THESIS

EFFICIENT ALGORITHMS FOR CAPACITATED MULTI-FACILITY LOCATION PROBLEMS

CHANSIRI SINGHTAUN

GRADUATE SCHOOL, KASETSART UNIVERSITY

2007

THESIS APPROVAL

GRADUATE SCHOOL, KASETSART UNIVERSITY

Doctor of Engineering (Industrial Engineering)
DEGREE

Industrial Engineering
FIELD

Industrial Engineering
DEPARTMENT

TITLE: Efficient Algorithms for Capacitated Multi-facility Location Problems

NAME: Miss Chansiri Singhtaun

THIS THESIS HAS BEEN ACCEPTED BY

THESIS ADVISOR
(Associate Professor Peerayuth Charnsethikul, Ph.D.)

COMMITTEE MEMBER
(Associate Professor Anan Mungwattana, Ph.D.)

COMMITTEE MEMBER
(Mr. Pornthep Anussornnitisarn, Ph.D.)

COMMITTEE MEMBER
(Assistant Professor Saowanee Lertworasirikul, Ph.D.)

COMMITTEE MEMBER
(Associate Professor Vira Chankong, Ph.D.)

DEPARTMENT HEAD
(Associate Professor Anan Mungwattana, Ph.D.)

APPROVED BY THE GRADUATE SCHOOL ON _____

DEAN
(Associate Professor Vinai Artkongharn, M.A.)

THESIS

EFFICIENT ALGORITHMS FOR CAPACITATED MULTI-FACILITY LOCATION PROBLEMS

CHANSIRI SINGHTAUN

A Thesis Submitted in Partial Fulfillment of
the Requirements for the Degree of
Doctor of Engineering (Industrial Engineering)
Graduate School, Kasetsart University
2007

Chansiri Singhtaun 2007: Efficient Algorithms for Capacitated Multi-facility Location Problems. Doctor of Engineering (Industrial Engineering), Major Field: Industrial Engineering, Department of Industrial Engineering. Thesis Advisor: Associate Professor Peerayuth Charnsethikul, Ph.D. 107 pages.

This research is created to find the solutions to squared- Euclidean distance capacitated multi-facility location problems where the costs are directly proportional to distances and the amount shipped. The algorithms for both specific and general cases of this problem are developed.

For the specific case, the problem with balanced transportation constraints is shown to be equivalent to a minimizing concave quadratic integer programming problem subject to transportation constraints. The extreme point ranking based method, which utilizes a special structure of constraints, is developed. It can be ensured to provide the optimal solution by comparing with a linearization method.

For the general case, the problem with unbalanced transportation constraints is shown to be a sum-of- ratio problem, which if each denominator functions is fixed at a certain value; the problem will be equivalent to the specific problem. The proposed extreme point ranking based method can be combined with explicit branching on the denominator functions as an empirical exact algorithm. Owing to the large computational time, a branch-and-bound based heuristics algorithm is created. Each branch uses a rectangular partition with lower bound and upper bound computed by utilizing the extreme point ranking based method and the proposed multistart trust-region method sequentially. The results show that with much less time the proposed heuristic algorithm can provide the solution within $0.62 \pm 0.42\%$ of absolute error with 99% confidence comparing with the empirical exact algorithm.

_____ _____ ___ / ___ / ___

Student's signature Thesis Advisor's signature

ACKNOWLEDGEMENTS

The completion of this research could not have been possible without the kind assistance and cooperation of a number of people.

My most profound gratitude goes to Assoc. Prof. Peerayuth Charnsethikul, my thesis advisor. He not only provided me immense academic supports, but he also gave me the valuable suggestions and comments on my research. I also wish to express my sincere appreciation to Assoc. Prof. Vira Chankong, Assoc. Prof. Anan Mungwattana, Dr. Pornthep Anussornnitisarn, and Assist. Prof. Saowanee Lertworasirikul, my committee members, for their academic suggestions and comments. My deep appreciation also goes to Assoc. Prof. Chatdanai Jiradecha, a graduate school representative, for his arrangement for my qualifying examination and final examination.

I also sincerely thank Mr. Suriya Natsupakpong, who enlightened me to write MATLAB program, for his encouragement and his useful comments and suggestions on my program.

Finally, my great appreciation is devoted to my beloved mom for her endless love and everlasting supports. I love you, mom.

Chansiri Singhtaun

September 2007

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

EFFICIENT ALGORITHMS FOR CAPACITATED MULTI-FACILITY LOCATION PROBLEMS

INTRODUCTION

Facility location problem, searching for the optimal locations of facilities to attain minimum cost or maximum profit, is a significant problem that almost every organization has to face with. A manufacturing organization needs to find the appropriate locations of new machines, warehouses, distribution centers, sub-assembly lines, or even new plants, and so on to distribute or supply parts (or products) to assembly lines (or customers) more efficiently and effectively with the lower transportation cost so that the company will gain higher productivity and make more profit. A service organization desires to find the locations of service centers, retail stores, outlets, and so on to be more easily reached by customers so as to capture more customers and then enhance its sells and profit. For a public section, the fire stations, clinics, schools, and so on should be appropriately located to provide quality service to the public efficiently and effectively. Conversely, if the facilities are not located at appropriate locations, the company will not only suffer from large investment but also suffer from the low manufacturing and service performance for a long time. Thus, it is undeniable that the excellent decision on facility location problem is indispensable for all types of organization to enhance the core competence of the company.

As a consequence, there are many researchers attempting to apply mathematical methods to provide the best or at least good solutions, which may be subsequently modified based on qualitative consideration to make a better decision. While the existing methods and algorithms have been continuously improved in order to increase quality of solutions for existing problems, new mathematical models representing more realistic problems have been constantly created. These mathematical models range from finding one facility location regardless of fixed cost and capacity restriction under certainty to finding the set of facility locations

considering fixed cost and capacity restriction under uncertainty. Most model said above excepting single facility location problem is proven to be NP-hard (Sherali and Tuncbilek, 1992) that is hard or even impossible to solve by the exact algorithms when the problem is large, which always occur in real-world case.

Multifacility location-allocation problem (MLP), which requires locating a set of facilities and simultaneously allocating to these facilities demands for service from a set of customers in order to optimize some performance criteria (Brimberg *et al.*, 1998), is a specific class of facility location problem that certainly cannot avoid this difficulty. Unsurprisingly, most of the developed algorithms for MLP are heuristic algorithms, which although cannot provide the best solution, it gives good solutions with much less computational effort than exact algorithms. Most of these heuristic algorithms devote to uncapacitated multifacility location-allocation problem (UMLP), which each facility can serve its customer with unlimited capacity, such as the well-known iterative location-allocation algorithm of Cooper (1964), the H heuristics of Love and Juel (1982), Cooper-NB method of Levin and Israel (2004) and so on. Few of them are developed for capacitated version of multifacility location-allocation problem (CMLP). Since CMLP has been studied in wild diverse versions such as the p-median problem that considers the CMLP on network and products are inseparable, planar CMLP with separable products, and so on, then the heuristic algorithms for CMLP have a wild variety depending on the problem definition.

This thesis proposes the efficient algorithms for a new class of CMLP. It will be hereafter called Capacitated Multi-facility Location Clustering Problem (CMLCP). This problem is to find the locations of m facilities on continuous plane and the allocations of n customers demanding for a certain amount of inseparable products to each facility with respect to its capacity limitation so as to minimize the total transportation squared-Euclidean distance or the corresponding nonlinear-function cost between facilities and customers. The applications of CMLCP are usually found in computer network or electrical system setup. For example, a company may need to find the appropriate locations for their computer servers and assignment of clients to those servers to minimize loss due to distance between servers and clients.

