Computer Science and Systems Analysis

Computer Science and Systems Analysis

Technical Reports

Miami University *Year* 1992

Location-Allocation Analysis of Retailing Using Microsoft Windows

Jessica Lee
Miami University, commons-admin@lib.muohio.edu

MIAMI UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE & SYSTEMS ANALYSIS

TECHNICAL REPORT: MU-SEAS-CSA-1992-004

Location-Allocation Analysis of Retailing Using Microsoft Windows

Jessica P. Lee

School of Engineering & Applied Science | Oxford, Ohio 45056 | 513-529-5928

Location-Allocation Analysis of Retailing Using Microsoft Windows

by

Jessica P. Lee
Systems Analysis Department
Miami University
Oxford, Ohio 45056

Working Paper #92-004 April 1992

SYSTEMS ANALYSIS DEPARTMENT

MASTER'S PROJECT FINAL REPORT

Presented in Partial Fulfillment of the Requirements
for the Degree of
Master of Systems Analysis
in the
Graduate School of Miami University

TITLE: LOCATION-ALLOCATION ANALYSIS OF RETAILING USING MICROSOFT WINDOWS

PRESENTED BY: JESSICA P. LEE

DATE: MAY 1, 1992

COMMITTEE MEMBERS:

Donald Byrkett, Advisor

Alton Sanders

Cyrus Young

Table of Contents

Project Summary

The goal of my project was to use my undergraduate knowledge of geography to develop a Windows based program for the location of retail activities. Specifically, I wanted to:

1. Develop a program that will provide a user-friendly interface for location-allocation analysis,
2. Provide an educational tool for courses in location studies,
3. Explore the programming features of Microsoft Windows.

The program I developed is called Loc.

This paper describes the general characteristics of location-allocation models, and the application of these models to retailing activities. It also describes how Microsoft Windows was used to provide a user friendly environment, and the use of the program Loc.

Acknowledgements

I would like to thank Dr. Donald Byrkett for providing guidance and encouragement to me when I was contending with Windows programming. His enormous time spent in revising this paper is also sincerely appreciated. I would also like to express my gratitude to Dr. Douglas Troy and Dr. Cyrus Young for their valuable comments on the program and this paper.

1. INTRODUCTION

Determining locations for new stores is crucial to the growth of revenues for a retail establishment. New stores that are optimally located may expand the market area to the largest possible extent and thus increase the potential for sales and profits to the maximum. Making location decisions is often a complex task. In a site-selection process, two sub-processes are actually involved. The first one is to choose the "best" locations for new stores, and the second one is to allocate customers to the new and existing stores according to the expected pattern of customer travel so as to determine the trade areas of the chain stores and to forecast revenues. These two sub-processes are interwoven. Optimal locations depend on consumer allocations and consumer allocations in turn depend on optimal locations. Location-allocation analysis addresses these two aspects of site-selection problems. When applied to retailing, it represents an attempt to find the optimal locations for a number of new stores so that the largest market area can be attained.

In conducting location-allocation analysis, graphics often plays an important role. For instance, before selecting a set of feasible (possible) solutions as the starting point, it is helpful to first study a map of the distribution of population or some other demand variables of the area under study. After results are obtained by solving the location-allocation problems, it is particularly important to plot the optimal locations on the map of distribution of demand in order to analyze the results effectively. Past research in this area has almost all used some form of graphics to present results and analysis.

The chief objective of this research project is to provide a user-friendly graphical interface for location-allocation analysis of retailing. An application program named Loc is developed with Microsoft Windows. The application program allows a user to interactively supply the parameters for a location-allocation algorithm. It also presents maps and reports to assist users to analyze the results. The program serves very well as an instructional tool for the concepts of location-allocation analysis. It is also a tool for decision makers for conducting quick visual comparison of location plans.

This paper is a description of the research project. The next section introduces the basic concepts of location-allocation analysis in general and the application of such analysis to retailing. The third section describes the characteristics and functionalities of the Loc application program. Since the

secondary objective of this research project is to explore the programming features of Microsoft Windows, the fourth section is a detailed discussion of the Windows programming environment and its object-oriented nature in particular. Some future enhancements of the application program are suggested in the last section.

2. LOCATION-ALLOCATION ANALYSIS IN GENERAL

2.1 Objectives and Characteristics of Location-Allocation Analysis

The accessibility to a service can always be maximized by locating one facility at each concentration of demand. However, it is often infeasible because of the high fixed costs. To take advantage of scale economies, it is more profitable to provide the service from a few central locations. A location-allocation analysis, in its most general form, is to determine the optimal locations of the central facilities and assignments of flows between the facilities and the demand concentrations, so that the total costs of operation are the least possible. In other words, the basic question is how to locate and allocate facilities so that the consumers are best served and the facilities are best utilized.

Location-allocation models are useful in practical planning. There have been studies on its application to a wide range of services of both public sector and private sector. Examples include warehouses, restaurants, schools, police stations, fire stations, and hospitals. Application to different types of services requires modification of the most general form, but all location-allocation models are comprised of a few basic components, and the results of the analysis always contribute to the planning of service provision in the most efficient way.

2.1.1 Five Components of a Location-Allocation Model

The five components of a location-allocation model include a set of demand points, a set of feasible sites, a distance matrix, an allocation rule, and an objective function [9].

Demand Points

To decide on the optimal locations and allocations of new facilities, the

distribution of demand must be known. In reality, distribution of demand is spatially dispersed on a continuous space; however, for ease of analysis, demand in location-allocation models is represented by discrete spatial co-ordinates called demand points. Each of the demand points is usually the centroid of a zone containing a certain level of demand, which is usually the population size weighted by some economic or social factors. The demand zones are small and compact areal units, such as census tracts, zip code areas, or counties, for which population data are easily gathered from the population census.

Feasible Sites

In some cases, a location-allocation model is applied to a discrete space. Travel is restricted to a network and demand points are nodes on the network. Then the optimal locations are confined to the network also. In other cases, travel is not restrained to a network. Still, the optimal solutions are restricted to a number of locations called the feasible sites. The restriction could reflect the minimal requirements or limitations of land availability, ease of access, or government zoning code. The selection of feasible sites is time-consuming but renders a more realistic analysis.

Other location-allocation models, on the other hand, eliminate the step of identifying feasible solutions and assume that any point in the area under study is a feasible site, that means no location is considered infeasible. These models are also termed "planar" models.

Distance Matrix

A distance matrix used in a location-allocation model records the shortest distance between each demand point and each feasible site. A simple approach for preparing a distance matrix is to calculate distances mathematically based on a co-ordinate system. There are mainly two types of such mathematical distances, Euclidean and city block distances. The Euclidean distance is a straight-line distance and is calculated with the following equation, given two points with co-ordinates (x_i, y_i) and (x_j, y_j),

$$d_{ij} = \left((x_i - x_j)^2 + (y_i - y_j)^2 \right)^{\frac{1}{2}}. \quad (1)$$

The city block method measures the distance between two points as the sum of the absolute differences of the horizontal and vertical co-ordinates as follows,

$$d_{ij} = |x_i - x_j| + |y_i - y_j|. \quad (2)$$

The city block distance is equivalent to that of a right-angle path between two points, and thus it is a better approximation of the actual travel distance in a small area than the Euclidean distance while the latter gives a close

approximation to average total distance between several pairs of cities in a region [14].

Another way of preparing distance matrix is to measure physical distances or travel time. To accurately reflect actual distance or time, factors such as physical and social barrier, the modes of transportation available, traffic congestion, travel speed limits, and so on, must be taken into account. Preparing such a distance matrix requires the collection of extensive physical, social, and economic information.

Although co-ordinate based distance calculations, both the city-block and Euclidean, are not as accurate as measuring physical distance or time, they can be easily prepared from a map without collecting detailed information about road networks.

Allocation Rule

The allocation rule specifies the manner in which demand are assigned to the set of optimal supply points. In basic location-allocation models, the allocation rule simply follows the proximal area principle, that is, a demand point is allocated to its nearest supply point. For different applications, various additional parameters may supplement distance for forming an allocation rule that is more relevant to the particular context. For example, physical attributes of retail outlets are important factors, besides distance, in influencing location decisions for retail activities. The allocation rule for such applications has to be adjusted to reflect these non-distance factors.

Objective function

The objective function of a location-allocation model states the objectives to be optimized. It reflects the allocation rule. For instance, when using the proximal area principle as the allocation rule, the objective function is to minimize total distances and can be simply stated as

$$\text{Minimize} \quad \sum_i \sum_j w_i \, d_{ij} \tag{3}$$

where w_i is some form of weight such as population size of demand zone i and d_{ij} is distance between demand zone i and supply center j. The objective function serves as the basis for comparison of alternative location plans.

2.1.2 The p-median Problem

The earliest form of a location-allocation model is expressed by Alfred Weber [22], who studied the location of a factory such that the total

transportation costs from a raw material site to the factory and from the factory to a market can be minimized by using mechanical and graphical methods. Weber's model is limited to the location of a single facility.

Cooper [3] extended the model to study the flows between a set of facilities and demand. He formulated the p-median model which became the classical location-allocation problem. The objective of the p-median problem is to find the optimal locations of a given number (p) of uncapacitated supply centers that **minimize the aggregate weighted distance** to serve a set of demand points with known positions. The assumption is that each demand point is allocated to only one center and the center is the nearest to it [1]. A p-median problem is so called because the optimal locations are indeed median locations which favor the areas of greater densities and are insensitive to locations at the periphery. On the other hand, extensions of the p-median problem, as discussed in the next section, produce optimal solutions that are not median points.

