points as did the other procedures. Therefore in all 1200 cases the FASL demonstrated good and fast convergence.

CHAPTER 4

APPLICATIONS OF THE FASL TO NETWORK DESIGN

The single-facility problem can be used in many applications. Finding the optimal location of a warehouse for a multi-store company, location of various service providing centers, location of emergency services are among many possible implementations for the FASL. In this thesis, the focus will be placed on the applications related to communication networks design.

4.1 Minimizing Transmission Costs

A straight-forward implementation is minimizing cable length on a network. For simplicity, a simple one-floor Ethernet CAT 5 network with attic space above the ceiling is considered. The objective is to find such a position for a hub or switch that will minimize the cable length used. Places for network connections and the number of connections at each place are given.

This problem can be easily solved using the FASL. At first, the coordinates of each of the demand points where network connections need to be present are obtained. The weights are set based on the number of connections per each demand point. For example, if two connections are coming to a demand point *i*, the weight in this case would be equal to 2, that is $w_i = 2$. Since there are no significant constraints to paths of the cables in the attic space, this problem can be reduced to straight-line distances, that is $p = 2$. The objective is to minimize the $\sum_{i=1}^{n} w_i l_2(S, P_i)$, which is exactly what the FASL was designed for. In practical settings, other factors, like power source availability, vertical distance from the ceiling to a plate in a wall for specific connections, would have to be considered. However, the idea would still be the same.

Problems, like the one mentioned above, are usually not cost-efficient because CAT 5 cable costs are very low compared to the labor costs. However, the same strategy with partially different "weight" criteria can be applied to other related more practical situations:

- statewide, nationwide, or global fiber buildouts of various network infrastructures where too much extra cable could be very expensive;
- finding an optimal position for a fiber switch connecting several buildings using fiber optic cables;
- finding an optimal position of a cable company data switch used to provide TV and Internet access to the surrounding area (the idea is to maximize the number of residential and business demand points for a specific switch, that is to minimize cable distances for the bulk of the potential customers);
- minimizing cable lengths when dealing with distances, where attenuation effects become significant.

4.2 Minimizing Link Costs:

Application of the FASL to Designing Large Networks

Another very important application of the FASL is minimization of link costs in designing an interstate, nationwide, or a global network where each cable spans many miles. A hypothetical problem is now considered. The data provided here is for illustration purposes only, but the ideas can be applied to many situations, as described at the end of the section. A company is designing a nationwide WAN and has decided in which cities access to the WAN will be provided. The objective is to find an approximately optimal position for the data center, so that the costs would be minimized. The data center would be placed in a big city that is nearest to the calculated optimal position to provide better human and technical resource availability. It may be possible that the data center location would coincide with one of the WAN access points. A simplified drawing on the next page shows the cities that will need to have access to the data center.

Speeds available for the connections to the data center and their prices per mile:

ISDN: 1.5$/month;

T1: 7$/month;

T3: 30$/month.

Suppose that P1, P6, P7, P10, and P13 need to have T1 connections. P2, P3, and P4 need to have ISDN connection. All other demand points are going to use T3. Offices P6, P7, P11, and P12 need to have redundant connections to the data center. The weight for an ISDN connection will be considered as the unit weight. Then weights for T1 and T3 are 4 and 20 correspondingly.

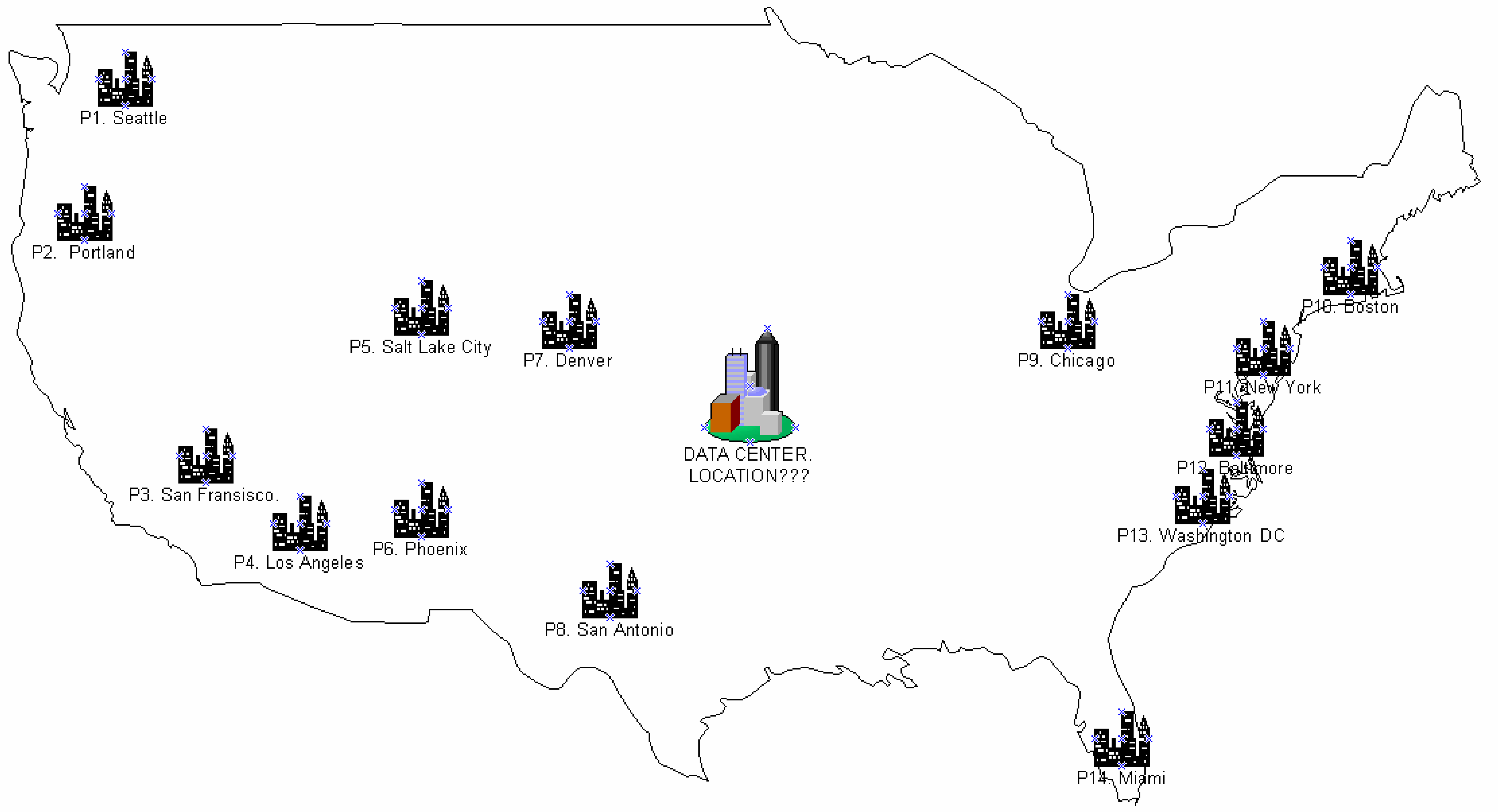

Figure 4.1 Map describing a problem where a data center location needs to be found for a nationwide WAN.

The table for all the demand points is given below.