OBJECTIVES

The objective of this research is to develop the efficient heuristic algorithms for CMLCP under the following research scope.

1. A plane is two dimensional plane and considered to be a valid approximation of the region defining the problems
2. The data defining the problems are deterministic
3. No interaction between facilities is considered.
4. Any point on the plane is permissible as a location for any facility (there are no forbidden areas).
5. The number of new non-redundant facilities is fixed, and subsequently the fixed cost associated with the installation of a facility is ignored.
6. The squared-Euclidean metric is used to measure distances.

To evaluate the efficiency and effectiveness of these algorithms, they will be applied with the randomly generated numerical problems. And then, the results will be analyzed by comparing with the exact or empirical exact algorithms.

LITERATURE REVIEW

This section reviews the related literatures for developing algorithms in this research. The topics of review are arranged according to the order of the methods required in each step of the proposed algorithm for the general case of studied problem. So, first of all the algorithm will be roughly explained as follows. The algorithm proposed in this thesis starts with reducing the location variables of the original mathematical model to allocation variables. Therefore, the problem remains only $m \times n$ allocation variables and becomes sum-of-nonlinear ratio problem, which is a class of fractional programming problem. Then, the algorithm turns into an iterative procedure of fixing the denominators, which transforms the problem to quadratic integer programming (QIP), and solving the corresponding QIP sequentially. Therefore, the review topics are divided into three parts as follows. The first part summarizes the general concepts of facility location problem and the mathematical methods having been proposed so far. The second part summarizes methods for the fractional programming problem. And the last part explores the methods for QIP.

Facility Location Problems

Facility location problem has been continuously studied by many researchers along past four decades because of its benefits. It is undeniable that the appropriate locations of facilities such as machines, warehouses, sub-assembly lines, and so on bring many advantages such as transportation cost reduction, productivity improvement, and enhancement of efficiency and effectiveness of parts or products supply to manufacturing organization. Many service organizations concede sincerely that the appropriate locations of their service centers, retail stores, outlets, and so on can improve efficiency of services and customer capture, and then finally make them more profitable. Also, the public sector can provide good services to public thoroughly because of the appropriate locations of the service centers, clinics, schools, and so on.

Many mathematical methods for finding the best or at least good locations for the facilities have been developed so as to provide better alternatives to be then combined with intuitive decision in the final stage. These mathematical methods have been consistently improved to solve the more complex facility location problems reflecting real-world situations with less computational effort. Up to now facility location problems have very wide ranges. They vary from finding one facility location regardless of fixed cost or locating cost and capacity restriction under certainty to finding the set of facility locations considering fixed cost and capacity restriction under uncertainty. Most of them excepting single facility location problems are NP-hard where the exact algorithms are restricted at only a small size of such problems, and then there are the considerable interests in developing the heuristic algorithms for such problems.

Multifacility location-allocation problem, which is a class of facility location problem, has been widely studied for a long time. It requires locating a set of facilities and simultaneously allocating to these facilities demands for service from a set of customers in order to optimize some performance criteria (Brimberg *et al.*, 1998). The problem can be classified further by capacity limitation of each facility into two categories: uncapacitated and capacitated multifacility location problem (hereafter called UMLP and CMLP respectively). For UMLP, each facility can serve customers as much as its customer wants without limitation. Conversely, for CMLP each facility can serve its customers as much as it has. The algorithms to solve each of them can be summarized as follows

Algorithms for UMLP

UMLP is locating certain facilities so as to serve optimally a given set of customers, whose locations and demands are known. It requires determining the locations of the facilities and assigning customers to facilities so as to minimize the sum of weighted distances from facilities to assigned points (Levin and Israel, 2004). The general mathematical model can be formulated as follows (Houck *et al.*, 1997).

$$\textit{Minimize} \quad \sum_{i=1}^{m}\sum_{j=1}^{n} z_{ij} w_j d(X_i, P_j) \tag{1}$$

$$\textit{subject to} \quad \sum_{i=1}^{m} z_{ij} = 1 \qquad ; \; j = 1,...,n$$

$$\textit{,where} \quad z_{ij} = \begin{cases} 1, \text{if customer } j \text{ is assigned to facility } i \\ 0, \text{otherwise} \end{cases}$$

w_j is demand of customer j ; $j = 1,...,n$

$d(X_i, P_j)$ is distance between facility i and customer j on continuous plane

X_i = unknown location of facility i at co-ordinate (x_i, y_i)

P_j = known location of customer j at co-ordinate (a_j, b_j)

The objective function above gives the total transportation cost, while the constraints set ensures that all customer demands are satisfied.

Because of NP-hardness, exact algorithms for the location-allocation problem have been restricted until very recently to relatively small instances. There are few exact algorithms for this problem as follows. Kuenne and Soland (1972) created a branch-and-bound algorithm, MULTIWEB, which allows the solution of problem sizes of 25 customers and 1 to 5 facilities. The set reduction method for the multisource Weber problem with rectangular distance developed by Love and Morris (1975) is also restricted to a similar-sized problem. Later, Drezner (1984) suggested an efficient procedure, separation line, developed for the special case of Euclidean distance facility location-allocation problem. This method uses a straight line to separate customer locations into two sets allocated from two facilities. Because there are $O\left(n^2\right)$ single facilities location problems to solve, computational time will increase rapidly when n grows. Subsequently, Chen *et al.* (1998) developed D-C (Differences of Convex Functions) programming which is a recent technique of global optimization to obtain a near-linear growth in the computational time as n increases. This approach is restricted to convex functions and thus it has a limited scope of application to real-world problems. Later, there are many algorithms, which are the extension versions of the above four algorithms, developed by combining the existing algorithms with new techniques. For example, the separation line method was

extended by Rosing (1992). This method separates set of locations iteratively by straight lines *n*-1 times in order to solve problem related to more than two facilities. The branch-and-bound algorithm and global optimization were combined with new tools, column generation, by Krau in 1997 (Brimberg *et al.*, 2000). It made drastically augmented the size of problems be solved exactly, but the restriction of computational effort still adheres in these exact algorithms, which impedes the application for the large-scale problems. To reduce high computational time, heuristic algorithms are required. Many such algorithms having been proposed so far can be classified into two following approaches: decomposition approach and enumerative approach.

1. Decomposition Approach

This approach uses the property that the location and allocation phase of the problem are very easy to solve in isolation. Given the facility locations, each customer is simply allocated to its nearest facility. Alternatively, knowing the allocation of the customers among the facilities, the problem reduces to the solution of independent single facility minisum problems.