2.1.3 Extensions of the p-median Problem

The p-median problem is the simplest classical form of location-allocation model that uses the nearest-center assumption as the allocation rule. When other constraints are applied, different location-allocation models can be formulated and each would be suitable for a particular type of services or goods. Examples of such extensions of the p-median problem are the p-center problem, the set-covering model and the maximal-covering model.

The p-center problem provides some minimum standard of individual access. It locates facilities so as to **minimize the maximum distance** between a demand point and its nearest facility. The solution produced by solving the p-center problem tends to be a more equitable location pattern with service areas of roughly equal sizes. On the other hand, the p-median problem yields a location pattern that exhibits a curvilinear relationship between demand density and facility density with relatively more facilities per unit area in areas of high demand density [8].

The set-covering model [5, 17, 21] is especially useful in designing a system of service industries, for example, emergency medical services. The objective of the model is to determine the locations of **the minimum number of facilities to serve all demand** such that each demand point is within some critical distance or time from the facility it is assigned to. Hillsman [11]

constructed a general mathematical structure termed a unified linear model which encompasses the p-median model and the set-covering model.

The maximal-covering model [2] is similar to the set-covering model in that they both have an accessibility constraint. However, the objective of the maximal-covering model is not to minimize the number of facilities but to determine the locations of a specified number of facilities such that the **proportion of demand covered by the facilities is maximized** and the covered demand points are all within the specified time or distance constraint. The model is particularly useful when the cost of providing universal covering service is too high to be feasible.

2.2 Solution Methods of Location-Allocation Problems

Among the existing literature on location-allocation analysis, there are four solution methods that are commonly mentioned: proximal solution methods, map overlay, linear programming, and heuristic algorithms.

Proximal solution, originated by Thiessen, is a simple geometric procedure for apportioning an area into regions about each center so that every location in a region is nearer to that region's center than to any other center [20]. Map overlay is a technique for allocating additional facilities. It is widely applied in medical geography for locating medical facilities [12]. These two methods are suitable only for small location-allocation problems. Linear programming, on the other hand, is a formal mathematical approach to optimization and is applicable to complex problems [11, 2, 13]. However, the linearity assumption precludes its use for problems with intricate non-linearities.

Heuristic algorithms are the most commonly used technique to solve location-allocation problems because of their simplicity and computational efficiency. A heuristic algorithm represents a set of rules and steps that regulates the iterative search procedure to produce solutions to a given problem, but the solutions are not necessarily the best possible [19]. The attainment of the best possible solution, the global optimum, depends on the nature of each problem. However, in many cases, the local optimal solutions are very good solutions and are not radically different from the global optimum. With this flexibility, many complicated problems can be solved with heuristic algorithms.

Cooper [4] suggested two heuristic approaches which are the framework for most of the heuristic algorithms used in location-allocation analysis. The first

type is called the destination subset algorithm. For a set of n destinations and m sources, the algorithm considers possible subsets of m destinations from the total set of n destinations and chooses the best. The subsets considered can be all possible combinations, can be generated in a random fashion (random destination algorithm), or can be formulated successively (successive approximations algorithm).

The second type is called the alternate location and allocation algorithm [4, 1]. The idea of this algorithm is to divide the n destinations into m groups of approximately equal size. For each group, a single source optimum is calculated with exact location method. The next step is to examine all destinations and see if there is any one which is possibly closer to another source than the one it was initially allocated. If there is any changes in allocation, sources are located in groups using the exact location method again. The process repeats until there is no change in allocation. This approach is a monotone-decreasing convergent process but it is not guaranteed that it converges to the global optimal solution. A few repeated runs using different starting values increase the chance of getting very close to the global optimum. When compared to the destination subset approach, the alternating approach is less costly and thus it is more commonly used [14].

2.3 Location-Allocation Analysis Applied to Retail Activities

There is a large body of literature on the application of location-allocation analysis to public sector services. However, relatively few studies exist on its application to retailing because of its complexity and the lack of reasonably accurate models [7]. In general, location-allocation analysis when applied to retailing determines the optimal locations of retail outlets such that the accessibility to customers is maximized. Since the cost of retail travel is incurred on customers, distance and travel time has significant impact on their decision of patronage. In the simplest case, it can be assumed that customers tend to patronize the nearest retail outlet and each outlet serves a well-defined market area. Therefore, the greater the accessibility, the higher the total level of patronage, and the higher the profitability. This assumption is the basis of the two distance-based models, the competition-ignoring model and the market-share model. On the other hand, the interaction-based models assume that distance, although important, is not the sole determining factor of customer

patronage; other factors such as store size and advertizing can also influence allocation of customers to stores.

2.3.1 Distance-Based Models

A location-allocation model of public service facilities deals with only one set of locations. On the other hand, in retailing, one may need to consider more than one set of locations, a set for the client firm, and other sets for its competitors. The market area of a firm's outlet can be highly influenced by that of other firms. In an area where there is no competing suppliers, all demand points are assigned to the outlets of the single firm which thus enjoys the whole market area. Then the location-allocation analysis of its new outlets is basically a p-median problem. On the other hand, when there are competitors, a firm may choose to locate its outlets optimally so as to attain the largest share of the market area. Depending on the different approaches to competition, two models were developed [8, 10] and implemented in Loc. They are named the competition-ignoring model and the market-share model.

The Competition-Ignoring Model (CIM)

The CIM ignores the locations of competing firms. It represents an aggressive strategy and maximizes market penetration. It assumes that locating outlets near to competitors' may indeed increase the number of potential customers, provided that the competing outlets are located in areas of high demand and easy accessibility. The model is particularly applicable to specialized products and services, for example, a particular type of restaurant, which has a strong image and high store loyalty. In such case, it is assumed that customers first decide on the particular chain to patronize and then chooses the nearest outlet of that chain [9]. Hence, the CIM attempts to maximize the potential sales and profits of a retail firm by locating its outlets closest to the maximum number of customers, regardless of the locations of competitors.

When the nearest-center allocation rule is used, the objective function of the CIM minimizes the total distances travelled by customers to their nearest outlets and is stated as follows [9]

$$\text{Minimize} \quad \sum_i \sum_{j \in J} a_{ij} w_i d_{ij} \tag{4}$$

where w_i is the relative demand of zone i, J is a subset of the feasible sites, d_{ij} is the distance between demand zone i and facility j calculated with either equation (1) or (2), and

$a_{ij} = 1$ if $d_{ij} < d_{ik}$ for all $k \in J$, $k \neq j$; 0 otherwise.

a_{ij} considers only the distance factor and assigns demand zones to each outlet. It takes a value of 1 when outlet j is closest to zone i and 0 for all other outlets; hence, the objective function only sums the shortest distances from each zone to its nearest outlet, as stated in the nearest-center allocation rule. The objective function (5) is subject to the following constraint,

$$\sum_{j \in J} x_{ij} = p$$

which makes sure that only p outlets are located.

With the assumption that demand has an opposite linear relationship with distance, minimizing total travel distance is equivalent to maximizing total demand.

The Market-Share Model (MSM)

The MSM, unlike the CIM, takes into consideration the presence of competing firms when locating outlets of a firm. It assumes that customers patronize the nearest outlet, irrespective of the chain store that operates the outlet. This assumption is particularly applicable to convenience-oriented stores and to low-order goods such as gasoline. The objective function of the model is

$$\text{Maximize} \quad \sum_{i} \sum_{j \in J} a_{ij} w_i \qquad (5)$$

$a_{ij} = 1$ if $d_{ij} < d_{ik}$ for all $k \in J \cup C$, $k \neq j$, 0 otherwise

where C is the set of competing outlets.

The model locates new outlets such that the total demand within the proximal area of the outlets is maximized. In other words, it maximizes the market share of the client firm by finding gaps among the market areas of competitors and locating outlets in the inadequately served areas. Thus, the MSM represents a more conservative corporate strategy than the CIM [9].

The two distance-based models are considered short-run strategies because they ignore the possible response from the competitors. The MSM assumes that the locations of competing outlets are known in advance and remain fixed throughout the lifetime of the client outlets. The CIM ignores the competing firms altogether and does not make any assumption about their reaction. Hence, the CIM may indeed be more rational than the MSM, given that the reaction of competitors is highly unpredictable [8].

2.3.2 Interaction-Based Models

Both the CIM and the MSM are distance-based models as they consider solely the distance factor. However, unlike assigning consumers to public services, customers may be influenced by non-distance factors in choosing the retail outlet they patronize. Thus, the nearest-center allocation rule may not be accurate in most cases. It may be necessary to take into consideration other store characteristics such as the size of stores. So besides these two distance-based models, there is a common type of location-allocation models for retailing analysis, called the interaction-based model.

This model assumes that consumer shopping patterns are determined by both distance and non-distance factors. To determine more accurately the allocation of demand, it may be necessary to observe consumer spatial behavior with respect to a set of existing retail outlets, and then use the observation to calibrate a spatial interaction model which can be applied to new outlets [8]. A typical spatial interaction model is in the form of [1]

$$S_{ij} = w_i A_j^{\alpha} e^{-\tau d_{ij}} \; / \; \sum_j A_j^{\alpha} e^{-\tau d_{ij}} \qquad (6)$$

where S_{ij} is the interaction between points i and j; w_i is the quantity demanded at zone i; A_j is some measure of attractiveness of center j; d_{ij} is the distance between i and j; α and τ are parameters measuring the relative importance of the attractiveness factor and the distance factor respectively. When consumers patronize only the nearest center, the parameter τ is infinite. But when non-distance factors become important to consumer decisions, τ tends towards 0 and distance factor is less crucial. The spatial interaction model can be incorporated into the nearest-center assumption to derive the allocation rule and to formulate the objective function [1, 10]. The objective function can be stated as

$$\text{Minimize} \quad \sum_i \sum_j A_i d_{ij} \; . \qquad (7)$$

For instance, one observation of consumer behavior is that consumers consider store size important and they are willing to travel further to patronize larger stores. In other words, while distance has a negative effect on demand, store size exerts a positive effect. Hence, a demand point which has close proximity to a small outlet may still be included in the market area of a further away but larger outlet because of the higher attractiveness of the latter due to its size.