Table 4.1 Distance and Weight Table for All the Demand Points

City	Code	Weight	Lat	Long	V	H
Seattle	P1	4	47.606ºN	122.331ºW	6337	8896
Portland	P2	1	45.524ºN	122.675ºW	6799	8915
San Francisco	P3	1	37.775ºN	122.418ºW	8495	8721
Los Angeles	P4	1	34.052ºN	118.243ºW	9212	7877
Salt Lake City	P5	20	40.761ºN	111.890ºW	7576	7066
Phoenix	P6	8	33.448ºN	112.073ºW	9133	6748
Denver	P7	8	39.739ºN	104.984ºW	7501	5897
San Antonio	P8	20	29.424ºN	98.493ºW	9226	4063
Chicago	P9	20	41.850ºN	87.650ºW	5994	3425
Boston	P10	4	42.358ºN	71.060ºW	4422	1249
New York	P11	40	40.714ºN	74.006ºW	5003	1405
Washington D.C.	P12	40	38.895ºN	77.037ºW	5623	1583
Baltimore	P13	4	39.290ºN	76.613ºW	5399	1653
Miami	P14	20	25.774ºN	80.194ºW	8351	528

In cases where redundant connections are necessary, the corresponding weights are multiplied by a factor of two. Now one gets to the most important question in this problem: dealing with coordinates.

It is relatively simple to find approximate Lat/Long coordinates of any point on the globe. To do that, one can use a simple mapping software, for example, Microsoft Streets & Trips. For this problem only free online tools are used, so that this experiment could be easily recreated. www.topozone.com was used as the source for the Lat/Long coordinates. This site gives the coordinates for all cities in the US; and since extreme accuracy is not required, the coordinates of the downtowns would be good enough for this problem. However, the problem that is faced is how to use those coordinates for the calculations?

The Earth is known to have a curved surface. Since Lat/Long system is highly dependent on these curvatures, neither latitude and longitude, nor its direct coordinate transformations, can be used in the calculations. Fortunately, in 1950s Mr. Jay K. Donald at AT&T Long Lines developed a V&H (Vertical and Horizontal) coordinate system to simplify rating and billing associated with Private Line Service and Message Telecommunications Service. He used a complex algorithm that projects the curvature of the Earth onto a flat plane. As the result he created a 10,000 x 10,000 grid, so called "V&H grid", for North America. A map showing this grid is displayed on Figure 4.2. Later this V&H coordinate system became a de facto standard in the telecommunications industry. More information on the V&H coordinate system can be found in Vertical and Horizontal Coordinates (2003).

There are many software packages that convert from Lat/Long to V&H and vice versa: Telcordia™ V & H Calculation (VHCALC) at http://www.trainfo.com/products_services/tra/catalog_details.html#V%20&%20H%20Calculation%20(VHCALC), V&H Tools by Stopwatch Maps at http://www.stopwatchmaps.com/services/products/software/vhtools.htm, and others. For this experiment, a free online conversion tool available at http://www.tuketu.com/dsl/lat-lon-form.htm was used. The calculated V&H coordinates for the cities of the problem are given in Table 4.1.

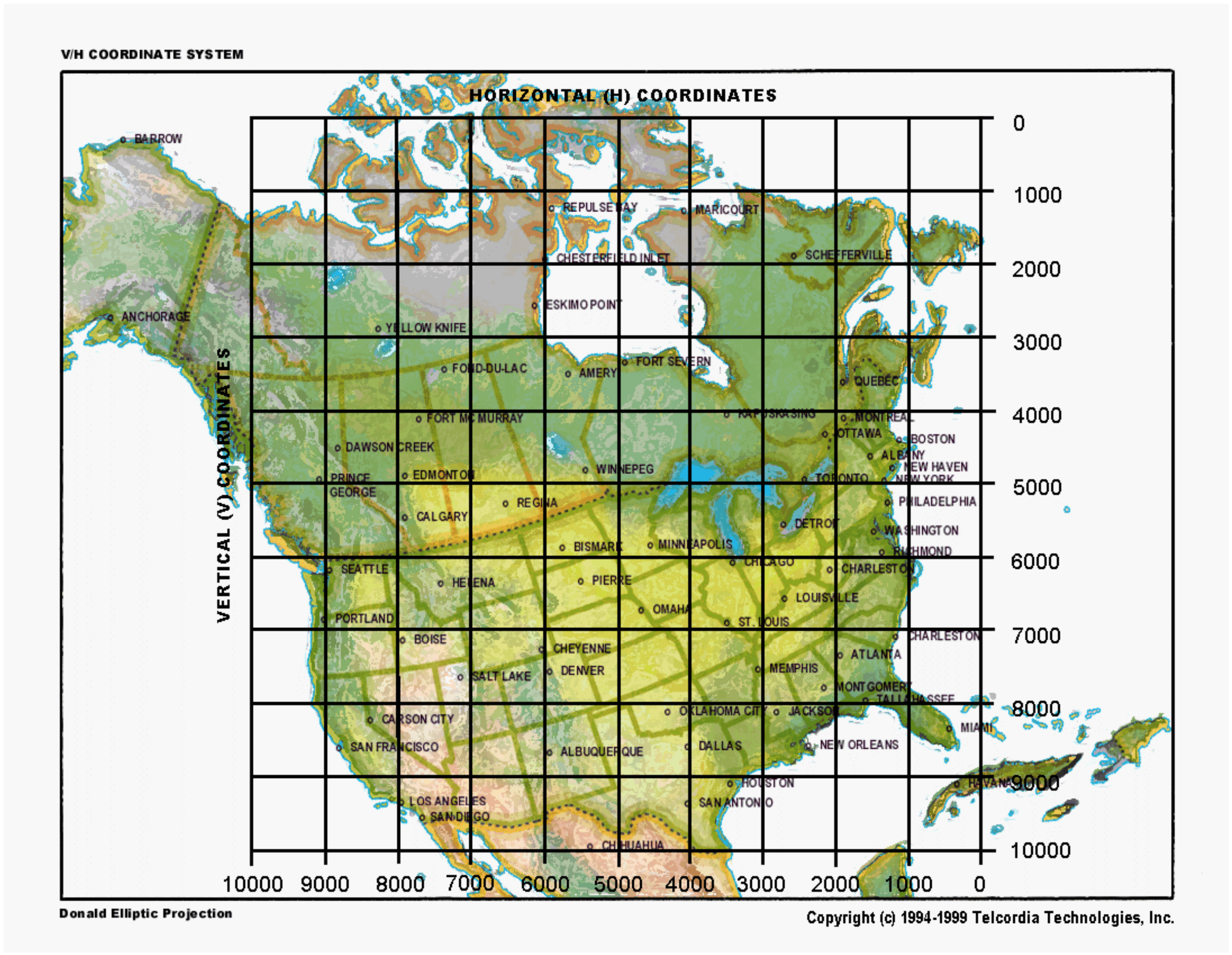

Figure 4.2 V&H coordinate system used in telecommunications industry.

Finally the data presented in Table 4.1 was loaded into the FASL Mathematica program described in the previous section. As the stopping criterion the distance $\varepsilon = 10^{-2}$ was used, which is more than enough since the original data was given in integers. The calculated optimal location is (1802, 5695). After converting it to Lat/Long, the optimal location becomes approximately 39.1833ºN/78.342ºW. By looking at the map, for example, Streets & Trips or www.topozone.com, it can be seen that this location is close to Winchester, VA, and the closest big city is Washington D.C. As a result western suburbs of Washington D.C. would be the optimal location for placing the data center. A three-dimensional plot of the total cost function for this case is given in Figure 4.3.