There are many algorithms under this approach. The earliest algorithm is the well-known iterative allocation-location algorithm (ALA) for Euclidean distance created by Cooper (1964). Starting with an initial partition of the customer set, the Cooper's algorithm alternately solves between the location and allocation phase. In each of the iterations, ALA produces a lower value of the objective function. This iterative procedure will proceed until the process becomes trapped at a local minimum or no further improvement on objective function value can be found. Afterwards, Cooper's algorithm has been extended in wide diverse versions. The most recent one was proposed by Levin and Israel (2004). The proposed multistart version involves repeating ALA many times from randomly generated initial solutions and retaining the best local minimum from the trials as the final solution. In addition, they developed their own procedure for location phase, Newton-Bracketing procedure, to use in place of Weiszfeld procedure in Cooper's algorithm. The earlier interesting extensions are as follows. Sullivan and Peters (1980) improved allocation phase of

Cooper's algorithm by proposing an efficient method to cluster customers into mutually exclusive subsets, in each of which a facility is then located. Gamal and Salhi (2003) improved Cooper's algorithm under the same concept as Sullivan and Peters (1980)'s algorithm, but in the different way. They proposed the two-phase algorithm, which phase I uses Cooper's algorithm to generate several configurations while phase II records information of those locations to construct new locations, and uses the solution of phase II to cut continuous space to smaller space like a cell. The searching will stop when the specified size of cell of the location is reached. Moreno *et al.* (1990) constructed a "drop" heuristic that begins with an initial solution of N clusters, where N is chosen between m and $2m$. Then surplus facilities are dropped in a greedy manner until exactly m facilities are left. This method was tested on problem sizes of up to 900 customers and 10 facilities and obtained results comparable to the Cooper's algorithm.

Love and Juel (1982) developed a heuristic method with a defined neighborhood structure. This neighborhood consists of all the points around a current solution, which are obtained by exchanging a specified number of assignments of customers from their current facilities to new ones. Five variants of the proposed method were investigated. The first three algorithms, denoted as H1, H2, and H3, use a single exchange, while the last two, H4 and H5, allow up to two exchanges. Different strategies such as first improvement and best improvement are employed to make descent moves from the current solution to a neighborhood point. Again, because the search is local, the H heuristics of Love and Juel (1982) are guaranteed only to obtain a local minimum. The motivation for the larger neighborhood of H4 and H5 is to better enable the algorithm to jump out of a "local optimum trap," but obviously, this comes at a large cost in computational time.

Mladenovic and Brimberg (1995) tested a hybrid algorithm that takes random points in a k-exchange neighborhood of the type used in the H-heuristics of Love and Juel (1982), and then applies Cooper's algorithm at each of these points. In Brimberg and Mladenovic (1996a), elementary tabu search rules are added to the H3 heuristic to allow ascent moves away from a local optimum. A variable neighborhood

concept, introduced by Brimberg and Mladenovic (1996b), systematically increases the number of exchanges (k) of the H-type neighborhood to expand the search radius toward a local optimum. Hansen *et al.* (1998) proposed the algorithm that gives an approximate solution by solving a related p-median problem followed by the solution of single facility Weber problems. Later, Garcia *et al.* (2003) suggested scatter search, which is a population based meta-heuristic. This algorithm uses a reference set to combine its solutions and construct new solution to solve the problem. It begins with selecting a set of m points for the facilities, which is a subset of the reference set. The selection of the m locations for the facilities will be evaluated and then the solution will be used to update the reference set. The improvement of the solution may occur from the combination procedure trying to add the good characteristics of the selected solution to get new current solution and local search procedure that improves the basic move.

A completely different heuristic approach was given by Chen (1983). Using an approximation proposed in Charalambous and Bandler (1976), the objective function is transformed by giving an exponent ($-N$) to all distances between customers and facilities and an exponent ($-1/N$) to the sum of modified distances from all facilities of each user. For sufficiently large N, this last quantity approaches the distance between the customer and its closest facility. In this way, the allocation decision variables are eliminated. The resulting problem is then solved by the Broyden-Fletcher-Shanno quasi-Newton method. Good results, but not always the best known, are obtained with N set at 100.

2. Enumerative Approach

Algorithms under this approach are to solve location and allocation simultaneously, which is totally opposite to the decomposition approaches mentioned above. There are much fewer algorithms under this approach because of intensive mathematical knowledge requirements. Murtagh and Niwattisyawong (1982) proposed a heuristic that uses MINOS, a large-scale nonlinear programming package, to solve simultaneously for both the locations and allocations. As the iterations

proceed, the algorithm fixes any allocation decision variables (z_{ij}) that reach either a value of 0 or w_j, and then updates only the free variables. The update uses a quasi-Newton approximation of the Hessian matrix within the space of the free variables at nondifferentiable point, which is a subgradient suggested by Kuhn (1973). More recently, Bongartz *et al.* (1994) developed a projection method for solving the multisource Weber problem. Instead of assuming Euclidean distances, they considered the more general l_p norm. Like Murtagh and Niwattisyawong (1982), the new method solves simultaneously for location and allocation decision variables. Simple projection formulas on subspaces of the domain are derived (instead of solving the system of equations in general) and used to find descent directions. The algorithm is guaranteed to converge to a local minimum.

Brimberg *et al.* (1998) tested a multistart version of their algorithm, where the initial solutions may be generated randomly or by partitioning customers in successive sets along a traveling salesman tour. The solutions were compared with multistart version of Cooper's, Murtagh and Niwattisyawong (1982)'s, and Chen (1983)'s algorithm. The projection method generally outperforms the other heuristics, but in several of the reported test problems Cooper's algorithm comes in a close second.

All algorithms to solve problem (1) mentioned above are usually developed for UMLP where distance function is measured in Euclidean or rectilinear metric. The study of problem (1), where distance function is measured in squared Euclidean metric, has not been discovered from reviewing literatures while only one variant version of problem (1) with capacity constraints is found and it will be mentioned in the next section.

Algorithms for CMLP

CMLP is locating certain facilities so as to serve optimally a given set of customers, whose locations and requirements are known. It is required to determine the locations of the facilities and assign customers to facilities with respect to capacity

restriction of each facility so as to minimize the sum of weighted distances from facilities to assigned points. There are wild varieties of CMLP studied so far depending on problem definitions. It can be mainly separated into three categories: classical CMLP, p-median clustering problem, and planar CMLP. The first two categories are done on network which has the determined vertices to be locations of the facilities with certain capacity. The difference between them is that one is used for separable products and the other is used for inseparable products. The last category is done on continuous plane and for separable products. The algorithms for these problems can be summarized as follows.

1. Classical CMLP

This problem is finding the optimal locations of *m* facilities to be open at the determined vertices. Each location can be assigned to the facility with a certain capacity. Each of the *n* customers' demand can be served from many facilities separately. Generally this problem considers fixed cost and has the corresponding mathematical model as follows (Nauss, 1978).

$$\textit{Minimize} \quad \sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij} z_{ij} + \sum_{i=1}^{m} f_i y_i \tag{2}$$

$$\textit{subject to} \quad \sum_{j=1}^{n} z_{ij} \le s_i y_i \quad ; \; i = 1,\ldots,m$$

$$\sum_{i=1}^{m} z_{ij} = w_j \quad ; \; j = 1,\ldots,n$$

, *where*

d_{ij} is distance between vertex i and vertex j

s_i is capacity of facility i

w_j is demand of customer j

z_{ij} is a positive fraction of products supplied from facility i to customer j

f_i is the fixed cost of facility i

$$y_i = \begin{cases} 1, \text{if facility at vertex } i \text{ is open} \\ 0, \text{otherwise} \end{cases}$$

The algorithms for problem (2) are the algorithms for attacking linear integer programming. Branch-and-bound algorithm is an exact algorithm for such problem. Most of the algorithms developed later come from improving the bound tightening and branching strategy. For example, Sa (1969) proposed the branch-and-bound algorithm with continuous relaxation, which replaces $y_i \in \{0,1\}$ by $0 \le y_i \le 1$. He observed that it is possible to substitute and remove y_i variables from the problem. The resulting relaxation can be formulated as a transportation problem, which may be solved using a network algorithm and then allows for significant savings in computational time. Later, Akino and Khumawala (1977) developed an efficient improved branch-and-bound algorithm based on this procedure. Their algorithm has dramatically reduced computational time as compared to existing algorithms. Successively, Nauss (1978) used this efficient branch-and-bound algorithm with specialized Lagrangean relaxation on the second constraint. The results showed that the computational time can be matched competitively with Akino and Khumawala's algorithm.