However, to derive an accurate spatial interaction model is often a

difficult and time-consuming task. The attraction of a retail outlet includes not only its physical characteristics, but also other short-term variables such as price and advertising [8]. It is costly to collect data and calibrate a model with high predicting power. Moreover, the more complex the model is, the more sensitive it is to future uncertainty. The advantage of the distance-based models over the interaction-based models is their simplicity and insensitivity to future uncertainties. For low-order retail activities, that are goods and services purchased frequently, distance is known to be a significant factor in consumer shopping decisions. Even for high-order goods, when detailed information on consumer pattern is hard to collect, distance-based models can provide a quick and easy analysis of the highly accessible sites.

3. THE LOC APPLICATION PROGRAM

3.1 Objectives and Characteristics of Loc

The application program Loc is an interactive, PC (personal computer) based program that embraces most of the location-allocation concepts discussed in the previous section. The main objectives of Loc are to:

- provide a user-friendly interface to help retail planners to conduct quick and easy analysis of location plans,
- present a graphical interface for uncomplicated location-allocation algorithms of retailing,
- serve as an instructional tool for the basic concepts of location-allocation problems.

The program provides two approaches for analyzing retail locations. The first one is to adopt the competition-ignoring model to determine a specified number of retail locations. The second one is to allow a user to interactively select locations which are evaluated against an objective function. The impact of competition and the effect of store size can be incorporated into the second approach.

When a firm decides to open a number of new stores, it is plausible that it first searches for all available sites which meet its corporate strategies, and then chooses among the potential sites. Therefore, the second approach of Loc, providing the capability of allowing a user to select locations, is in line

with the practice in reality. A user can obtain reports of market shares and distances for different sets of prospective sites by specifying their locations one set at a time. The user may also obtain a set of optimal locations by the first approach and use the results as the yardstick for comparing the performance of the various sets of potential locations. A set that shows results close to the optimal is generally a good choice, while one that has performance measures far from the optimal may need to be reevaluated.

Loc provides a graphical user interface to allow users to perform a location-allocation analysis and to evaluate the results. Loc makes use of the Graphics Device Interface of the Microsoft Windows, which will be discussed Section 4.2. The program displays maps of demand distribution on a window and also plots the user-selected locations and the optimized locations on the maps. So it is easy for the users to correlate the results with the physical layout.

The graphical interface of Loc is also user-friendly. Input controls are graphical objects like pull-down menus, buttons and scroll bars. Users can specify parameters easily by manipulating the graphical objects with a pointing device such as a mouse. Hence, Loc is very easy to use. Thus, many alternative location plans can be evaluated in a shorter time.

Besides serving as a quick and easy tool for analyzing alternative locations, Loc has high educational value. As mentioned, most of the basic concepts of location-allocation models for retailing are incorporated in Loc. Students can change the location parameters easily and observe the effects of the parameters on the objective functions. They can then quickly appreciate the underlying concepts in the study of retail location.

3.2 The Five Location-Allocation Components in Loc

As discussed in the previous section, the five components of a location-allocation model are a set of demand points, a set of feasible sites, a distance matrix, an allocation rule, and an objective function.

Demand Points

Loc reads demand data from an IMAP document file which contains an array of some demand variable for areal units of the area of interest. IMAP is an instructional program developed by Renwick and Young [16] for analysis of geographic phenomena. The detailed discussion of IMAP document and image files can be found in [16].

The demand variable array that Loc obtains from an IMAP document file can be population sizes. The areal units can be zip code areas or census tracts. The IMAP document file also contains the name of its corresponding IMAP image file which provides the boundary information of the areal units. Loc uses the boundary information to calculate the centroid of each areal unit. The centroids of all areal units become the set of demand points. The data that Loc collected from an IMAP document file is listed in Appendix A.

Feasible Sites

In Loc, a location-allocation analysis is conducted on a continuous space. When a user chooses to perform an optimization analysis to obtain a set of optimal locations, there is no restriction on the possible solutions. Also when a user chooses to evaluate a location plan by selecting his/her locations, any point on a map can be picked. It means that the set of feasible sites in Loc contains all the points in the area under study.

Distance Matrix

Loc provides two distance calculation methods, the Euclidean distance (equation 1) and the city block distance (equation 2). The chief advantage of these co-ordinate based distance calculations, as discussed in Section 2.1.1, is that they eliminate the need for the expensive overhead of collecting road network information.

Allocation Rule

As stated in Section 3.1, Loc provides two approaches of analyzing retail locations. When the competition-ignoring model is used to calculate a specified number of locations, the nearest-center allocation rule is used and a demand point is assigned to its nearest retail outlet.

On the other hand, when a user selects locations and makes use of Loc to analyze them, Loc provides a choice of two allocation rules. One is the nearest-center allocation rule as used above. The other is allocation according to store-size attraction. This rule reflects the importance of store-size in affecting a consumer behavior pattern. As discussed in Section 2.3, consumers tend to travel further to patronize larger stores. So under the store-size attraction allocation rule, a demand point is allocated to a supply center which has the largest attraction value, A_{ij}, with respect to the demand point. A_{ij} is measured as

$$A_{ij} = S_j / d_{ij} \qquad (8)$$

where S_j is the store size of outlet j, d_{ij} is the distance between demand zone

i and outlet j. The equation shows that attraction of an outlet is positively affected by its store size and negatively affected by its accessibility.

Objective Function

As mentioned, Loc has two approaches for location analysis. In the first case, a user specifies a number of outlets and then Loc calculates the optimal locations. The nearest-center allocation rule is used in this approach. If the number of outlets to locate is one, it is a single facility location-allocation problem and the objective function is in the form of

$$\text{Minimize } Z = \sum_i w_i d_i \qquad (9)$$

where w_i is the relative demand of zone i, and d_i is distance between zone i and the single optimal supply point as calculated with either equation (1) or (2) of Section 2.1.1. Loc finds the co-ordinates of the single optimal location so that the total of d_i is minimized.

To find the optimal locations for multiple facilities, the objective function to use is similar to that of the competition ignoring model, equation (4) of Section 2.3 as follows,

$$\text{Minimize } Z = \sum_i \sum_j a_{ij} w_i d_{ij} \qquad (10)$$

where d_{ij} is the distance between demand zone i and outlet j, and
$a_{ij} = 1$ if $d_{ij} < d_{ik}$ for all k , $k \neq j$; 0 otherwise.

The second approach of analysis is for a user to select a number of locations on a map and have Loc evaluate the selection against some objective functions. Three cases are possible and each has a different objective function.

- *Competition and store size factor ignored:*

This problem is equivalent to the competition-ignoring model using nearest-center allocation rule. The objective function used is the same as equation (10).

- *Competition ignored, store size factor considered:*

This problem is a modified competition-ignoring model using the store-size attraction allocation rule. Instead of allocating a demand zone to its nearest center, it is allocated to the outlet with the largest attraction value, A_{ij}, as given in equation (8). Since demand is inversely related to distance, minimizing aggregate distance is equivalent to maximizing demand. Hence, the objective function can be similar to the one used by the market share model as equation (5) of Section 2.3. It can be written in the form of

$$\text{Maximize } Z = \sum_i \sum_j a_{ij} w_i \qquad (11)$$

where

$a_{ij} = 1$ if $A_{ij} > A_{ik}$ for all k, $k \neq j$; 0 otherwise.

- Competition and store size factor considered:

This is a market share model using the store-size attraction allocation rule. The objective function is the same as the previous equation (11). Only the allocation definition is different, as follows

$$\text{Maximize } Z = \sum_i \sum_j a_{ij} w_i \qquad (12)$$

where

$a_{ij} = 1$ if $A_{ij} > A_{ik}$ for all $k \in J \cup C$, $k \neq j$; 0 otherwise.

J is the set of feasible sites of the client firm, and C is the set of competing locations. In this case, the attraction values of all the client locations (set J) as well as all the competing locations (set C) have to be compared for allocating a demand point.

3.3 The Solutions Methods Used in Loc

The two approaches provided by Loc require different analysis. The competition-ignoring optimization uses heuristic algorithms to find out the requested number of optimal points. On the other hand, analyzing points that are selected by users is less complex. It is not necessary to conduct location and allocation at the same time; so heuristic algorithms are not needed. Instead, the user-selected points are evaluated by finding out the assignment of demand points with respect to the appropriate allocation rule.

Case 1: Calculation of Single Optimal Location

Locating a single optimum is a much simpler problem than determining multiple locations. An exact iterative technique as illustrated in [18] can be used to find out the optimum $O(x_0, y_0)$. The algorithm is comprised of the following steps.

Step 1: Compute the centroid as the initial location

$$x_0 = \sum_i x_i / n \qquad (13)$$

$$y_0 = \sum_i y_i / n \qquad (14)$$

where (x_i, y_i) is the co-ordinates of demand point i, and n is the number of

demand points.

Step 2: Find a better location towards the high density area as determined by

$$x_0^* = \sum_i [x_i w_i / d(0,i)] \; / \; \sum_i [w_i / d(0,i)] \qquad (15)$$

$$y_0^* = \sum_i [y_i w_i / d(0,i)] \; / \; \sum_i [w_i / d(0,i)] \qquad (16)$$

where d(0,i) is defined by either equation (1) or (2).