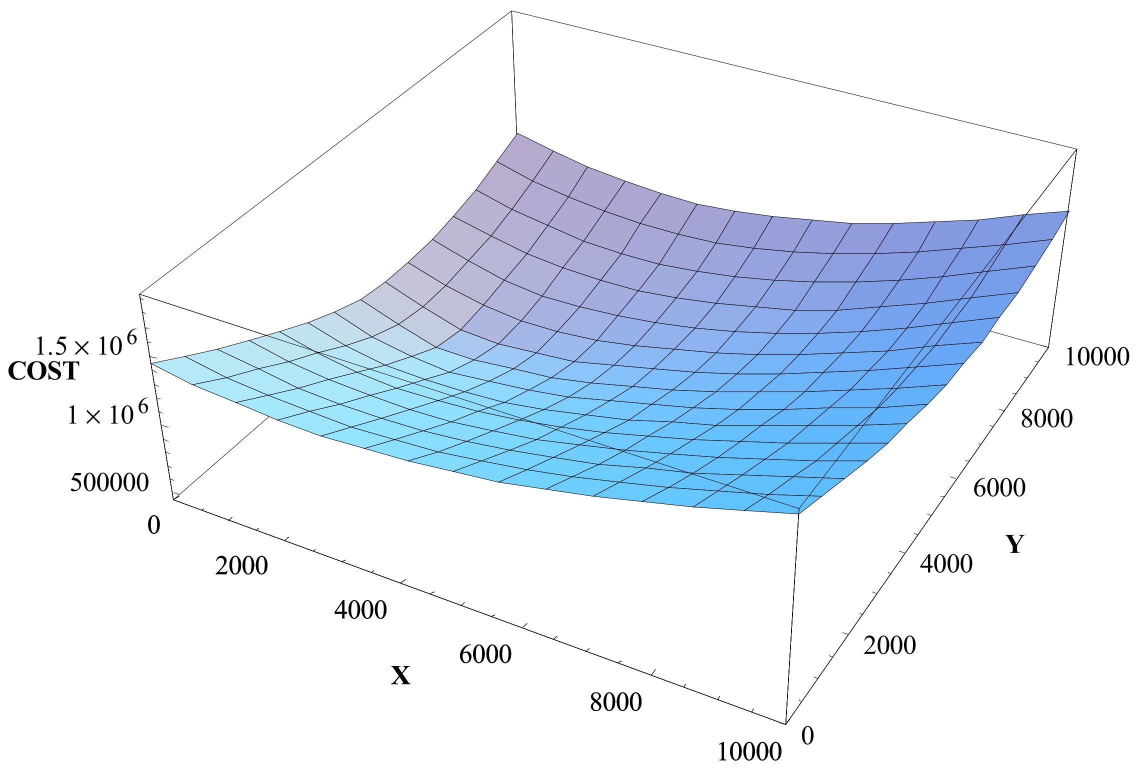

Figure 4.3 A 3D plot for the cost function of the data center location problem.

It is noteworthy that the calculated optimal location for the data center is far from obvious. One would expect that configuration described in Table 4.1 would give a value for the optimal location somewhere close to the middle of the United States. In fact, the center of gravity (3115, 6650), or 38.508ºN/87.858ºW, is close to the border of Illinois and Indiana, and the closest big city is Evansville, IL (38.090ºN/89.938ºW). The distance between Evansville and Washington D.C. found using http://www.airways.com/java/coordcalc.html is 609 miles and the total cost of placing the data center in Evansville, IL, is 6% higher than for Washington, D.C. For a big project, especially one involving fiber optic buildouts, differences like this could cost millions and millions of dollars.

In real applications, the cost is rarely linearly proportional to the distance and straight-line distances are usually not possible. In fact, the cost usually depends on the infrastructure of the provider of leased lines, like Qwest, WorldCom, etc. However, the same ideas as illustrated above can be applied to this and the following practical cases:

- Building a large fiber network by a company like Sprint or Qwest when costs of fiber buildout per foot are usually significant and star topology is used. In this case Lat/Long coordinates for the demand points could be easily obtained using some simple GPS devices and the same procedures as described above can be applied for the further steps.
- Building a large wireless network with some central location where distances and "weights" would be related to number of hops, representing the cost of a connection.
- Any other network where star topology is used.

4.3 Flow-Based Optimization of Single Switch Location

Another type of a problem where the FASL can be applied is finding the optimal location for a switch on a network with the star topology when the costs are determined by the flows between pairs of users. A network of seven users with known coordinates for each user and known flows for each logical connection between a pair of users, for example, connection AC shown on Figure 4.4, will now be considered.

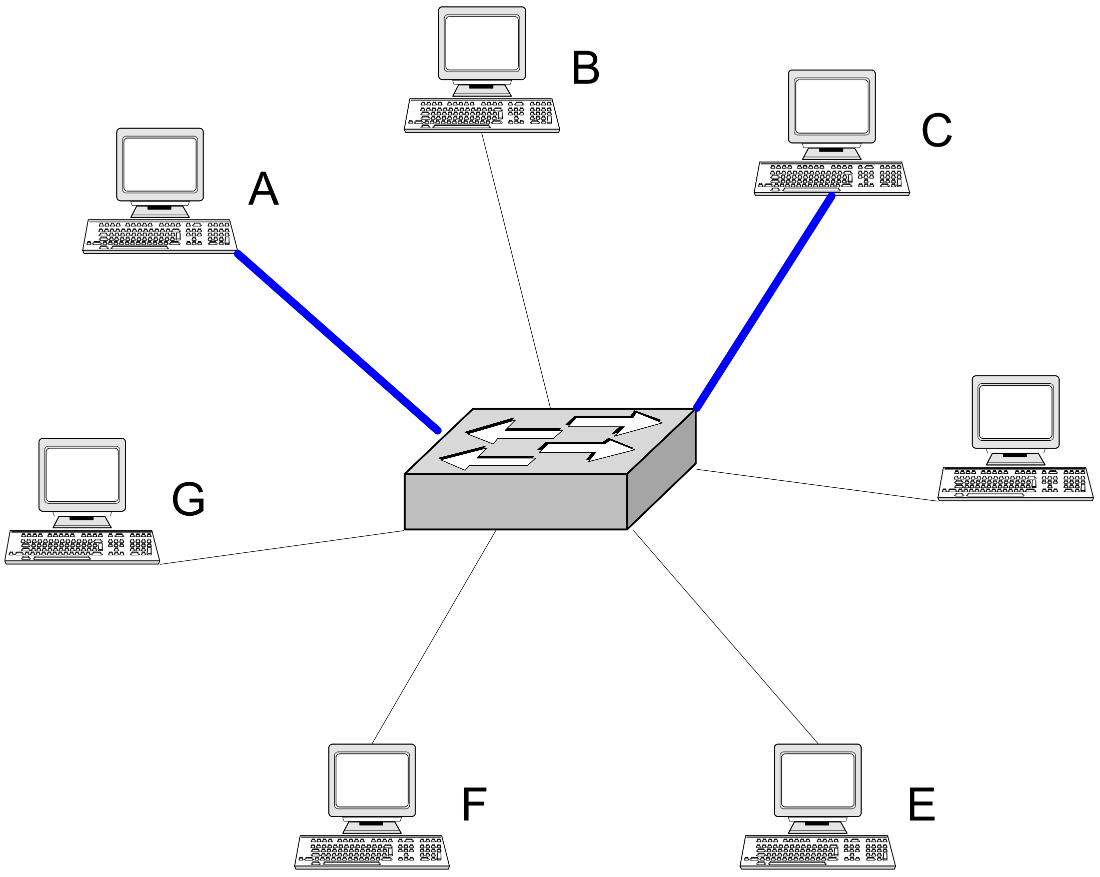

Figure 4.4 Single-switch network with star topology.