2. P-median Clustering Problem

This problem is similar to classical CMLP excepting that the demand of customer cannot be separated. Each customer has to be served by only one facility. The clustering problem is to partition a given set of objects with *m* known attributes to clusters of similar objects, so that objects in different clusters are dissimilar. P-median is identified as a special clustering problem if the sets of customers served by the same facility are considered as clusters. This problem can be mathematically formulated as follows (Lorena and Senne, 2003).

$$\textit{Minimize} \quad \sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij} z_{ij} \tag{3}$$

$$\textit{subject to} \quad \sum_{j=1}^{n} w_j z_{ij} \le s_i y_i \quad ; i = 1,..m$$

$$\sum_{i=1}^{m} z_{ij} = 1 \qquad ; j = 1,\ldots,n$$

$$\sum_{i=1}^{m} y_i = m$$

$$, \textit{ where} \qquad d_{ij} \text{ is distance between vertex } i \text{ and vertex } j$$
$$s_i \text{ is capacity of facility } i$$
$$w_j \text{ is demand of customer } j$$
$$y_i = \begin{cases} 1, \text{if facility at vertex } i \text{ open} \\ 0, \text{otherwise} \end{cases}$$
$$z_{ij} = \begin{cases} 1, \text{if customer } j \text{ is assigned to facility } i \\ 0, \text{otherwise} \end{cases}$$

Since this problem is also a linear integer programming, then again the algorithm is based on the same concept as the algorithms mentioned in the last section. The difference between algorithms for problem (3) and problem (2) may come from using the special properties of the problems. Some of the algorithms for problem (3) are as follows. Mulvey and Beck (1984) used Lagrangean relaxation of assignment constraints in a 0-1 linear integer programming problem. A primal assignment heuristic is embedded within a subgradient method improved by interchanging medians in clusters. Koskosidis and Powell (1992) improved Mulvey and Beck's results suggesting various algorithms to find initial solutions for knapsack problems (Lagrangean subproblems). While, recent approaches applied meta-heuristics, such as simulated annealing, tabu search, and genetic algorithm. Lorena and Senne (2003) explored improvements on upper bounds, the Lagrangean/surrogate primal counterpart. The Lagrangean/surrogate relaxation is combined with location-allocation heuristics proposed by Cooper (1964). The heuristics improved solutions and maintained feasibility by the Lagrangean/surrogate optimization process, swapping medians and vertices inside the clusters, reallocating vertices, and iterating until no more improvement occurs.

3. Planar CMLP

This problem considers the locations of facilities and customers on plane. The optimal location of each facility can be found by solving single facility problem, if we know who the customers in the cluster or facility are. The transportation constraints and the products, which are separable, are used to make the problem easier

but still realistic. It can be formulated mathematically as follows (Sherali and Tuncbilek, 1992).

$$\text{Minimize} \quad \sum_{i=1}^{m}\sum_{j=1}^{n} z_{ij}\, d(X_i, P_j) \tag{4}$$

$$\text{subject to} \quad \sum_{i=1}^{m} z_{ij} = w_j \quad ; \; j = 1,\ldots,n$$

$$\sum_{j=1}^{n} z_{ij} = s_i \quad ; \; i = 1,\ldots,m$$

$$z_{ij} \geq 0$$

where z_{ij} is demand of customer j supplied from facility i

s_i is capacity of facility i

w_j is demand of customer j

$d(X_i, P_j)$ is distance between facility i and customer j on plane

X_i = unknown location of facility i at co-ordinate (x_i, y_i)

P_j = known location of customer j at co-ordinate (a_j, b_j)

The algorithms for such problem generally replace location variables in terms of allocation variables obtained from solutions for single facility location problem to reduce the variables. And then, the special properties of each distance function combining with transportation constraints are used to update bound in a branch-and-bound algorithm. For example, Sherali and Tuncbilek (1992) solved this problem with $d(X_i, P_j)$ = squared-Euclidean distance by fixing z_{ij} in the objective function and reformulating the problem in the space of z_{ij} variables resulting in a convex maximization problem. Then, they presented upper and lower bound function for the problem derived from Reformulation Linearization Technique (RLT) and used a branch-and-bound enumeration algorithm. Their solution methods are able to solve problems of sizes ranging from (n, m) = (120, 6) to (60, 20) within 2% of optimality.

The problem considered in this thesis falls between problem (3) and problem (4). In this research, planar CMLP with inseparable demand is considered. Because it is planar CMLP the algorithm for the single facility location problem can be used to reduce location variable as mentioned in section 3. However, without the balanced

transportation constraint denominators cannot be fixed at a certain constant as done in Sherali and Tuncbilek (1992). Therefore, the problem becomes sum-of-ratios, which is a class of fractional program. The algorithms for fractional program are studied and summarized as follows.

Fractional Programming Problems

Fractional programming problem is a specific class of nonlinear programming problems, where the objective function is in ratios of two functions. Some objective functions of fractional programming are to minimize a ratio of two functions while that of others involve more than one ratio of two functions. Fractional programming problem arises from economy problem. The general ratios, which are maximized, are profit/capital, profit/cost, return/risk, and so on. One of the earliest fractional programming problems is an equilibrium model for an expanding economy introduced by von Neumann in 1937. The model determined the growth rate of an economy as the maximum of the smallest of several output-input ratios. Later, problems in various fields, such as layered manufacturing, multistage stochastic shipping problem, minimizing of the mean response time in queuing–location problem, and so on, also produce wide diverse versions of the fractional programming problem. At this time, the fractional programming problem can be mainly classified to four classes as follows (Frenk and Schaible, 2004).

Single-Ratio Fractional Program

The objective of this problem is to find minimum (or maximum) of a ratio of two functions. One form of this problem is as follows.

$$\inf_{x \in S} \frac{f(x)}{g(x)} \tag{5}$$

, where $f, g : R^n \to [-\infty, \infty], 1 \le i \le m$ be extended real-valued functions, which are finite-valued on convex set $S = \{h_k(x) \le 0 \ ; k = 1, ..., l\}$ with $g(x) > 0$ for every $x \in S$.

If $f, g,$ and $h_k; k = 1,...,l$ are affine functions (linear plus a constant) then the problem is called a single-ratio linear fractional program. Moreover, problem (5) is called a single-ratio nonlinear fractional program if at least one of these three functions ($f, g,$ and $h_k; k = 1,...,l$) are nonlinear.

In 1962 Charnes and Cooper published their classical paper in which they showed that a single-ratio linear fractional program can be reduced to a linear program using a nonlinear variable transformation. Later, Martos (1964) showed that it can be solved with an adjacent vertex-following procedure as same as linear programs using the simplex method. He also showed that generalized convexity properties (pseudo-linearity) of linear ratios enable such an extension of linear programming techniques.

Although the theory set forth in Charnes and Coopers (1962) and Martos (1964)'s paper now makes it possible to use any linear programming techniques to solve linear fractional programming, there continues to be considerable interest in methods that are specially designed for solving this particular class of problems as well as in further developments which can provide insight into the general nature and possibilities of parametization methods.