Step 3: Obtain the differences in the co-ordinates, (x_0^*, y_0^*) and (x_0, y_0). If both of the differences in x-coordinates and y-coordinates are less than an arbitrarily small number, say 0.05, then the point (x_0^*, y_0^*) is the optimal point; otherwise, return to step 2 and recompute the equations with (x_0^*, y_0^*) in place of (x_0, y_0).

For succeeding values of (x_0^*, y_0^*) as calculated at each round of step 2, the value of the objective function (9) should decline. The terminating point should be very close to the global optimum.

Case 2: Calculation of Multiple Optimal Locations

A location-allocation problem involving multiple optimal centers is complicated because the best location of any one center is highly affected by all other centers in the same system. The optimal solution yielded by a heuristic algorithm has a smaller chance of being close to the global optimal solution than in the single-center case. Nevertheless, an exact solution method is difficult for a problem on a continuous space with no restrained set of feasible sites. Hence, to provide a rough but quick analysis, Loc uses the alternating heuristic algorithm which is discussed in Section 2.2. The steps involved are as follows [18].

Step 1: Select a set of initial locations.

Step 2: Form groups of demand points by allocating each to its nearest center, and then calculate Z as in equation (10).

Step 3: For each group of demand points, calculate a new center location by applying the exact iterative technique such that

$$x_j^* = \sum_i [a_{ij} x_i w_i / d(i,j)] \; / \; \sum_i [a_{ij} w_i / d(i,j)] \qquad (17)$$

$$y_j^* = \sum_i [a_{ij} y_i w_i / d(i,j)] \; / \; \sum_i [a_{ij} w_i / d(i,j)]. \qquad (18)$$

Step 4: Return to Step 2 and repeat until the change in the Z value is arbitrarily small, say 0.05.

The resulting locations (x_j, y_j) of the n retail outlets are affected by

the set of initial locations used at step 1. The algorithm, as discussed in [18] has not specified a recommended way to generate these initial locations. Loc provides two options of designating the set of initial locations. The default is to use the equations shown in Table 1 to calculate the initial points. The first point lies roughly at the center of the map area surrounded by all the other points. The underlying rationale is that the center of the map area has a higher probability of having a high demand density.

The second option of designating initial locations is for the user to specify the points. The advantage of this option is that the user can examine the impact of different sets of initial locations on the calculated optimal points. It is desirable for the user to repeat the same analysis with different sets of starting locations and then choose the locations yielding the lowest average weighted distance.

Although the algorithm as discussed in [18] has not stated that it is limited to any type of distances, an observation of its performance in Loc is that the resulting locations tend to be more affected by initial locations when using city-block distance calculation than Euclidean distance calculation. In other words, when Euclidean distance method is used, different sets of initial locations produce resulting locations that are very close in every case; hence,

Table 1 : Equations for A Initial Set of Locations (Default Option)

(x_1, y_1)	=	(ncol * 0.5, nrow * 0.5)
(x_2, y_2)	=	(ncol * 0.6, nrow * 0.6)
(x_3, y_3)	=	(ncol * 0.4, nrow * 0.4)
(x_4, y_4)	=	(ncol * 0.5, nrow * 0.6)
(x_5, y_5)	=	(ncol * 0.5, nrow * 0.4)
(x_6, y_6)	=	(ncol * 0.6, nrow * 0.5)
(x_7, y_7)	=	(ncol * 0.4, nrow * 0.5)
(x_8, y_8)	=	(ncol * 0.6, nrow * 0.4)
(x_9, y_9)	=	(ncol * 0.4, nrow * 0.6)
(x_{10}, y_{10})	=	$(x_1+(x_1-x_7)/2,\ y_1+(y_1-y_7/2)$
(x_{11}, y_{11})	=	$(x_1+(x_1-x_8)/2,\ y_1+(y_1-y_8/2)$
.		.
.		.
.		.

where ncol = the maximum horizontal extent of the map area
and nrow = the maximum vertical extent of the map area

the chance of approximating the global optima is high. So it could be concluded that the algorithm works better with Euclidean distance calculation method than with the city-block method.

Case 3: Analyzing User-Selected Locations (competition and size factor ignored)

In this case, Loc examines all demand points one by one. According to the nearest-center allocation rule, each point is assigned to the user-selected location that has the shortest distance from the point. The distance is calculated according to the distance calculation method specified by the user.

Case 4: Analyzing User-Selected Locations (competition ignored; size factor considered)

The allocation rule in this case is the store-size attraction allocation and the objective function is equation (11) of the last section. Hence, for each demand point, Loc first calculates its distances from all user-selected locations (d_{ij}), and divides the store sizes (S_j) of the locations by the corresponding distances to obtain the store size attraction values as in equation (8). Loc then assigns the demand point to the user-selected location that has the highest attraction value.

Case 5: Analyzing User-Selected Locations (competition and size factor considered)

The method of allocation is similar to the previous case (Case 4). The difference is that besides the user-picked locations, Loc also needs to consider the competing locations. So for each demand point, Loc first calculates its distances from all user-selected points as well as competing points. Then it calculates the store size attraction values as discussed in the previous case, and assigns the demand point to the location that has the highest attraction value, either belonging to the set of the user-picked locations or to the set of the competing locations.

In all cases, after assigning demand points to retail centers, Loc calculates the average distance and the percentage market share for each center. The average distance of a center is defined as the aggregate distance from all demand points assigned to it divided by the number of the assigned demand points.

The market share percentage of a center j, M_j, is calculated as in

$$M_j = (\sum_i a_{ij} w_i / \sum_i w_i) * 100 \quad (19)$$

where a_{ij} = 1 if demand zone i is assigned to center j; 0 otherwise.

For each set of retail locations, Loc also calculates the average weighted distance, WD, by using the following equation,

$$WD = \sum_j M_j/100 * t_j \quad (20)$$

where t_j is the average distance of center j.

3.4 Functionalities and Menu Options of Loc

When a user runs Loc on Microsoft Windows, a single main window appears. The window has a system menu bar on line 1 which will display the file name of the demand file after a user opens one. On the second line of the window, there are three pull down menus on the left and a menu option on the right, as shown in Figure 1. The menu options of the pull-down menus under FILE and MAP 1 are also illustrated in Figure 1. Under MAP 2, the pull-down menu is the same as that under MAP 1 since Loc displays two identical maps on the main window and allows a user to perform the same types of operations on each map. The three periods (...) that follow some of the menu options indicate that when a user select this option, a dialog box will appear to obtain or to provide additional information. For example, when a user clicks on the HELP... menu option, a large dialog box will show on the window with a help index and help information. A user may select a menu option by using a mouse to open the pull-down menu and point at the desired option.

All menu options, except OPEN DEMAND FILE, EXIT, ABOUT, and HELP, are inactive when a user first starts Loc. A demand file must first be opened before any analysis can proceed. Loc is programmed to read demand data only from IMAP files. Hence, when a user chooses to open a demand file, a dialog box appears and it lists all files with the extension '.idc', the extension of IMAP document files. After a file is successfully opened and the demand data are read, Loc obtains the IMAP image file name from the document file and reads the image file for boundary information. Then Loc paints two identical maps on the window. The maps can reflect the distribution of demand in the area by classifying demand of all areal units into five categories, each represented by a different color. An index is painted at the bottom of the window to show the ranges of the five

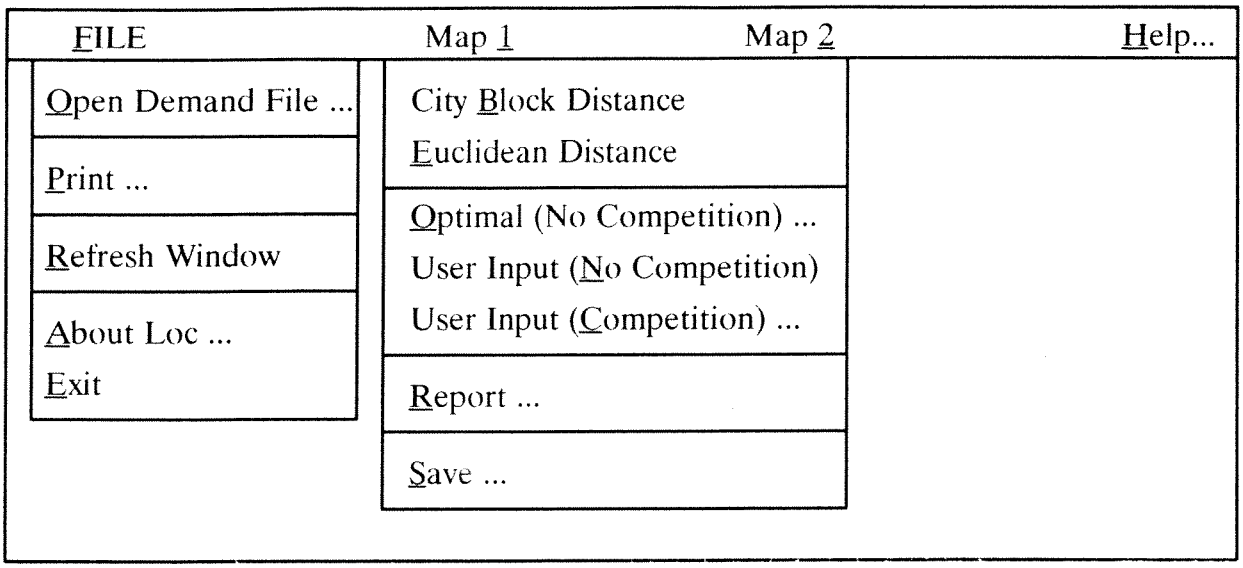

Fig 1. Menu Options Provided by Loc.