The following coordinates are given: *A*(1.02, 2.03), *B*(3.40, 3.20), *C*(5.20, 2.81), *D*(7.34, 1.50), *E*(4.50, 0.50), *F*(2.34, 0.81), *G*(0.51, 1.20). The flows are given in Table 4.2.

Table 4.2 Flows for a Single-Switch Problem with the Star Topology

	A	B	C	D	E	F	G
A	0	3	1	4	5	4	5
B	1	0	6	8	1	3	3
C	5	2	0	6	5	2	3
D	1	2	5	0	2	4	1
E	4	3	1	4	0	3	4
F	1	3	4	4	5	0	2
G	1	3	4	5	6	7	0

For simplicity, the cost function that is linearly dependent on the flow for a given link will be considered. Furthermore, for any link the cost function is given to satisfy the following equation:

$$C_{ik}\left(W_{ik}, P_i, P_k\right) = W_{ik} l_2(P_i, S) + W_{ik} l_2(S, P_k),$$

where C_{ik} is the cost function and W_{ik} is the flow between user *i* and *k*. Then the minimization problem for this case can be formulated as:

$$\min \sum_{i=1}^{n} \sum_{k=1}^{n} W_{ik} l_2(P_i, S) + W_{ik} l_2(S, P_k) \qquad (4.1)$$

This problem cannot be directly solved using the single-switch algorithm studied in this paper and some preliminary mathematical rearrangements need to be made. The same idea as in O'Kelly (1986) will be used

$$\sum_i \sum_k W_{ik} l_2(P_i, S) + W_{ik} l_2(S, P_k) = \sum_i l_2\left(P_i, S\right) \sum_k W_{ik} + \sum_k l_2\left(P_k, S\right) \sum_i W_{ik} .$$

Let $O_i = \sum_k W_{ik}$, total outflow from user *i*, and $D_k = \sum_i W_{ik}$, total inflow to user *k*, then

$\sum_i l_2(P_i, S) O_i + \sum_k l_2(P_k, S) D_k = \sum_v l_2(P_v, S) M_v$ where $B_v = O_v + D_v$.

Therefore Equation 4.1 becomes

$$\min \sum_{v=1}^{n} B_v \, l_2(P_i, S),$$

which is the standard Fermat-Weber problem (Equation 1.1) and can be solved using the FASL where B_v, which is the sum of all inflows and outflows coming to or going from a user *v*, is used as "weight".

For point *A* (letters and numbers will be used interchangeably in indexes), for example, the following is obtained

$B_A = W_{AB} + W_{AC} + W_{AD} + W_{AE} + W_{AF} + W_{AG} + W_{BA} + W_{CA} + W_{DA} + W_{EA} + W_{FA} + W_{GA} =$

$= 1 + 5 + 1 + 4 + 1 + 1 + 3 + 1 + 4 + 5 + 4 + 5 = 35$

Then for the rest of the users $B_B = 38$; $B_C = 44$; $B_D = 46$; $B_E = 43$; $B_F = 42$; $B_G = 44$. All that is left to do now is run the FASL for the initially given user locations and calculated "weights". The result obtained using the FASL gave *S*(3.43, 1.67) as the optimal position for the switch that would minimize the total cost based on the link flows.

There is another approach for solving this problem. In the classical Fermat-Weber case, "weights" are considered as some demand from a user to a switch per unit of distance. Since only flows on the links are given, Professor Verkhovsky suggested to calculate the demand from a specific user to the switch as the sum of all the flows coming to or going from the user. In the case of this hypothetical problem this demand (total flow) is equal to the sum of all link flows in the same row as that user (outflow) plus the sum of all link flows in the same column as that user (inflow), except for the flow from the user to itself, which does not go through any network link.

If the cost function is linearly dependent on the flow, both of these approaches are identical. However, when the linear dependence is not present, the first approach can no longer be used because the cost for transmission from one user to another is no longer equal to the sum of the costs from the first user to the switch and from the switch to the second user. At the same time, network flows can still be added up to yield the total flow for a user and thus the second approach is still feasible.

CHAPTER 5

FASL AS A SUBROUTINE FOR MULTI-SWITCH NETWORK PROBLEMS

5.1 Two-Switch Location Problem

In real situations, and especially in large networks, single-switch networks are used less often than more complex multi-switch networks. However, in these cases the multi-switch network can often be partitioned into clusters with a single switch each. One criterion for partitioning, for example, can be associating the users with the nearest switch. The studied FASL can then be solved for each cluster individually. Three approaches for solving the two-switch location problem where no inter-switch interaction is present will be discussed in this thesis. For a more exhaustive review and analysis of most of the existing methods, both heuristic and exact, to solve the multi-switch location problem the reader is referred to Brimberg *et al.* (2000).

A good example of the two-switch location problem with no inter-switch interaction is locating two earth stations (switches) that are shared by several small companies and are installed, configured, and maintained by some telecommunications provider. Traditional non-expensive "dishes" usually provide low throughput, cannot always guarantee a good quality, and are usually limited to simultaneous communication with only one or two satellites. While being acceptable for residential customers, these devices are often not feasible when a higher throughput, better quality, and simultaneous multi-satellite communication are needed. Earth stations are built to address these issues. However, bringing more complexity to the communication devices makes them more expensive and thus less affordable for a lot of smaller companies. To address this issue, telecommunications companies found a niche where they purchase, install, configure, and maintain earth stations that are then shared by many small companies. Henceforth, finding the optimal location of earth stations to minimize the costs is a serious concern for these telecommunications companies.

Professor Verkhovsky proposed the following general formulation for this type of problems (Verkhovsky & Polyakov, 2003):

1) Consider locations of *n* users which are specified by coordinates $P_i = (a_{i1}, a_{i2})$, $i=1,\ldots,n$. Each user is characterized by a "volume" of incoming and outgoing communication flow w_i ("weight" of *i*-th user's flow).
2) Let *m* be the number of switches, G_k be a set of all users P_i connected with the *k*-th switch C_k, and (u_k, v_k) be the coordinates of C_k.
3) $f(w_i, P_i, C_k)$ is a cost function for the transmission link connecting the *i*-th user P_i and *k*-th switch C_k.
4) The objective is to minimize the total cost of all links and all switches:

$$\min W(C) = \sum_{k=1}^{m}\left[\sum\nolimits_{i\in G_k} f(w_i, P_i, C_k) + q_k \sum\nolimits_{i\in G_k} w_i\right] \qquad (5.1)$$

where $q_k \sum_{i\in G_k} w_i$ is the cost of the *k*-th switch as a function of all outgoing and incoming flows.

For simplicity, the two-switch problem when the switching costs can be neglected and the cost function satisfies the Fermat-Weber problem (Equation 1.1) will be considered. In this case Equation 5.1 becomes

$$\min W(C) = \sum\nolimits_{i\in G_1} w_i\, l_2(P_i, C_1) + \sum\nolimits_{k\in G_2} w_k\, l_2(P_k, C_2) \qquad (5.2)$$

It is noteworthy that this problem does not only involve solving the location problem to find the optimal switch positions, but it also needs to determine the optimal partition (G_1, G_2). For this reason, two stages are usually present in each iteration of this kind of problems: location and reassignment.