The parametric approach for single-ratio linear fractional program was first proposed by Isbell and Marlow in 1956. It was extended and applied for single-ratio nonlinear (quadratic) fractional programming where f_i, g_i are quadratic functions by Ritter in 1967. Later, Dinkelbach (1967) proposed general methods for single-ratio fractional program. This method considers the problem as the global optimization problem

$$\max_{x \in S} f(x) - \lambda g(x) \tag{6}$$

, where $\lambda \in R$ is a constant. Using this method, a sequence of values λ that converges to the global optimum function value is generated. This method has since then been

applied to many specific types of single-ratio fractional programs such as maximizing single-ratio whose *f* and *g* are both quadratic (single-ratio quadratic function) or *f* is nonnegative concave and *g* is convex (single-ratio concave-convex fractional program), and so on. Later, Phillips (1998) reported that although Dinkelbach's method worked in many types of single fractional program, it did not work well for some fractional programs which maximize the ratio of two concave or two convex functions, or the ratio of a convex to a concave function. He also reported that this method does not provide a sequence of improving upper bound and hence even though the sequence λ may converge to global optimum function value, no bound on the error is available at any iteration. He also proposed the algorithm that improves Dinkelbach's algorithm by providing a means for obtaining a sequence of improving upper bounds which, along with the corresponding sequence of improving lower bounds, will provide a bound on the error at each iteration of the solution procedure. In addition, both the sequence of lower bounds and the sequence of upper bounds converge to the global optimum function value at a "super linear" rate. This algorithm is also appropriate for the class of quadratic fractional programs where the ratio may involve concave, convex, or even indefinite terms.

Another different method, which was proposed by Pardalos (1986), is to replace the nonlinear functions by suitable linear under estimators and then obtain the global optimum by a vertex ranking procedure. This method is applicable only when $f(x)$ is a convex quadratic function and $g(x)$ is linear (hence the ratio is quasiconvex).

Generalized Fractional Program

In some applications more than one ratio appears in the objective function. One form of such an optimization problem is the nonlinear programming problem

$$\inf_{x \in S} \sup_{1 \le i \le m} \frac{f_i(x)}{g_i(x)} \tag{7}$$

with extended real-valued functions $f_i, g_i : R^n \to [-\infty, \infty], 1 \le i \le m$, which are finite-valued on S with $g_i(x) > 0$ for every $1 \le i \le m$ and $x \in S$. It can also be solved by parametizing approach (Frenk and Schaible, 2004). Owing to large computational effort of algorithms under this approach, Birbil *et al.* (2005) proposed an approximation approach that can be controlled by a predetermined parameter. Their proposed algorithm is promising to find an effective near-optimal solution in an efficient manner.

Sum-of-Ratios Fractional Program

This problem optimizes a sum of multiple ratios of functions over a convex set *S*. It arises naturally in decision making when several rates are to be optimized simultaneously and a compromise, which optimizes a weighted sum of these rates, is sought. It can be mathematically represented by

$$\inf_{x \in S} \sum_{i=1}^{m} \frac{f_i(x)}{g_i(x)} \tag{8}$$

with $g_i(x) > 0$ for every $1 \le i \le m$ and $x \in S$. There are many situations arising this kind of problems such as analyzing of a multi-stage stochastic shipping problem, analyzing of the optimal partitioning of a given set of entities into a number of mutually exclusive and exhaustive clusters to minimize the sum of the squared distance within groups, minimizing the mean response time in queuing-location problems, a bond portfolio optimization, a hospital fee optimization problem used by hospital administrator in the state of Texas to decide on relative increases of charges for different medical procedures in various departments, and so on (Frenk and Schaible, 2004).

Even in the simplest case where the ratios are all linear, their sum is neither quasiconvex nor quasiconcave, though each of them has both properties. As a result, the problem has multiple local maxima (minima), many of which fail to be globally

optimal. Since the difficulty of the problem strongly depends on the number of ratios, some algorithms proposed so far assume it to be a few and solve this multi-extremal optimization problem by exploiting the low-rank nonconcavity. When number of ratios is not limited, branch-and-bound algorithms has to be used at this time to solve the problem within a practical amount of time (Kuno, 2002).

Among the algorithms for maximizing sum of linear ratios, the outperform branch-and-bound algorithms were proposed by Kuno (2002) and Benson (2002), both of which use concave envelopes of ratios on quadrangles to compute upper bounds on the optimal value in the bounding process. In the case of convex functions, it is rather easy to compute tight upper bounds on rectangles or simplices generated by subdividing the feasible set in the branching process. However, the sum of ratios is not convex, as mentioned above. Thus Kuno (2002) subdivided the projection of the feasible set on each denominator-numerator space into trapezoids and defined a concave envelope over each of them using two affine functions. Benson (2002) showed that the similar concave envelopes can be defined on a rectangle, as in the usual rectangular branch-and-bound algorithms. Other than branch-and-bound approach, an interesting approach is the image space analysis proposed by Falk and Palocsay (1994). They associated a new variable with each ratio and defined an "image space", in which optimization is easy along the coordinate axes. Kuno (2005) proposed the corrected trapezoidal branch-and-bound algorithm that was proposed in Kuno (2002) and also added an inexpensive procedure for tightening the upper bound (for maximizing problem) of the trapezoidal algorithm significantly, which makes it faster to obtain final solution. At this time, this method becomes one of two methods (another is Konno and Fukaishi's algorithm) that can solve the problem up to ten ratios, which is the largest size of this problem at this time. Although Kuno's algorithm is faster than Konno and Fukaishi's algorithm, it provides inferior solutions (Schaible and Shi, 2003).

For sum of nonlinear ratios problem, the earliest paper was proposed by Freund and Jarre (1999). They considered the following problem.

$$\min_{x \in S} \quad p(x) + \sum_{i=1}^{m} \frac{f_i(x)}{g_i(x)} \tag{9}$$

They considered this problem as global optimization problem (10) shown below.

$$x(r) = \arg\min \left\{ p(x) + \sum_{i=1}^{m} \frac{f_i(x)}{r_i} \middle| x \in S(r) \right\}, \tag{10}$$

$$, where \;\; S(r) := \left\{ x \in S \middle| g_i(x) \geq r_i \text{ for all } i = 1, 2, ..., m \right\}$$

$$and \quad q(r) := \left\{ p(x(r)) + \sum_{i=1}^{m} \frac{f_i(x(r))}{r_i} \right\}$$

And, they considered the problem under the following assumption: $S \subset R^n$ is a compact convex set such that $f_i(x) \geq 0$ and $g_i(x) > 0$ for all $i = 1, 2, ..., m$ and all $x \in S$. For minimization problem (9), the functions $p(x)$ and $f_1, f_2, ..., f_m$ are convex and the functions $g_1, g_2, ..., g_m$ are concave. The functions $g_1, g_2, ..., g_m$ are treated separately. They showed how the problem can be reduced to the minimization of a function of *m* variables, where the function values are given by the solution of certain convex subproblems. Based on this reduction, they proposed an algorithm for computing the global minimum of problem (10) by means of an interior-point method for convex programs.

Multi-Objective Fractional Program

It is the most recent and challenging class of fractional programming. It relates to the second and third class, which can be formulated as following mathematical model (Frenk and Schaible, 2004).

$$\inf_{x \in S} \left(\frac{f_1(x)}{g_1(x)}, ..., \frac{f_m(x)}{g_m(x)} \right) \tag{11}$$

Most of the algorithms for this problem are generalized Dinkelbach transformation in sense of parametric approach. For an example, Tammer *et al.* (1999) proposed a different concept solution for generalizing Dinkelbach transformation. This algorithm is using vector optimization to provide both exact and approximate solution for the original problem.