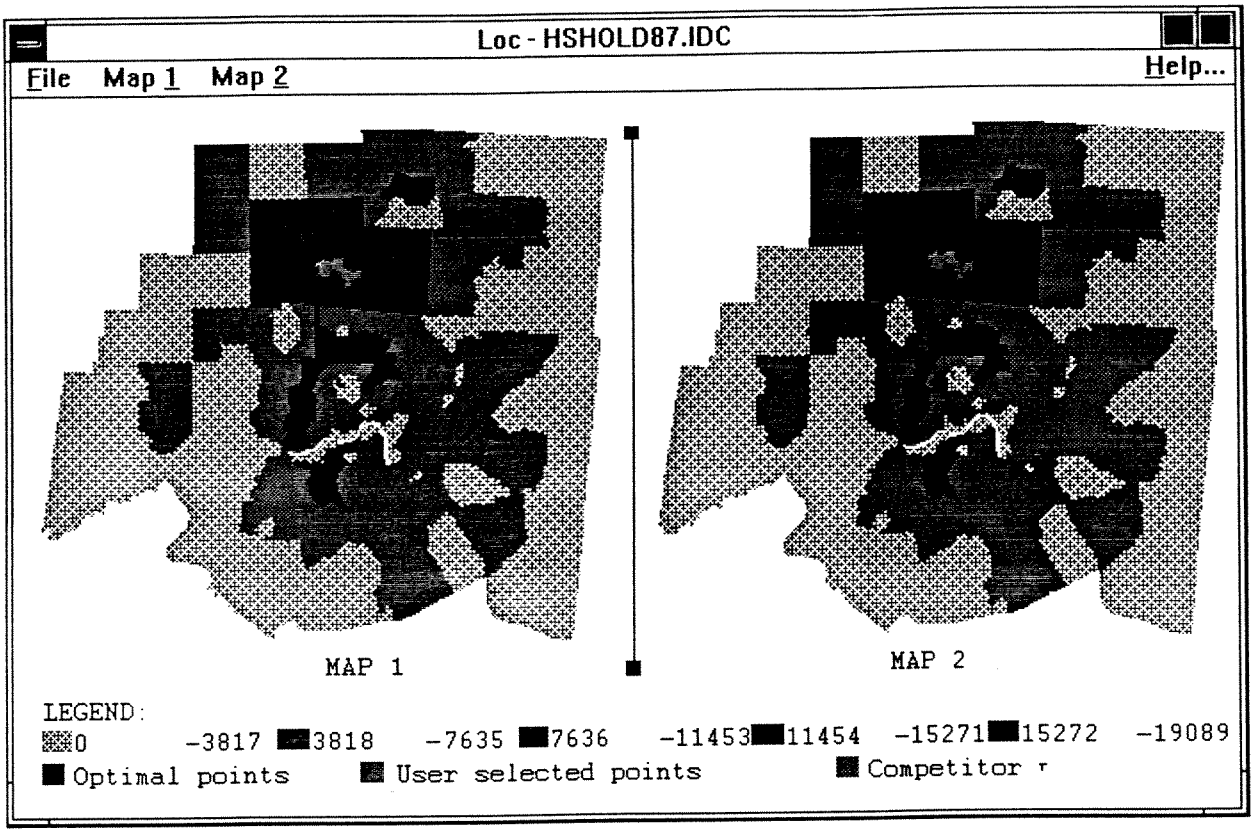

Fig 2. Loc window after a demand file is opened.

classes of demand. Figure 2 shows the main window after a demand file is opened. For each map, a user can perform any operation by selecting a menu option of the pull-down menu associated with the desired map.

PRINT is the second menu option under FILE. Loc supports printing of reports and maps. Examples are shown in Figure 3 and Figure 4. The maps are printed is in black and white, but they show the locations with index numbers that match with those on the corresponding reports.

The third menu option under FILE, REFRESH WINDOW, is necessary whenever the main window is not repainted properly. This may happen occasionally after closing a dialog box or after switching back to Loc from another Windows application. Loc repaints the two maps and the index and also redraws locations on the maps if there are any.

For each map, a user can choose from the two types of distance calculation methods, CITY BLOCK DISTANCE or EUCLIDEAN DISTANCE. The first one is the default. When a user changes the distance type by pointing to the option that has no check mark in front, Loc clears the map if there are any locations painted on it. Then the user can start a new analysis which will use the selected distance method.

The OPTIMAL (NO COMPETITION) option allows a user to perform the first approach of Loc, i.e. calculating optimal solutions based on the competition-ignoring model. When a user selects this option, Loc first displays a dialog box to ask the user to specify the number of centers to locate. The maximum number that can be calculated by Loc is ten. If the number specified by the user is one, then Loc will conduct the single-optimum location-allocation analysis following the steps in the exact iterative technique as listed in Case 1 of Section 3.3. If the number of centers to locate is larger than one, then Loc will display a dialog box and ask the user whether he/she wants to specify initial locations. If the user chooses not to do so, Loc uses the equations in Table 1 to designate the initial locations. Otherwise, the user can choose any points on the map by clicking with the mouse. Any selected point can also be erased by clicking on top of it. After the user finishes, Loc checks if the number of initial points selected by the user is the same as the number of centers to locate. If there is a mismatch, Loc will display a warning message and allow the user to apply corrections. Loc then calculates the multiple optimal locations by following the steps listed in Case 2 of Section 3.3. The calculated optimal points are painted in green on the map and the user-picked

```
Method: User Selected Locations with Competition
Distance Calculation: City Block Distance
Number of New Centers: 3
Number of Competitors: 4
File of Competitors: SUPFILE2.XYZ
Market Share of User Selected Points: 33.32
Market Share of Competitors: 66.68
Average Weighted Distance of User Selected Points: 24.64
Average Weighted Distance of Competitors: 32.57

User Selected Points:
  id      x         y       size      market share    average distance
   1   198.00    73.00     10.00       29.56            80.30
   2    83.00   156.00      5.00        2.01            37.87
   3   240.00   167.00      2.00        1.74             7.80

Competitor Points:
  id      x         y       size      market share    average distance
   4   111.10    80.40      5.00        1.97            33.79
   5   133.40   187.50      6.00       17.63            38.82
   6   201.40   150.45      6.00       37.94            40.86
   7   105.60   222.30     10.00        9.15           104.53
```

Fig 3. An Example of Reports Produced by Loc.

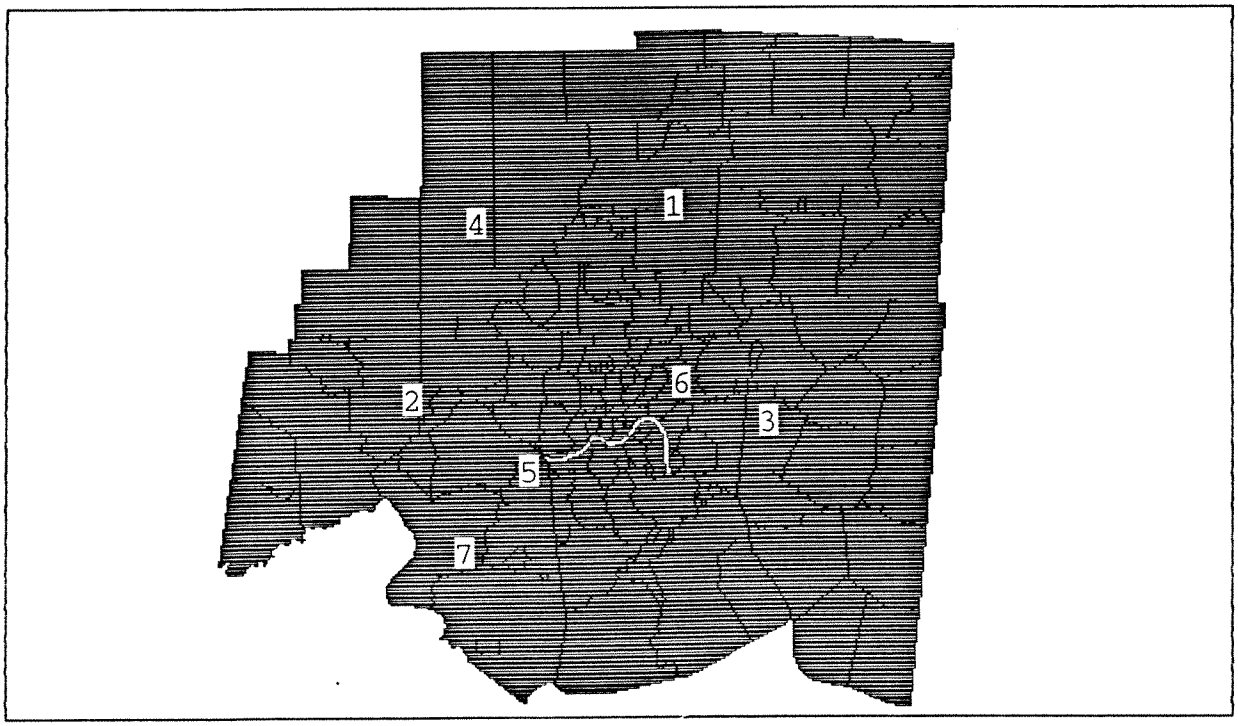

Fig 4. An Example of Maps Printed by Loc.

initial points are blue.