The first approach, originally discovered in Cooper (1963) and further studied in Levin & Ben-Israel (2001), is based on the idea that assigning users to the nearest switch results in total cost reduction. Initially a random partition is inputted. Then for every iteration optimal switch locations for each of the clusters are computed using the FASL. After that users are reassigned to the nearest switch if the current distance between a specific user and the switch is more for the current cluster than for the other one. This idea is applied over and over until no more reassignments are possible. The actual Cooper's algorithm, similar to the one presented in Levin & Ben-Israel (2001), is listed below:

```
Given a partition $G^k = \{G_1, G_2\}$ of the set of points $\{P_1, ..., P_n\}$
**While** no more reassignments are possible **do**:
        1. Solve $\min \sum_{i \in G_k} w_i l_2(P_i, C_k)$ to find $C_k$, optimal switch location, where $k = 1,2$.
        2. For $i \in G_1$, **if** $l_2(P_i, C_1) > l_2(P_i, C_2)$ **then** reassign $P_i$ to $G_2$.
           For $i \in G_2$, **if** $l_2(P_i, C_2) > l_2(P_i, C_1)$ **then** reassign $P_i$ to $G_1$.
```

The same idea has been successfully applied to multi-switch location problems (Cooper, 1963; Levin & Ben-Israel, 2001). As was demonstrated in those papers, this heuristic algorithm does not always yield the optimal solution.

The second considered algorithm (Ostresh, 1973; Drezner, 1984), which will be referred to as the Drezner algorithm, is based on the fact that the optimal partition consists of two non-overlapping clusters and a theorem stating that one of the straight lines going through all possible pairs of users must provide the optimal partition. The proof of this theorem relies on proportionality between the cost function and the distance, thus limiting this solution to cases when this proportionality is present. Therefore, for example, in a problem where the economy of scale would be considered, this algorithm might no longer work.

The main idea of this algorithm is to rotate a line going through each user point, used as a pivot, in such a way that after each rotation only one of the remaining users is reassigned from one cluster to another. For each user point selected as a pivot, this can be easily accomplished by calculating the angles that the n-1 straight lines going through the pivot and the remaining points make with the abscissa axis. Then the angles are sorted and for each rotation two single-switch problems are solved. It is obvious that $n(n-1)$ different partitions are going to be considered and for each case the single-switch problem needs to be solved. Sorting only by angles creates a potential issue when two user points lie on the same straight line. Special resolution for this situation would have to be used (Love *et. al.*, 1988).

Drezner (1984) also suggested to use an effective lower bound for the minimum value of the objective function (total cost) for a single-switch algorithm, such as FASL. If the value of the minimal bound is higher than the currently calculated minimum value of the total cost function, then the FASL does not need to be run. Both summands for the lower bound can easily be obtained because they represent one-dimensional single-facility minisum location problems which are known to be solved by finding the median point (Love *et al.*, 1988). Expression for the this lower bound is given below:

$$W_D(S_0) = \min_x \sum_{i=1}^{n} w_i^x \left|x - a_{i1}\right| + \min_y \sum_{i=1}^{n} w_i^y \left|y - a_{i2}\right|,$$

$$w_i^x = w_i \left|x_0 - a_{i1}\right| / l_2\left(P_i, S_0\right),$$

$$w_i^y = w_i \left|y_0 - a_{i2}\right| / l_2\left(P_i, S_0\right),$$

where $S_0 = \left(x_0, y_0\right)$ is the center of gravity for this single-switch problem.

The interesting feature of this algorithm is that it provides the exact solution, which cannot be guaranteed for most of the existing algorithms to solve the multi-switch location problem. At the same time it comes at its price resulting in the complexity of $\mathrm{O}\left(n^2\right)$ single-switch location problems that need to be solved. However, this is much less compared to the exhaustive approach where all possible partitions, $\mathrm{O}\left(2^{n-1}\right)$, would have to be considered. Unfortunately, this approach cannot be easily extended to more-than-two switch location problems since partitions are divided into non-overlapping Voronoi clusters (diagrams), which can no longer be separated by one simple straight line, like in the two-switch location case. Rosing (1992) demonstrated that problems with up to six switches can be solved using this approach.

The third approach was proposed by Professor Verkhovsky (Verkhovsky & Polyakov, 2003). It is in some sense similar to the previous (Drezner) algorithm in that it rotates a straight line, although it has two major differences which will be discussed later in the thesis. At first a generalized clustering algorithm on which the Verkhovsky algorithm is based is formulated below:

1. Find a center of rotation *R*.

2. Convert all points/users to the polar system of coordinates (α, r) where *R* is the center of the coordinate system.

Comment: All points are also divided into the ones above and below the horizontal line going through *R*.

3. If $\alpha>180°$ then $\alpha := \alpha - 180°$; $(\alpha,r):=(\alpha, r, below)$, else $(\alpha,r):=(\alpha, r, above)$.

4. Lexicographically sort all points: Let $U_1 := (\alpha_1, r_1, flag_1)$; $U_2 := (\alpha_2, r_2, flag_2)$. If $\alpha_1 < \alpha_2$, then $U_1 < U_2$; if $\alpha_1 = \alpha_2$ and $r_1<r_2$, then $U_1 < U_2$; if $\alpha_1 = \alpha_2$ and $r_1 = r_2$ and $flag_1=above$, then $U_1 < U_2$.

5. Rotate straight line *L* around *R* for all *x* between 0° and 180°. Let $i = 1,\dots, n$. If ($\alpha_i <=$ x and $flag_i = above$) or ($\alpha_i > x$ and $flag_i = below$), then U_k belongs to cluster G_2; else U_k belongs to cluster G_1.

The algorithm is very similar to the Drezner algorithm, though in this case the rotation pivot is known and thus only *n* rotations need to be made. The question is what should the pivot point be to result in the optimal or close-to-optimal solution? Professor Verkhovsky suggested to use the optimal switch location for the single-switch problem solved for all *n* users. Since this suggestion cannot be mathematically verified, the algorithm cannot be considered exact and thus its accuracy compared to Cooper's algorithm will be discussed. As some calculations showed situations with “fugitives”, for which the cost function value for the current cluster is higher than if the user would be placed into the other cluster, Cooper’s algorithm is suggested as the last step to send the “fugitives” where they belong. Complete algorithm for the two-switch Weber location problem is presented below:

Step 1: Solve the single-switch minisum problem for all *n* users

$$\min_{(u_0,v_0)} \sum_{i=1}^{n} w_i l_2\left(P_i, C_0\right)$$

to find C_0, the pivot for rotation.

Step 2: Do steps 2 – 4 of the clustering algorithm with C_0 used instead of *R*.

Step 3: For all points *i* with $flag_{\mathrm{i}}$ = *above*, add their Cartesian coordinates to cluster G_1.

The rest of the points are added to cluster G_2.

Step4: Let h_{min} be an arbitrary large number, $GO_1 := G_1$, $GO_2 := G_2$.

For all users $i = 1,\ldots,n$ do the following (rotate the line *L*)

{

Step 4a: Let $x := \alpha_i$. If $flag_{\mathrm{i}}$ = *above*, then reassign the point *i* from cluster G_1 to G_2.

Else reassign the point *i* from cluster G_2 to G_1.