The proposed algorithm for the studied problem, which falls in the third class of fractional program, will iteratively fix the denominator function and then solve nonconvex subproblems in the same way as usual algorithms for sum of nonlinear ratios problem. Unlike subproblem of existing sum of nonlinear ratios problem, the subproblem of the studied problem is not convex. It is a concave quadratic integer programming problem. Therefore, the special methods for this kind of problem are required. These methods can be summarized as follows.

Quadratic Integer Programming Problems

Quadratic integer programming problem (QIP) is a specific class of nonlinear programming with discrete variables in quadratic objective function and a linear set of constraints. It arises from a variety of very important problems in various applications such as the process industry, the financial, engineering, and so on (Zizong *et al.*, 2004). Most of the algorithms for QIP having been proposed so far devote to minimizing convex QIP problem. Few of them are developed for minimizing concave QIP problem, which can be represented by the following mathematical model (Gupta *et al.*, 1996).

$$\begin{aligned} &\textit{Minimize} \quad Q(x) = c^T x + x^T G x \\ &\textit{subject to} \quad Ax = b \\ &\qquad\qquad x \geq 0 \\ &\qquad\qquad x \in R^n, \text{ and integer vector} \end{aligned} \tag{12}$$

, where G is a symmetric negative (semi) definite matrix. These algorithms are under two approaches as follows.

Enumerative Approach

This approach attacks the nonlinear objective function directly with some kind of enumerative scheme. The classical algorithm under this approach is branch-and-bound algorithm. The idea of this algorithm is to solve the QIP problem directly in a manner exactly analogous to that so successfully employed in linear integer programming problem. Under this idea, the QIP problem is solved by relaxing integer variables and then using the methods for minimizing concave quadratic programming (QP) to construct bound in each branch. These methods can be classified into five main methods: extreme point ranking, cutting plane, convex envelopes, reduction to bilinear programming, and reduction to separable form method (Floudas and Visweswaran, 1995). They can be described as follows.

Extreme point ranking method uses the fact that all solutions (local or global) to concave QP lie at some vertices of the feasible region. The basic idea is to rank the vertices of the polytope defining the feasible region in order of importance regarding the global solutions. Starting from one of the vertices of the polytope, the nearby vertices are ranked using an extreme point approach. This provides a new vertex to move to, and the process continues until no adjacent vertices can be found with a decreasing objective function value. At each step, usually a linear programming problem (LP) is solved to provide a bound on the global optimum. Cabot and Francis (1970) proposed such an approach where a LP is solved (with a linear under estimator of the concave function being used for the objective function of the LP), utilizing the extreme point ranking approach proposed by Murty (1968).

In general, extreme point ranking method is not very attractive because in the worst case, this approach might degenerate to a complete inspection of all vertices. As a result, cutting plane method, which bases upon the existence of vertex solutions but explores some of these vertices, was developed by Tuy (1964). Tuy's basic approach starts from a specific vertex of the polytope. The edges of the polyhedron issuing from this vertex are used to define a cone that contains the feasible region. Tuy proposed the use of cuts to successively reduce this cone. These cuts are essentially

used to eliminate parts of the feasible region from further consideration. At each step, an auxiliary subproblem is solved in the subcone, which gives rise to a new vertex point, and possibly a better candidate for the global solution. A similar approach using cutting planes to reduce the feasible region was proposed by Ritter (1966). However, it was shown by Zwart (1973) that both of these proposed approaches can be nonconverge. He cited two examples to show that the methods either fail to converge due to cycling, or need an infinite sequence of cutting planes. The reason for the cycling behavior as well as nonconvergence of these approaches lies in the fact that although the approaches generate cones during the algorithm, they fail to explicitly incorporate these cones into the remaining steps. This is essential to avoid the reemergence of vertices that have already been considered. This difficulty with the approaches was recognized by Zwart (1974), who proposed a modified ε-convergent approach where, at each iteration, constraints are added to ensure that the solution of the corresponding LP at each iteration is contained in the cone with the vertex at the current local minimum and a perturbed set of extreme rays coincident at that vertex. Although several other variants of Tuy's algorithm have been proposed later, the difficulties to guarantee convergence associated with the Tuy-cut approaches cannot be eradicated.

Convex envelopes method was first developed by Kleibohm in 1967. The concept of this method is to construct linear underestimating function for the original concave function over different regions of feasible domain, and does not involve any cuts of the feasible region. The linear underestimating function was developed using the convex envelopes, which was later proven by Falk and Hoffman (1976). The algorithm converges finitely. However, a disadvantage of the approach is that in order to approximate the convex envelopes implicitly the size of the linear subproblems grows from one iteration to the next.

Other remaining methods use the concept of reduction methods, which try to relate (by reduction) the concave QP to other type QP and then solve the reduced problem using the efficient algorithm and transform the solution back to solution of the original problem. Reduction to bilinear programming method uses the fact, which

was first proven by Konno (1976), that the concave QP can be reduced to associate bilinear programming. Konno (1976) used this concept to propose a cutting plane algorithm for bilinear programming, which he then used to solve the concave QP. The resulting bilinear programming is greatly simplified because of the problem structure. And, the verification of necessary and sufficient conditions for the existence of an optimal solution is reduced to the solution of an LP.

For reduction to separable form method, Rosen and Pardalos (1986) proposed that it is possible to reduce the concave QP to separable quadratic form. This separable quadratic form is a mixed-integer programming used to approximate the original problem. Rosen and Pardalos (1986) utilized this approach to solve large-scale concave programs. The first step is to obtain a bound on the relative error of the approximation used (where the relative error is the difference in the approximation from the original objective function). In general, the relative error is guaranteed to be at most 0.25. If the approximate solution obtained is not satisfactory, then a single mixed 0-1 problem is solved. This problem is formulated by a single piecewise linear underestimators of the separable objective function. More recently, Phillips and Rosen (1988) proposed a parallel algorithm for large-scale concave QP. This algorithm also first reduces the concave objective function to a separable form. At each iteration, it combines a heuristic step that attempts to eliminate part of the feasible region, with a parallel implementation of the *2n* multiple cost row linear programs. The algorithm has a guaranteed convergence in a finite number of steps to an ε-approximate solution, and the number of subproblems that need to be solved is bounded by $(4n/\varepsilon)^{n/2}$.

Recently, Thoai (1998) established a finite branch-and-bound algorithm, in which the branching procedure is integral rectangular partition, and the bound estimation is performed by the new three methods: an outer approximation procedure, the use of convex envelopes for separable concave function, using linear 0-1 mixed-integer programming problem. He also compared the solution of using these bound estimation methods. Although the results showed that the first and the last bound estimation method give the same lower bound, which is better than the other, the

second method has a numerical advantage that it only requires solving ordinary linear programs.