There are two options, USER INPUT (NO COMPETITION) and USER INPUT (COMPETITION), that allow a user to perform the second approach of Loc to analyze locations. When a user selects the first one, Loc shows a dialog box and asks if the user wants to specify a different size for each picked location. Then the user can select and deselect locations by clicking with the mouse on the map. The maximum number of points that a user can select is ten. If the user has chosen to specify size, after the user clicks on a location, Loc will display a dialog box to allow the user to input the size of the point just selected. After the user has finished picking locations, Loc assigns demand points by following either Case 3 of Section 3.3 if sizes are not defined or Case 4 if the user has specified sizes. Again, Loc paints the user-picked points on the map in blue.

The USER INPUT (COMPETITION) option allows a user to analyze his/her selected locations against competing locations. When a user chooses this option, Loc first displays a dialog box and asks the user to specify the type of competitor file to use. Loc can read two types of competitor files, the IMAP files and the "xyz" files. In the first case, Loc uses the centroids of the areal units as the competing locations, each having a size that is equal to the number of retail outlets in that areal unit. If the number is less than one, then the location is ignored. In the second case, Loc reads a file that has an extension of "xyz". A file of this type contains the x-coordinates, the y-coordinates, and the sizes of all competing locations. When a user chooses to open a competitor file of "xyz" type, Loc first checks if all the data in the file are floating point numbers and then checks if all the coordinates are within the map area before proceeding to analyze the locations. In either case, after obtaining the locations and their sizes, Loc paints all the competing points in pink on the map. It also calculates the minimum, maximum, and average size of the competing points and displays the values to the user in a dialog box. At the same time, Loc provides the user the choice of specifying a different size for each user-picked point or making all picked points uniform. Then Loc receives mouse clicks from the user to select locations and paints the points in blue. As in other analysis options, Loc can erase any points on the map if the user so desires.

After a user has performed an analysis, he/she can view the report of market shares and average distances by selecting the menu option REPORT. A report, as shown in Figure 3, contains information about the method of analysis

and the distance calculation used, the number of centers located, and the average weighted distance. For each point, it also shows the coordinates, the size (if it is specified), the market share percentage, and the average distance.

The last option that a user can perform on a map is SAVE. After conducting an analysis, a user can instruct Loc to save a file that contains the coordinates of the locations and the assignments of demand points to the locations. The file saved has an extension of "loc". An example of a "loc" file is shown in Figure 5. The format of this type of file is discussed in Appendix B.

```
USR_COMP
CBD
3
0 198.000000 73.000000 10.000000 29.563234 80.302498
1 83.000000 156.000000 5.000000 2.013453 37.868469
2 240.000000 167.000000 2.000000 1.744829 7.804962
SUPFILE2.XYZ
4
3 111.099998 80.400002 5.000000 1.966543 33.788929
4 133.399994 187.500000 6.000000 17.625694 38.823875
5 201.399994 150.449997 6.000000 37.936092 40.856060
6 105.599998 222.300003 10.000000 9.150154 104.526443
4 6  5 6  6 6  7 4  8 4  9 6  10 4  11 4  12 4  13 6
14 6  15 6  16 6  17 5  18 6  19 4  20 5  21 5  22 5  23 5
24 6  25 6  26 6  27 6  28 6  29 1  30 0  31 0  32 0  33 0
34 0  35 3  36 0  37 0  38 0  39 0  40 0  41 0  42 0  43 1
44 3  45 0  46 3  47 0  48 0  49 0  50 0  51 0  52 0  53 6
54 5  55 6  56 5  57 6  58 0  59 0  60 5  61 5  62 0  63 6
64 6  65 0  66 5  67 0  68 5  69 6  70 5  71 4  72 4  73 4
74 5  75 5  76 5  77 5  78 5  79 4  80 5  81 5  82 4  83 5
84 5  85 5  86 0  87 5  88 5  89 5  90 5  91 5  92 5  93 5
94 5  95 5  96 5  97 0  98 5  99 4  100 5  101 5  102 4  103 5
104 0  105 0  106 5  107 5  108 5  109 2  110 0  111 4  112 4  113 0
114 0  115 0  116 5  117 6  118 6  119 1  120 1  121 6  122 1
```

Fig 5. An Example of "loc" File Saved by Loc.

4. PROGRAMMING WITH WINDOWS

People with experience using Windows applications tend to have high comments about their user-friendliness, their ease of manipulation, and the tightened interaction between users and programs. On the other hand, people experiencing Windows programming have quite the opposite feelings. Indeed,

Petzold [15] commented that a normal first reaction to Windows programming is to find it "difficult, awkward, bizarrely convoluted, and filled with alien concepts". The "alien" concepts which are responsible for the sharp learning curve of Windows programming include the object-oriented nature of programming, the message-passing concept, and the Graphical Device Interface. These uncommon concepts make Windows programming very different from conventional programming with computer languages such as COBOL or Pascal. It is not exceptional for a Windows programmer, at least at the initial stage, to feel confused by these concepts as well as the 550 function calls supported by Windows.

4.1 Object-oriented Nature of Windows Programming

Windows programming is a form of object-oriented programming. Buttons, scroll bars, menus, dialog boxes, windows, etc. are all considered objects and they communicate with each other by passing messages.

The most important object of all in an application program is the window that the program uses to receive user input from the keyboard or mouse and to display graphical output on its surface. Items associated with a window are anchored to the window object itself and not to the screen. Therefore, the window can be moved together with all the items associated with it to a new location on the screen by a simple call. Items that are associated with the application window are also objects and they must be referenced with respect to the window object. For instance, an item to be displayed must be placed in the window relative to the window itself and not relative to the physical screen.

A handle is the means of describing and referencing an object such as a pull-down menu, a window or a dialog box. It is a sixteen-bit value used to name the object that it is describing. One type of handles is especially important for an application and that is instance handles which are used to keep track of an application program. Since Windows is a multitasking environment, multiple copies of an application can be running at the same time. An instance is created the first time an application is started. Instances are used by Windows as handles to the multiple copies of the application which are currently running. Each of the copies has a unique instance. The value of the prior instance of the application is also made known. In this way, Windows can effectively maintain only one copy of the code segments of the application to be shared by the various copies each with its own data.

Windows communicates with an application through messages and events. An event is some action that affects some objects. For instance, selecting a pull-down menu item or pressing the mouse button on a window are events. Windows has a rich set of predefined messages which are closely tied to common events. Since Windows resides between application programs and hardware (Figure 6), it handles all the interfacing tasks. Application programs rely on it for notification of any events arising from key-board, mouse, screen, or printer. Windows notices an event and sends a message containing the handle of the affected object to the application to which the object is associated.

A Windows application program always contains a function WinMain(), which is called by Windows when the application is started. The function is responsible for initializing its main window, executing any other start-up initialization, and informing Windows about some of the basic characteristics of the application and the name of the routine for processing messages for that window. WinMain() has a loop for preprocessing messages. Raw messages are translated and dispatched to the application's window-processing procedure, say WinProc(). Figure 6 illustrates this event-message processing in Windows.

For instance, when a user clicks the mouse button on a menu item, Windows responds to this mouse event by the following steps.

(1) Windows manages the physical tracking of the mouse. When a user points to an application and clicks the mouse button, it determines which window on the screen was clicked.

(2) Windows generates a mouse click event message and sends it to the processing loop in WinMain() of the involved application.

(3) WinMain() of the application takes the message and preprocess it and then instructs Windows to post it to the window-processing procedure, WinProc(), of that application.

(4) WinProc() processes the message and responds accordingly to the menu item selected.

This object-oriented nature of Windows environment facilitates programming because an application program can be broken down into sections, each dealing with an object. Since programs can be produced in a modular fashion according to the objects, new functions can be easily added and existing functions can be modified without affecting other parts of the programs. Object-orientation also helps to save development effort by means of instantiation and reusability. For instance, by defining a window class which identifies the procedure to process

the messages sent to the window, multiple windows can be created based on the same window class, sharing the same window procedure.

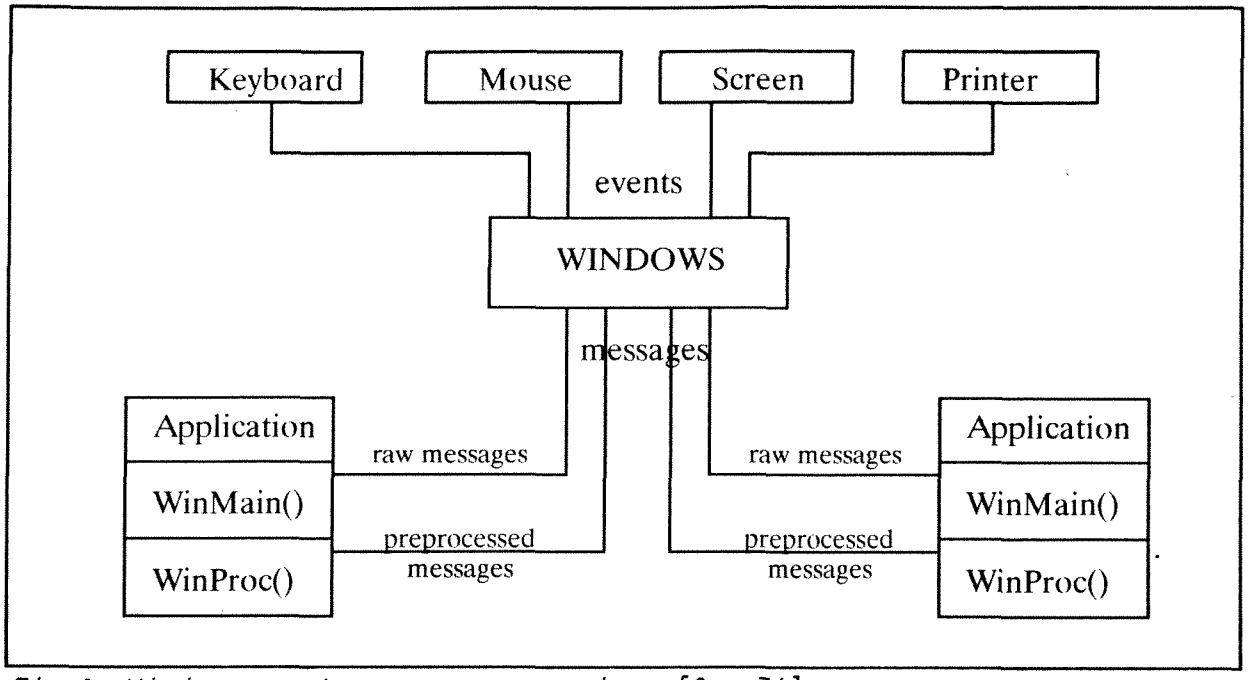

Fig 6. Windows event-message processing. [6:p.71]