Step 4b: Compute

$$g_k\left(G_k(x)\right) := \min_{(u_k,v_k)} \sum_{i \in G_k(x)} w_i\, l_2\left(P_i, C_k\right),\ k = 1,2.$$

Step 4c: Compute

$$h(x) := g_1(G_1(x)) + g_2(G_2(x)). \qquad (5.3)$$

```
        Step 4d: If $h(x) < h_{min}$ then $h_{min} := h(x)$, $GO_1 := G_1$, $GO_2 := G_2$, $CO_1 := C_1$, $CO_2 :=$
                 $C_2$}
Step 5: {"Fugitive" handling after the main algorithm is complete}
        Let $count := 1$;
        While $count > 0$ {reassignments were done in the previous step or the this
                            part is called the first time}
        {       $count := 0$;
        Step 5a: If $l_2(P_i, CO_1) > l_2(P_i, CO_2)$ for $i \in GO_1$, then reassign $i \in GO_2$; $count$++.
                 If $l_2(P_i, CO_2) > l_2(P_i, CO_1)$ for $i \in GO_2$, then reassign $i \in GO_1$; $count$++.
        Step 5b: Using the FASL find the optimal locations of $CO_1$ and $CO_2$ for the new
                 values of $GO_1$ and $GO_2$.}
$C_1$, $GO_1$, $C_2$, $GO_2$ is the final solution for the two-switch location problem.
```

The lower bound of the objective value introduced for the Drezner algorithm can also be successfully employed in this case.

Before the complexity of the three algorithms above is discussed in detail, it is important to mention the main differences between the Drezner and Verkhovsky algorithms. First, it is obvious that the Verkhovsky algorithm is much faster since approximately *n* different partitions are considered compared to *n*(*n*-1) for Drezner's. Second, the Drezner algorithm was designed with a convex cost function proportional to the weight in mind. At the same time, the idea used in the Verkhovsky algorithm can be applied to any general cost function and some results demonstrating that are presented in

(Verkhovsky & Polyakov, 2003). Therefore the Verkhovsky algorithm should be appropriate for many practical problems for which the Drezner algorithm could not even be considered due to the constraints in the algorithm formulation.

5.2 Computer Experiments and Optimizations

More than one hundred experiments with coordinates for the users and the associated weights generated using uniform random distribution on the interval (0,1) were run. The three algorithms discussed above, Cooper's, Drezner's, and Verkhovsky's, were compared. Experiments were run for values of *n* ranging from 15 to 1000. The results obtained here are similar to the results discussed in (Veroy, 1989; Verkhovsky & Polyakov, 2003).

As could be expected, both the Cooper and Verkhovsky algorithms did not always yield the optimal solution, which obviously was always provided by the Drezner algorithm. However, in most of the cases when non-optimality occurred the deviations of the minimum total cost calculated using the Verkhovsky algorithm from the exact solution was of order 0.1%, which was often remedied by the reassignment-location steps run after the main part of the algorithm. At the same time, the Cooper algorithm sometimes resulted in the minimal values of the objective function that were as far as 7% higher than the exact solution. Therefore, the same kind of accuracy as provided by the Verkhovsky algorithm can only be achieved after several different random partitions are used to initialize the Cooper algorithm.

Now the complexity of each algorithm in terms of the number of the single-switch location problems will be analyzed and discussed. It is obvious that the highest complexity is attributed to the Drezner algorithm since $O\left(n^2\right)$ location problems need to be solved. Though providing exact solutions, this algorithm becomes non-feasible for a powerful modern workstation when the number of users exceeds 50-100.

As seen from the algorithm itself, the complexity of the Verkhovsky algorithm is O(n) plus the additional iterations of the Cooper algorithm executed after the main part. In approximately 40% of the computations, this number of additional steps was zero (no single-switch location problems needed to be solved). For other situations, the maximum number that was usually seen was not higher than three. Therefore the complexity of this algorithm is O(n). It is noteworthy that the number of these reassignment-location steps slowly increased for higher values of n. For example, when $n < 25$, no reassignments were usually necessary. However, when n was about 250, two or three steps were sometimes needed.

The best performance was demonstrated by the Cooper algorithm. Usually up to 10 iterations was enough to solve problems with up to 250 users. At the same time, to compensate for the deviations this algorithm would need to be restarted several times.

For the Verkhovsky algorithm, Professor Verkhovsky suggested a special Fibonacci optimal search developed in (Veroy, 1989) that can be applied to any discrete periodic bimodal function. The Fibonacci search was mathematically proven to be an optimal algorithm for finding a minimum (or maximum) of any discrete periodic bimodal function, which means that no algorithm to provide a smaller asymptotical complexity, that is to be faster, can be developed for such a function and only slower or same-complexity algorithms are possible. The function that is suspected to demonstrate the bimodal behavior is $h(x)$, total cost for a specific partition determined by the current angle x of rotation, formulated in Equation 5.3. Therefore, modality of this function and applicability of the Fibonacci search were studied.

Before the results are discussed, it seems reasonable to list the Fibonacci search algorithm developed in (Veroy, 1989). Two situations are possible: (a) if the number of users is equal to some Fibonacci number F_u; (b) if it is not equal to any Fibonacci number. For simplicity of notation $x(i) = x_i$.

(a) Optimal algorithm if $n = F_u$:

1. **If** $F_u = 1$, **then** $h_r = h_1$ and $x_r = x_1$; **stop**.
2. Select an integer L_0 arbitrarily, $R_0 = L_0 + F_{u-1.}$ (in most cases $L_0 = 1$ is easier to work with).
3. **Compute** $e(L_0)$ and $e(R_0)$, where $e(t) = h(x_t)$.
4. (Selecting an initial detecting state). **If** $e(L_0) \geq e(R_0)$, **then** $A_1 = L_0, Z_1 = L_0 + F_u$, $R_1 = R_0, L_1 = A_1 + F_{u-2}$, else $Z_1 = R_0, A_1 = R_0 - F_u, L_1 = L_0, R_1 = Z_1 - F_{u-2}$.
5. **If** $e(L_k) \geq e(R_k)$, then $A_{k+1} = L_k, temp = e(R_k), L_{k+1} = R_k, R_{k+1} = 2L_k - A_k$, **compute** $e(R_{k+1}), Z_{k+1} = Z_k, I_{k+1} = Z_k - L_k, e(L_{k+1}) = temp$; **else** $Z_{k+1} = R_k$, $temp = e(L_k), R_{k+1} = L_k, L_{k+1} = 2R_k - Z_k$, **compute** $e(L_{k+1}), A_{k+1} = A_k$, $I_{k+1} = R_k - A_k$, $e(R_{k+1}) = temp.$
6. **While** $I_k > 1$, **repeat** step 5.
7. $h_r = temp$, **stop**.

(b) Optimal algorithm if $F_{u-1} < n < F_u$.

Let T be the period of the periodic bimodal function $h(x)$.

The algorithm is the same except for the following two modifications:

In step 2 set $L_0 = 1$.

In step 4 add the following:

If $e(L_0) \geq e(R_0)$, **then** select arbitrary $F_u - n$ fictitious points $x(i)$ for $n+1 \leq i \leq F_u$ where $x(n) < x(n+1) < x(F_u) < x(L_0 + T)$ and $e(n+1) = ... = e(F_u) = e(L_0)$;

else select arbitrary $F_u - n$ fictitious points $x(i)$ for $R_0 + 1 - F_u \leq i \leq R_0 - n$ where $x(R_0) - T < x(R_0 + 1 - F_u) < ... < x(R_0 - n) < x(R_0 + 1) - T$ and $e(R_0 + 1 - F_u) = ... = e(R_0 - n) = e(R_0)$;

Comment: In other words, the remaining points needed to reach F_u are set to the same value as L_0 if $e(L_0) \geq e(R_0)$ and R_0 otherwise.