Other than branch-and-bound algorithm stated above, there still exist another recent algorithm under enumerative approach. It uses the basic idea of extreme point solutions as same as that of extreme point ranking methods for QP but it uses the different movement mechanism to move to the next adjacent extreme point. This algorithm, which is called extreme point ranking for QIP problem, was proposed by Gupta *et al.* (1996). A linear integer programming (LIP) problem

$$Minimize_{x \in S} \ (c+U)x \tag{13}$$

$$, where \quad S = \left\{ x \in R^n : Ax = b, \quad x \geq 0 \ \text{ and an integer vector} \right\}$$

$$U_p = \text{the } p\text{-th component of } \ U \in R^n$$

$$= \ \min_{x \in S} x^T G_p \quad ; \ \text{for every } \ p = 1, ..., n$$

$$G_p = \text{the } p\text{-th column of matrix } G \text{ shown in equation (12)}$$

is constructed which provides bounds on the values of the objective function of QIP problem (12). The integer feasible solution of this related LIP problem are systematically scanned to rank the integer feasible solution of QIP problem in non-decreasing order of the objective function value. Method to scan and move to the next adjacent solution is not pivot operations as being used in extreme point ranking for QP but it is a special cutting plane developed by Verma *et al.* (1991). Gupta *et al.* (1996)'s algorithm can be applied not only for minimizing concave QIP problem with linear constraint sets but for nonlinear constraint sets as well.

Linearization Approach

This approach is to transform QIP problem to the equivalent mixed-integer programming problem and then solve the latter problem. The linearization technique is usually applied for 0-1 QIP problem, which is a specific class of QIP problem and can be formulated as follows.

$$\textit{Minimize} \quad \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} G_{ij} x_i x_j \tag{14}$$

$$\textit{subject to} \quad x \in S, \text{ and } x \text{ is binary}$$

$$\textit{, where} \quad S \equiv S_1 \cap S_2 \cap S_3 \text{ is a nonempty polyhedron with}$$

$$S_1 = \left\{ x \in R^n : \sum_{i=1}^{n} a_{ki} x_i = b_k \text{ for } k = 1, ..., K \right\},$$

$$S_2 = \left\{ x \in R^n : \sum_{i=1}^{n} H_{li} x_i \geq h_l \text{ for } l = 1, ..., L \right\}, \textit{and}$$

$$S_3 = \left\{ x \in R^n : 0 \leq x_i \leq 1 \text{ for } i = 1, ..., n \right\}.$$

The general linearized model of problem (14), which comes from replacing each polynomial term with a single additional 0-1 variable and two additional constraints, is as follows.

$$\textit{Minimize} \quad \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} G_{ij} y_{ij} \tag{15}$$

$$\textit{subject to} \quad (a_{kj} - b_k) x_j + \sum_{i<j} a_{ki} y_{ij} + \sum_{i>j} a_{ki} y_{ji} = 0 \qquad \forall\, j,k \tag{15.1}$$

$$\sum_i H_{li} x_i - h_l \geq \sum_{i<j} H_{li} y_{ij} + \sum_{i>j} H_{li} y_{ji} + (H_{lj} - h_l) x_j \geq 0 \qquad \forall\, j,l \tag{15.2}$$

$$x_i - y_{ij} \geq 0 \qquad \forall\, i, j, i < j \tag{15.3}$$

$$x_j - y_{ij} \geq 0 \qquad \forall\, i, j, i < j \tag{15.4}$$

$$-x_i - x_j + y_{ij} \geq -1 \qquad \forall\, i, j, i < j \tag{15.5}$$

$$y_{ij} \geq 0 \qquad \forall\, i, j, i < j \tag{15.6}$$

$$x \in S_1, x \in S_3, x \text{ is binary} \tag{15.7}$$

However, as tight as this model is, it is impractical when the problem is large. Glover and Woolsey (1974) proposed the proven equivalent problem of problem (15), which is more concise. It can be easily obtained by deleting constraints (15.1) and (15.2) but explicitly adding $x \in S_2$ in constraint (15.7). Later, Glover (1975) proposed an alternative linearization technique for QIP problem, which does not employ additional 0-1 variables. This technique produces the following equivalent linear mixed-integer representation of problem (14).

$$
\begin{aligned}
\textit{Minimize} \quad & \sum_{i=1}^{n} c_i x_i + \frac{1}{2}\sum_{i=1}^{n} B_i && (16)\\
\textit{subject to} \quad & \sum_{j<i} G_{ji}x_j + \sum_{j>i} G_{ij}x_j - d_i^{+}(1-x_i) \le B_i \quad \forall i\\
& \sum_{j<i} G_{ji}x_j + \sum_{j>i} G_{ij}x_j - d_i^{-}(1-x_i) \le B_i \quad \forall i\\
& g_i^{-} x_i \le B_i \le g_i^{+} x_i \quad \forall i\\
& x \in S \text{ and binary}
\end{aligned}
$$

, where g_i^- and g_i^+ are respectively the minimum and maximum values of the expression $\sum_{j<i} G_{ji}x_j + \sum_{j>i} G_{ij}x_j$ over the set *S* for each *i*. As suggested by Glover (1975), one may equivalently use simply lower and upper bounds on the optimal values of these problems for g_i^- and g_i^+ respectively, although this of course weakens the linear programming relaxation.

More recent linearization technique was proposed by Adams and Sherali (1986). Their proposed algorithm is an implicit enumerative algorithm, which uses Lagrangean Relaxation, Bender's cutting plane, and local explorations to exploit the strength of this linearization. In their paper, they also showed the numerical experiments compared with solving linearization problem (15). The results showed that their algorithm works well when c_i and G_{ij} are positive real numbers but it does not converge to the good solution when c_i is a negative real number.

MATERIALS AND METHODS

Materials

The materials used in this research can be categorized into two groups.

1. Computer Hardware

A personal computer with the following specifications

1.1 CPU 1.6 GHz

1.2 Pentium IV

1.3 Ram 760 MB

1.4 Microsoft operation system

is used to do the numerical experiments, to test the proposed algorithm for CMLCP and to evaluate the results.

2. Computer Software

There are three software programming packages used in this research.

2.1 MATLAB version R2006a is used to code the program according to the proposed algorithm and the empirical exact algorithm.

2.2 SPSS 14.0 is used to validate the solutions of the proposed algorithm by constructing confidence interval of the percentage of error of the solutions comparing with that of the empirical exact algorithm.

2.3 Microsoft Word is used to develop this research manuscript.

Methods

This part mentions about the methodology to develop the algorithms for CMLCP. In this part, the theoretical principles supporting the proposed algorithms are

shown. The methods to design the numerical experiments and to generate data for these experiments are also discussed. Moreover, the method to validate the results from the numerical experiments using relevant statistical techniques is also explained. Firstly, the methodology composes of four main steps as follows.

1. Problem Definition and Mathematical Model Formulation

The studied problem is described as follows. There are $m > 1$ new facilities with a certain capacity to be located on the continuous plane. They have to serve n customers in their responsibilities whose locations and inseparable products or demands are known and deterministic. The objective is to find the good locations of these new facilities and allocation of customers to them so as to minimize total distances measured in squared-Euclidean metric with respect to facility capacity. It can be mathematically formulated as follows.

$$\textit{Minimize} \quad \sum_{i=1}^{m}\sum_{j=1}^{n} z_{ij} w_j [(x_i - a_j)^2 + (y_i - b_j)^2] \tag{17}$$

$$\textit{subject to} \quad \sum_{i=1}^{m} z_{ij} = 1 \qquad ; j = 1,...,n$$

$$\sum_{j=1}^{n} z_{ij} w_j \le s_i \qquad ; i = 1,...,m$$

$$\textit{,where} \quad z_{ij} = \begin{cases} 1, \text{if customer } j \text{ is assigned to facility } i \\ 0, \text{otherwise} \end{cases}$$

s_i is capacity of facility i ; $i = 1,...,m$

w_j is demand of customer j ; $j = 1,...,n$

(a_j, b_j) is known co-ordinate of customer j on plane

(x_i, y_i) is unkown co-ordinate of facilities i on plane

The objective function above gives the total transportation cost, while the first constraint set ensures that all customer demands are satisfied and the second constraint set ensures that all facility capacity limitations are controlled. If the allocation variable z_{ij} is fixed, the unconstrained minimum of the strictly convex

objective function is readily obtained by partial derivatives of equation (17) to x_i and y_i at the solution

$$x_i = \frac{\sum_{j=1}^{n} z_{ij} w_j a_j}{\sum_{j=1}^{n} z_{ij} w_j} \quad \text{and} \quad y_i = \frac{\sum_{j=1}^{n} z_{ij} w_j b_j}{\sum_{j=1}^{n} z_{ij} w_j} \quad ; \quad i = 1, ..., m \tag{18}$$