4.2 Graphics Device Interface of Windows

Even though a Windows program is indeed basically a language C or C++ program with Windows function calls, the standard C output functions such as printf are not acceptable. This is because a Windows application has to display output on its window by use of a graphics programming language, called the Graphics Device Interface (GDI). The GDI supports the display of graphics and text to the screen or to the printer. It provides a rich set of built-in functions to facilitate the development of applications with a graphical user interface. Information can be conveyed more effectively than traditional video display by using a visually rich environment composed of graphical objects. The display context (DC) is a storage spot for Windows to store graphical objects including text. The DC is considered an object itself and has a unique handle.

Another advantage of the GDI is that it is device independent. Windows provides support for various devices and thus device driver programs are not necessary. This not only facilitates the development of Windows applications but also benefits users of Windows programs since the programs requires very little in the way of installation [15].

4.3 Programming Structure of Loc

Loc is coded in the language C using the Microsoft C 6.0 Compiler and the Microsoft Windows 3.0 Software Development Kit. The program consists of ten source program files, five header files, eleven help files, a resource script file, a dialog file, a module definition file, and a make file.

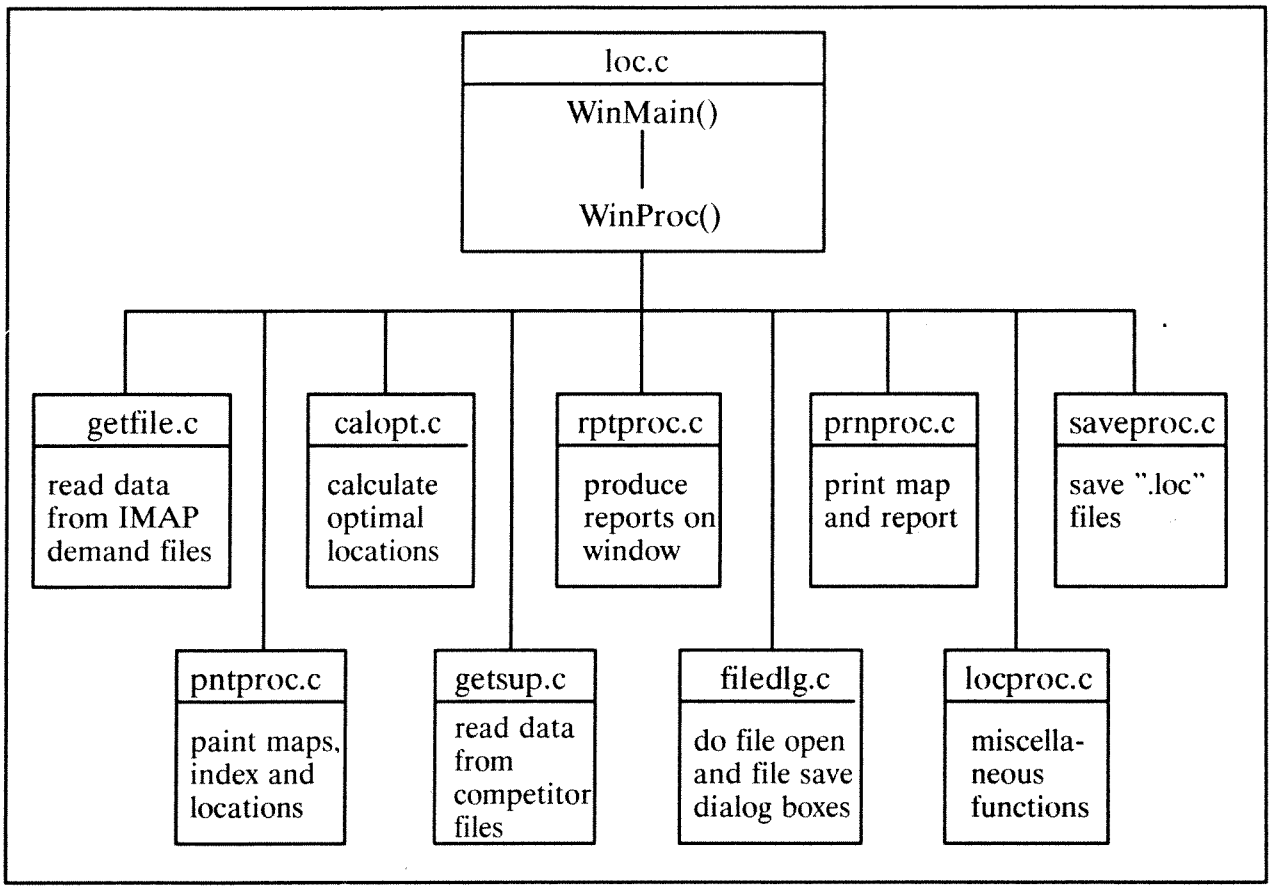

Fig 7. Program structure of Loc.

The loc.c program file contains the procedures, WinMain() and WinProc(). As illustrated in Section 4.1, WinMain() is the entry point to the program. It receives messages from Windows, preprocess the messages, and sends them to WinProc(). WinProc() interprets the preprocessed messages and calls the appropriate functions which are coded in the other nine source program files. Figure 7 illustrates the structure of the Loc program.

The resource script file and the dialog file defines the pull-down menus and the dialog boxes respectively.

The module definition file supplies information about the characteristics of the program's code segment and data segment in order to instruct Windows to maintain a separate data area for each instance of Loc but to make the code segment movable and sharable by all instances. The file also contains information about the size of the program's local heap and the size of the program's stack. The information is used by the linker in creating the executable file.

The make file records the dependencies between the source files and the header files and other dependent files. It is used to exploit the NMAKE utility of the Microsoft C Professional Development System.

The diskette enclosed in this report contains all the files of the program Loc. Appendix C lists the names and functions of the files.

5. CONCLUSION AND FUTURE ENHANCEMENTS

There are three objectives of this research project. The first is to design and implement a program that supports a user-friendly graphical interface for the location-allocation analysis of retailing activities. The second is to provide an educational tool for the concepts of location-allocation analysis. The third is to explore the programming features and characteristics of Microsoft Windows. This report is an account of the results attained. The Loc Windows application program makes use of the Graphics Device Interface of Windows to supply a user-friendly and interactive interface for users to conduct various kinds of analysis of retail locations. Loc's educational value is also illustrated in this paper.

A few enhancements to the Loc application may be desirable in the future:
(1) Computational efficiency and memory management: These issues were not major concerns when the program was being developed; rather the focus was on the user interface. When the program is used in the future to analyze large demand areas or to calculate more than ten centers, modifications have to be made to speed up the program and to handle the large amount of input data.
(2) Multiple optimal locations: At the time of the research, literature on algorithms for solving market-share models of multiple optimal location-allocation problems is scarce. Such an algorithm has to be defined in order to

add the functionality of calculating multiple optimal locations with competition taken into account.

(3) Printing color maps of demand distribution

(4) Addition of the option for users to specify feasible sites.

Even without these enhancements, the program Loc has fulfilled the three objectives of this research project.

REFERENCES

1. Beaumont J. R., "Location-allocation models and central place theory", in Ghosh, A. and Rushton, G. (eds.), *Spatial Analysis and Location-Allocation Models*, Van Nostrand Reinhold Company, New York, 21-54, 1987.

2. Church, R. L. and ReVelle, C. S., "Theoretical and computation links between the p-median, location set-covering and the maximal covering location problem" *Geographical Analysis*, 8:406-415, 1976.

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

4. Cooper, L., "Heuristic methods for location-allocation problems", *SIAM Review*, 6,1:37-53, 1964.

5. Daskin, M. S., and Stern, E. H., "A hierarchical objective set-covering model for emergency medical service vehicle deployment", *Transportation Science*, 15:137-152, 1981.

6. Farrell, T., *Programming with Windows*, Que Corporation, Carmel, Indiana, 1987.

7. Goodchild, M.F. and Noronha, V. T., "Location-allocation and impulsive shopping: the case of gasoline retailing", in Ghosh, A. and Rushton, G. (eds.), *Spatial Analysis and Location-Allocation Models*, Van Nostrand Reinhold Company, New York, 121-136, 1987.

8. Goodchild, M. F., "ILACS: a location-allocation model for retail site selection", *Journal of Retailing*, 60,1:84-100, 1984.

9. Ghosh, A. and McLafferty, S. L., "Developing retail-outlet networks", in *Location Strategies for Retail and Service Firms*, Lexington Books, Lexington, 127-178, 1987.

10. Ghosh, A. and Rushton, G., "Introduction: process in location-allocation modeling", in *Spatial Analysis and Location-Allocation Models*, Van Nostrand Reinhold Company, New York, 1-18, 1987.