To check if the Fibonacci optimal search algorithm can be applied in this case, the function $h(x)$ was extended to also include $h(x-T) = h(x)$ for the Fibonacci algorithm to work properly. Complexity analysis in (Veroy, 1989) showed that the search is $\mathrm{O}(\log n)$.

In the same experiments as used above, bimodality of $h(x)$ and applicability of the Fibonacci search were checked. For values of $n < 50$, function $h(x)$ rarely demonstrated bimodal behavior and thus the Fibonacci search was sometimes stopping at a value of $h(x)$ which was up to 7% higher than the optimal. For values of $n > 50$, function $h(x)$

usually had several local minimums, but the overall behavior was bimodal and thus the Fibonacci search always returned a value of $h(x)$ that was not more than 0.5% higher than the optimal. In addition to that, the Cooper algorithm that is run after the main part of the Verkhovsky algorithm usually compensated for these small deviations by running additional one or two reassignment-location steps. As the result, the accelerated Verkhovsky algorithm demonstrated a better accuracy compared to the Cooper algorithm although only 1 – 4 times as many reassignment-location iterations were needed. To achieve the same accuracy, the Cooper algorithm would have to be run for several random initial partitions. Thus for $n > 50$ the accelerated Verkhovsky algorithm is the preferred method to solve the two-switch location problem. The obtained results seem to be in a good agreement with Veroy (1989) and Verkhovsky & Polyakov (2003).

CHAPTER 6

CONCLUSIONS

The Feedback Algorithm for Switch Location in which the current Weiszfeld iterate is multiplied by the feedback factor shows the better performance than the existing accelerating procedures for the single-facility minisum problem. The complete algorithm including this acceleration technique and handling of singularities is presented. Computer-related applications of the single-facility minisum problem are thoroughly discussed. Aspects of working with coordinates for geographical locations are paid special attention to.

Due to its simplicity, the FASL can be used for a wide range of practical computer-related location problems involving minimization of links costs. Finding an optimal location for a data switch on a network is a good example of a case when the algorithm can be successfully applied. In addition, the FASL can be used as a subroutine for the multi-switch location problem. A brief discussion of how this can be accomplished is presented.

APPENDIX A

FEEDBACK ALGORITHM FOR SWITCH LOCATION

The complete version of the Feedback Algorithm for Switch Location in pseudocode is presented.

Notations and definitions

{All algebraic notations (but not numbers, not parenthesis, and not other punctuation marks), must always be in *italics*};

n – number of users, {integer};

V – number of iterations;

Points (users): $P(k) = [a1(k),a2(k)]$, k=1,..,n;

Center (switch): $S = (x,y)$;

“Weights” of links: $w(k)$, k=1,..,n;

Total weighted distance of all links (Criterion): $W(x,y)$;

The **goal** is to find with accuracy $z>0$ such a location of the center $S=(x,y)$ that minimizes $W(x,y)$.

Algorithm for the FASL procedure

```
1. Initialize $x1 := [a1(1)w(1)+..+a1(n)w(n)]/[w(1)+..+w(n)]$;
2. Initialize $x2 := [a2(1)w(1)+..+a2(n)w(n)]/[w(1)+..+w(n)]$;
3. $x := x1$; $y := x2$; {use temporary variables}
4. $V := 0$.
```

```
{Iteration starts}
5. $M$:=0; $N$:=0; $D$:=0;
6. $V$:=$V$+1;
For $k$ from 1 to $n$ do {
7. $d(k)$:=sq.root{[$a1(k)$-$x$]^2+[$a2(k)$-$y$]^2};
8. $r(k)$:=$w(k)$/$d(k)$;
    If division by zero exception is not triggered, then proceed further.
    Else
    {
            $x1 = x1 + \delta_1$; $y1 = y1 + \delta_2$;
            Go to Step 3. {Rerun the procedure with the new values for $x$ and $y$ }
    }
9. $M$:= $M$+$a1(k)r(k)$;
10. $N$:= $N$+$a2(k)r(k)$;
11. $D$:= $D$+$r(k)$;
} {End of for};
12. $Q$:=$M/D$;
13. $R$:=$N/D$;
14. If $V > 1$ then $x0 = u$; $y0 = v$;
15. $u$:=$x$; $v$:=$y$;
16. Iterate $x$:=$Q$^2/$x$;  $y$:=$R$^2/$y$;
17. If sq.root{[$a1(k)$-$x$]^2+[$a2(k)$-$y$]^2}<z then stop;
```

```
18. If V > 2 Then
        a. If $(u - x0)(x - u) < 0$ then $x = (x + u)/2$;
        b. If $(v - y0)(y - v) < 0$ then $y = (y + v)/2$;
{Iteration ends}
```

APPENDIX B

FASL IMPLEMENTATION IN MATHEMATICA 4.1

A complete listing of the source code in Mathematica 4.1 for the algorithm described in Appendix A is presented.

```
(* declare the file to write output to *)
SetDirectory["D:\CS Research\Center Location Algorithm\Mathematica files"];
strm1 = OpenWrite["resultsabbrev.txt"];

 (* evaluates the initial values for x1 and x2 *)
FASL[x1_, x2_] :=
   (
     Module[{x, y, iV, iM, iN, iD, iQ, iR, k, r, d, ak, x0, y0, u, v}, (*
          declare local variables *)

        x = x1;
        y = x2;

        (* initialization *)
        iV = 0;

        While[True,
          iV++;
          iM = 0;
          iN = 0;
          iD = 0;
```

```
For[k = 1,
 k < n + 1 ,
 k++,
 d = N[Sqrt[(a[[k, 1]] - x)^2 + (a[[k, 2]] - y)^2]];

 Check[
  r = N[w[[k]]/d],

  Print["Explanation: The current iterate coincides with one of \
the demand points.\nRestarting with different initial conditions."];

  WriteString[strm1,
   "The current iterate coincides with one of the demand \
points.\nRestarting with different initial conditions."];
  iDone = 0;
  Return[],
  Power::infy
  ];

 iM = N[iM + a[[k, 1]] r];
 iN = N[iN + a[[k, 2]] r];
 iD = N[iD + r];
 ];

iQ = N[iM/iD];
iR = N[iN/iD];

If[(iV > 1), x0 = u; y0 = v;];

u = x; v = y;
x = N[ iQ^2/x]; y = N[ iR^2/y];
```

```
      If[N[Sqrt[(x - u)^2 + (y - v)^2]] < z, Break[];];

      If[iV > 2,
        If[(u - x0) (x - u) < 0, x = N[(x + u)/2]];
        If[(v - y0) (y - v) < 0, y = N[(y + v)/2]];
        ];
      ];

    Print["\nResults obtained using the Feedback Algorithm:\n n= ",
      n // N, "; z=", z // N, "; x1 =", N[x, 20], "; x2 = ", N[y, 20],
      "; iV = ", iV // N, "\n"];

    WriteString[strm1,
      "\nResults obtained using the Feedback Algorithm:\n n= ", n // N,
      "; z=", z // N, "; x1 =", N[x, 20], "; x2 = ", N[y, 20],
      "; iV = ", iV // N, "\n"];

    ]; (* end of module *)

  );

(*-----INITIALIZATION STARTS---------------*)

(* initialization of the variables *)
a1min = 0;
a1max = 100;
a2min = 0;
a2max = 100;
wmin = 1;
wmax = 100;
```

```
n = 20;
z = 1.00 10^(-6);

(* random generation for test purposes *)
(*
fa[x_] := {Random[Integer, {a1min, a1max}], Random[Integer, {a2min, a2max}]};
fw[x_] := Random[Integer, {wmin, wmax}];
a = Array[fa, n];
w = Array[fw, n];
*)

Print["Initial Conditions:\n a = ", a // N, "\n w= ", w // N, "\n"];
WriteString[strm1, "Initial Conditions:\n a = ", a // N, "\n w= ", w // N,
   "\n"];

wSum = N[Sum[w[[i]], {i, 1, n}]];
 x1 = N[Sum[a[[i, 1]] w[[i]], {i, 1, n}]/wSum];
 x2 = N[Sum[a[[i, 2]] w[[i]], {i, 1, n}]/wSum];

(*-----------INITIALIZATION ENDS--------------------*)

iDone = 0;
While[iDone == 0,
   iDone++;
   FASL[x1, x2];
   If[iDone == 0,
     x1 = x1 + 0.5;
     x2 = x2 + 0.5;
     ];
   ];
Close[strm1];
```

REFERENCES

Brimberg, J. (1995). The Fermat-Weber location problem revisited. Math. Programming., 71, 71-76.