11. Hillsman, E. L., " The p-median structure as a unified linear model for location-allocation analysis", *Environment and Planning A*, 16:305-318, 1984.

12. Jackman, M. E., "Flying doctor services in Zambia" in McGlashan, N.D. (ed.) *Medical Geography*, Methuen, London, 1972.

13. Jucker, J. V. and Carlson, R. C., "The simple plant-location problem under uncertainty", *Operations Research*, 24:1045-1055, 1976.

14. Love R. F., Morris, J. G., and Wesolowsky, G. O., Facilities Location: Models and Methods, North-Holland, New York, 1988.

15. Petzold, C., *Programming Windows*, 2nd Edition, Microsoft Press, Redmond, Washington, 1990.

16. Renwick, W. H. and Cyrus W. Y., *IMAP Users Manual, Version 1.1i*, Department of Geography, Miami University, Oxford, 1990.

17. ReVelle, C. S., Toregas, C., and Falkson, L., " Applications of the location set-covering problem", *Geographical Analysis*, 8:65-76, 1976.

18. Rushton, G., *Optimal Location of Facilities*, Compress Inc., Wentworth, 1979.

19. Scott, A. J., *Combinatorial Programming, Spatial Analysis and Planning*, Methuen, London, 1971.

20. Taylor, P. J., *Quantitative Methods in Geography: an Introduction to Spatial Analysis*, Houghton Miffin Company, London, 1977.

21. Toregas, C., Swain, R., ReVelle, C. S., and Bergman, L., "The location of emergency service facilities", *Operations Research*, 19:1363-1373, 1971.

22. Weber, A., *Theory of the Location of Industries*, University of Chicago Press, Chicago, 1929.

Appendix A

File Format of a IMAP Document File[‡] (Data used by Loc)

Line Number	Column Number	Data Type	Description
4	16-80	ascii	file name of the image file extension ibd: boundary file
5		binary int	file type 0: attribute file 1: image file
5	16-80	ascii	data type of image file byte: unsigned char integer: signed integer
7	16-80	ascii	number of rows in image file
8	16-80	ascii	number of columns in image file
14		binary int	number of values in doc file (nobs)
		8 binary int 2 binary long 24 binary int	not used not used not used
		nobs binary float	values for each areal unit in boundary file

[‡] The file format is supplied by Dr. Cyrus Young of Department of Geography, Miami University.

Appendix B

File Format of a ".loc" file saved by Loc

Line Number	Data Type	Description
1	ascii	method of analysis
2	ascii	distance calculation method (EUC or CBD)
3	integer	number of locations located (n)
n times of		
	integer	label (start from 0 to n-1)
	float	x-coordinate of location
	float	y-coordinate of location
	float	size of location
	float	market share of location
	float	average distance of location
if the method of analysis is USER SELECTED (COMPETITION):		
	ascii	name of competitor file
	integer	number of competing locations (nsup)
nsup times of		
	integer	label (start from n to n+nsup-1)
	float	x-coordinate of competing location
	float	y-coordinate of competing location
	float	size of competing location
	float	market share of competing location
	float	average distance of competing location
nobs times of (nobs = number of observations in document file)		
	integer	label of areal unit (demand point)
	integer	label of location assigned to the areal unit
if the method of analysis is OPTIMAL (with INITIAL LOCATIONS SPECIFIED):		
	integer	number of initial locations (n)
n times of		
	integer	label (start from 0 to n-1)
	float	x-coordinate of initial location
	float	y-coordinate of initial location
	float	size of initial location (0)
	float	market share of initial location
	float	average distance of initial location

Appendix C

Program Files of Loc

```
**********************************************************************
Header Files:

(1) LOCINIT.H -- global variables
(2) GENERAL.H -- define common constants
(3) LOC.H -- define ID of menu options
(4) METHOD.H -- define methods
(5) FILEIO.H -- define file constants
(6) LOCDLG.H -- define ID of dialog controls



**********************************************************************
C Files:

(1) LOC.C  -- Main Program

      WinMain : the window main program
      WinProc : the window procedure


(2) GETFILE.C -- Extract data from imap demand file

      getfiledata : Called by WinProc to read the imap file and
                      the image file
      goto_colon  : Skip the text label, called by read_doc
      skip_line   : Skip a number of lines, called by read_doc
      read_doc    : Extract data from the imap file and store them
                    into global and static variables
      read_img    : Read the image file and calculate the centroids
                      of areal units


(3) CALOPT.C -- Calculate optimal locations

      eudist   : Calculate euclidean distance
      cbdist   : Calculate city block distance
      one_nocp : Calculate the optimal location of a single center
      cal_z    : Calculate the z value (the total minimized distance value),
                   called by mult_nocp
      mult_nocp: Calculate the optimal locations of multiple centers


(4) SAVEPROC.C  -- Save ".loc" files

      savefile   : Get save file name, open the file and call write_file to
                     put data in. Called by WinProc.
      write_file : Write data into the save file
```

(5) RPTPROC.C -- Produce reports on window

```
show_rpt    : Depending on the specified algorithm, determine the
              kind of dialog box to use. Call the dialog box procedure
                to display the report.
                Called by WinProc.
cal_rpt     : Depending on the specified algorithm, call the corresponding
                function to calculate the average distances and market
              shares.
rpt_single  : Calculate the average distance for a single optimal pt
rpt_mult    : Calculate the average distances, market shares, and assignment
                for multiple pts, disregarding size effects
rpt_msize   : Calculate the average distances, market shares, and assignment
                for multiple pts, considering size effects
rpt_comp    : Calculate the average distances, market shares, and assignment
                for multiple pts, with competitors
wtdist      : Calculate the average weighted distance
FillBlank   : Replace blanks in a string with underscores
RptDlgProc  : Display the report in a dialog box
```

(6) PRNPROC.C -- Print maps and reports

```
print_proc       : Display dialog boxes and control printing,
                      called by WinProc
PrintRstDlgProc  : Collect the items that the user selects to print
GetPrinterDC     : Create a printer device context
PrintDlgProc     : The Print Abort dialog box procedure
AbortProc        : The Print Abort procedure
PrintMapRpt      : Set up procedure to print reports and/or maps
PrintRpt         : Print a report
PrintMap         : Print a map
```

(7) PNTPROC.C -- Paint procedures (functions) for the WM_PAINT message

```
startover : Reinitialize some variables, called after painting (clearing)
              a map
refresh   : Set the appropriate flags to refresh the window
paintnum  : Put numbers on the map for reporting purpose
paintidx  : Paint the index
paintmap  : Paint a map with five colors, each corresponding to a demand
              class
paintcal  : Paint the calculated optimal pts or paint all the user picked
              pts (only when refreshing window)
paintsel  : Paint or erase a used picked pt
examsup   : Examine if all the competitor pts are on the map area
paintsup  : Paint the set of competitor pts
```

(8) GETSUP.C -- Extract data from a competitor file (xyz type or imap type)

```
get_spc_supply : Read a competitor file of xyz type
get_doc_supply : Read a competitor file of imap type
```

(9) FILEDLG.C -- Open and Save File Dialog Boxes
(This c file is extracted from Petzold, Programming Windows, p.448-453, 2nd Edition, 1990.)

```
DoFileOpenDlg
DoFileSaveDlg
FileOpenDlgProc
FileSaveDlgProc
lstrchr
lstrrchr
```

(10) LOCPROC.C -- miscellaneous procedures (functions) for WinProc

```
initialize  : Initialize flags, check and enable menu items
              (called after opening a new demand file)
enable      : Enable menu items after they are disabled when the user is
              picking point
disablemenu : Disable menu options, called before the user starts to pick
              pts
uncheckmenu : Gray and uncheck menu options of report, save and algorithms
              after opening a new demand file or changing distance option
DoCaption   : Put the demand file name onto the window caption bar
OkMessageBox: Display a message box with two char strings
show_okbox  : Create a child window of button class and disable menu
              options
change_dist : Check selected distance menu option, and clear the map if
              necessary
drawopt     : Send message to paint calculated optimal pts, and check the
              selected algorithm menu option
calmult     : Prepare the user picked initial points for map 2, and call
              mult_nocp to calculate the optimal locations
AboutDlgProc    : A generic dialog procedure
SizeCompDlgProc : Get the option of specifying size for picked pts in the
                  case of USR_SELECTED (COMPETITION) menu option
GetSizeDlgProc  : Get the size of each user picked pt
IgnorAlgDlgProc : Get the number of new centers to locate in the case of
                  of OPTIMAL (NO COMPETITION) menu option
UsrCmpDlgProc   : Get the option of file type of competitor file to be
                  opened
LocHelpDlgProc  : Display help messages
```

```
**********************************************************************
Module Definition File:

  LOC.DEF

**********************************************************************
Make File:

  LOC.MAK

**********************************************************************
Resource Script File:

  LOC.RC

**********************************************************************
Dialog Definition File:

  LOCDLG.DLG

**********************************************************************
ICON Definition File:

  LOC.ICO

**********************************************************************
Help Files:

(1) HLPINIT.HLP -- about getting help
(2) HLPG.HLP -- about general procedure
(3) HLPO.HLP -- about opening demand file
(4) HLPP.HLP -- about printing maps and reports
(5) HLPR.HLP -- about refreshing window
(6) HLPX.HLP -- about exiting Loc
(7) HLPD.HLP -- about distance calculation methods
(8) HLPM1.HLP -- about optimal (no competition)
(9) HLPM2.HLP -- about user input (no competition)
(10) HLPM3.HLP -- about user input (competition)
(11) HLPRPT.HLP -- about viewing reports
(12) HLPS.HLP -- about saving ".loc" file

**********************************************************************
```