Brimberg, J., & Chen, R. (1998). A note on convergence in the single facility minisum location problem. Comput. Math. Appl., 35, 25-31.

Brimberg, J., & Love, R.F. (1992). Local convergence in a generalized Fermat-Weber problem. Ann. Oper. Res., 40, 33-66.

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.

Brimberg, J., Chen, R., & Chen, D. (1998). Accelerating convergence in the Fermat-Weber location problem. Oper. Res. Letters, 22, 151-157.

Brimberg, J., Hansen, P., Mladenovic, M., & Taillard, E. (2000). Improvements and comparison of heuristics for solving the uncapacitated multisource Weber problem. Operations Research, 48, 444-460.

Burden, R. L., Faires, J. D., & Reynolds, A. C. (1981). Numerical Analysis. 2nd Ed., Boston: Prindle, Weber and Schmidt.

Chandrasekaran, R., & Tamir, A. (1989) Open questions concerning Weiszfeld's algorithm for the Fermat-Weber location problem. Math. Programming., 44, 293-295.

Cohen, A. I. (1981). Stepsize analysis for descent methods. J. Optim. Theory Appl., 33, 187-205.

Cooper, L. (1963). Location-allocation problems. Operations Research, 11, 37-52.

Drezner, Z. (1984). The planar two-center and two-median problems, Transport. Sci., 18, 351-361.

Drezner, Z. (1992). A note on the Weber location problem. Ann. Oper. Res., 40, 153-161.

Drezner, Z. (1995). A note on accelerating the Weiszfeld procedure. Location Science, 3, 275-279.

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.

Handler, G. Y., and Mirchandani, P. B. (1979). Location on networks: Theory and algorithms. Cambridge, MA: MIT Press.

Katz, I.N. (1974). Local convergence in Fermat's problem. Math. Programming, 6, 89-104.

Kuhn, H.W., and Kuenne, R. (1962). An efficient algorithm for the numerical solution of the generalized Weber problem in spatial economics. Journal of Regional Science, 4, 21-34.

Levin, Y., & Ben-Israel, A. (2001). A heuristic method for multifacility location problems. RUTCOR Research Report, RRR 36-2001. Rutgers University.

Levin, Y., and Ben-Israel, A. (2002a). Directional Newton methods in *n* variables. Math. of Computation, 71, 237-250.

Levin, Y., and Ben-Israel, A. (2002b). The Newton Bracketing method for convex optimization, Computational Optimization and Applications, 21, 213-229.

Love, R. F., Morris, J. G., & Wesolowsky, G. O. (1988). Facilities location, models and methods. North-Holland. New York.

Miehle, W. (1958). Link-length minimization in networks. Operations Research, 6, 232-243.

O'Kelly, M. E. (1986). The Location of Interacting Hub Facilities, Trans. Sci., 20, 92-106.

Ostresh, L. M., Jr. (1973). TWAIN – exact solutions to the two-source location-allocation problem. In G. Rushton, M. F. Goodchild, & L. M. Jr. (Eds), Computer programs for location-allocation problems (pp. 15 – 28), Monograph No. 6, Department of Geography, University of Iowa, Iowa City, IA.

Ostresh, L. M. (1978). On the convergence of a class of iterative methods for solving the Weber problem. Operations Research, 26, 597-609.

Rosing, K. E. (1992). An optimal method for solving the (generalized) multi-Weber problem. European J. Operations Research, 58, 414 – 426.

Uster, H., & Love, R. F. (2002). A generalization of the rectangular bounding method for continuous location models. Comput. & Mathem. with Applic., 44, 181-191.

Uster, H., & Love, R.F. (2000). The convergence of the Weiszfeld Algorithm. Comput. Mathem. App., 40, 443-451.

Verkhovsky, B. (1976a). Algorithm for controlled feedback for system of equations with stochastic matrix. IBM Technical Disclosure Bulletin, Vol.18, No.10.

Verkhovsky, B. (1976b). Algorithm for system of equations with stochastic matrix. IBM Technical Disclosure Bulletin, Vol.18, No.10.

Verkhovsky, B. (1976c). Algorithm with nonlinear acceleration for a system of linear equations. Research Report, No.76-WR-1, Princeton University.

Verkhovsky, B. (1976d). Feedback algorithm for system of equations. IBM Technical Disclosure Bulletin, 18, No.10.

Verkhovsky, B. (1977). Smoothing systems design and parametric markovian programming, Markov Decision Theory, Ed. H.C. Tijms and J. Wessels, Mathematische Centrum, Amsterdam, pp.105-117.

Verkhovsky, B. (1978). Delinearization algorithms for elliptic partial differential equations. Research Report, 78-WR-3, Princeton University, 1978, pp. 1-12.

Verkhovsky, B., & Polyakov Yu. (2002b). Feedback algorithm for the single-facility minisum problem, Annals of the European Academy of Sciences.

Verkhovsky, B., & Polyakov, Yu. (2002a). Accelerated algorithm for the single-facility minisum problem. Research Report CS 02-05, New Jersey Institute of Technology.

Verkhovsky, B. S & Polyakov, Yu. S. (2003, Jul. – Aug.). Highly Efficient Algorithm for Two-Switch Location Problem. 15th Int'l Conf. on Systems Research, Informatics and Cybernetics. Baden-Baden, Germany (accepted).

Veroy (now Verkhovsky), B. (November 1985). Projection algorithm for a stochastic dynamic programming problem, Proc. NEACP Technology & Science Conference, Newton, Massachusets.

Veroy, B. (1989). Optimal search algorithm for extrema of a discrete periodic bimodal function. J. Complexity, 5, 238-250.

Vertical and horizontal coordinates (2003). [Article posted on Web site Telcordia Technologies]. Retrieved in February of 2003 from the World Wide Web: http://www.trainfo.com/products_services/tra/vhpage.html#V&H%20 Coordinates:%20The%20Mystery%20Unveiled.

Weber, A. (1909). Ueber den standort der industrien. Tubingen (English Translation: by Friedrich, C. J. (1929). Theory of the location of industries. Chicago: University of Chicago Press).

Weiszfeld, E. (1937). Sur la point pour lequel la somme de distances de n points donnés est minimum. Tohoku Math. J., 43, 355-386.

Zacharias, M. (1931). Elementargeometrie und Elementare Nicht-Euklidische Geometrie in Synthetischer Behandlung. Encyklopadie der Mathematischen Wissenschaften. III AB 9. 859-1172.