Substituting (18) into the objective function of problem (17), the objective function can be projected onto the space of z_{ij} variables as follows.

$$\textit{Minimize} \quad \sum_{i=1}^{m}\sum_{j=1}^{n} \frac{z_{ij} w_j}{\left(\sum_{j=1}^{n} z_{ij} w_j\right)^2} \left[\left(\sum_{k=1}^{n} z_{ik} w_k a_k - a_j \sum_{k=1}^{n} z_{ik} w_k\right)^2 + \left(\sum_{k=1}^{n} z_{ik} w_k b_k - b_j \sum_{k=1}^{n} z_{ik} w_k\right)^2 \right]$$

$$= \sum_{i=1}^{m}\sum_{j=1}^{n} \frac{z_{ij} w_j}{\left(\sum_{j=1}^{n} z_{ij} w_j\right)^2} \left[\left(\sum_{k=1}^{n} z_{ik} w_k a_k)\right)^2 - 2a_j \left(\sum_{k=1}^{n} z_{ik} w_k a_k\right)\left(\sum_{k=1}^{n} z_{ik} w_k\right) + a_j{}^2 \left(\sum_{k=1}^{n} z_{ik} w_k\right)^2 \right]$$

$$+ \sum_{i=1}^{m}\sum_{j=1}^{n} \frac{z_{ij} w_j}{\left(\sum_{j=1}^{n} z_{ij} w_j\right)^2} \left[\left(\sum_{k=1}^{n} z_{ik} w_k b_k\right)^2 - 2b_j \left(\sum_{k=1}^{n} z_{ik} w_k b_k\right)\left(\sum_{k=1}^{n} z_{ik} w_k\right) + b_j{}^2 \left(\sum_{k=1}^{n} z_{ik} w_k\right)^2 \right]$$

$$= \sum_{i=1}^{m}\sum_{j=1}^{n} \frac{z_{ij} w_j}{\left(\sum_{j=1}^{n} z_{ij} w_j\right)^2} \left[\left(\sum_{k=1}^{n} z_{ik} w_k a_k\right)^2 + \left(\sum_{k=1}^{n} z_{ik} w_k b_k\right)^2 \right] - 2\sum_{i=1}^{m}\sum_{j=1}^{n} \frac{z_{ij} w_j}{\left(\sum_{j=1}^{n} z_{ij} w_j\right)^2} \left(\sum_{k=1}^{n} z_{ik} w_k\right)$$

$$\times \left[a_j \left(\sum_{k=1}^{n} z_{ik} w_k a_k\right) + b_j \left(\sum_{k=1}^{n} z_{ik} w_k b_k\right) \right] + \sum_{i=1}^{m}\sum_{j=1}^{n} \frac{z_{ij} w_j}{\left(\sum_{j=1}^{n} z_{ij} w_j\right)^2} \left(\sum_{k=1}^{n} z_{ik} w_k\right)^2 \left(a_j{}^2 + b_j{}^2\right)$$

Therefore, the objective function (17) becomes

$$\textit{Minimize} \quad -\sum_{i=1}^{m} \frac{1}{\sum_{j=1}^{n} z_{ij} w_j} \left[\left(\sum_{j=1}^{n} z_{ij} w_j a_j\right)^2 + \left(\sum_{j=1}^{n} z_{ij} w_j b_j\right)^2 \right] + \sum_{j=1}^{n} w_j (a_j{}^2 + b_j{}^2) \tag{19}$$

, which is equivalent to

$$Minimize \quad -\sum_{i=1}^{m} \frac{1}{\sum_{j=1}^{n} z_{ij} w_j} \left[(\sum_{j=1}^{n} z_{ij} w_j a_j)^2 + (\sum_{j=1}^{n} z_{ij} w_j b_j)^2 \right] \tag{20}$$

Or,

$$Maximize \quad \sum_{i=1}^{m} \frac{1}{\sum_{j=1}^{n} z_{ij} w_j} \left[(\sum_{j=1}^{n} z_{ij} w_j a_j)^2 + (\sum_{j=1}^{n} z_{ij} w_j b_j)^2 \right] \tag{21}$$

The problem (17) with objective function (20) can be rewritten in matrix form as following model.

$$Minimize \quad f(z) = -\sum_{i=1}^{m} \frac{1}{r_i} \left[(z_i{}^T wa)^2 + (z_i{}^T wb)^2 \right] \equiv -\sum_{i=1}^{m} \frac{1}{r_i} \left[z_i{}^T H z_i \right] \equiv \frac{1}{2} z^T G z \tag{22}$$

$$subject\ to \quad \sum_{i=1}^{m} z_{ij} = 1 \quad ; \ j = 1,...,n$$

$$r_i = \sum_{j=1}^{n} z_{ij} w_j \le s_i \quad ; \ i = 1,...,m$$

$$z_{ij} \in \{0,1\} \quad ; \ i = 1,...,m \ and \ j = 1,...,n$$

$$, where \quad wa = [w_1 a_1, ..., w_n a_n]^T \ ; \forall \ j = 1,...,n$$

$$wb = [w_1 b_1, ..., w_n b_n]^T \ ; \forall \ j = 1,...,n$$

$$z_i = [z_{i1}, z_{i2}, ..., z_{in}]^T \ ; \forall \ i = 1,...,m$$

$$z = [z_{11}, z_{21}, ..., z_{m1}, z_{1j}, z_{2j}, ..., z_{mj}, z_{1n}, z_{2n} ..., z_{mn}]^T ; \forall i \text{ and } j$$

$$H = (wa \bullet wa^T + wb \bullet wb^T)$$

$G =$ a negative semi-definite symmetric matrix

$=$ Hessian Matrix of objective function (20)

Since the reduced problem (22) is equivalent to the original problem (17) but contains a smaller number of variables, then model (22) will be used to represent the studied problem and the methods proposed here is created for problem (22).

2. Algorithm Development

Observe that problem (22) is a sum-of-ratios problem. According to literature review, up to now no exact algorithm or even heuristic algorithm for this kind of

problem is found. As a result, both exact algorithm and heuristic algorithm are proposed in this research. The former, which provides the optimal solution, is created to verify and validate the latter, while the latter is developed in order to reduce the unacceptable computational effort of the former. These two algorithms are as follows.

2.1 Exact and Empirical Exact Algorithm for CMLCP

Naïve approach to develop exact algorithm is to construct all possible solutions or allocations (z_{ij}) and then replace them into the objective function and finally the solution that gives the minimum objective function value is an optimal solution. Using this approach is terrible, even it can ensure to obtain the optimal solution, because the number of solutions to be considered increases nonlinearly when number of z_{ij} grows. For problem with *m* facilities *n* customers, there are at most $2^{m\times n}$ possible solutions. To reduce the computational task of using this approach, the good property of the studied problem will be used. The exact algorithm developed in this research will implicitly generate the optimal solution z_{ij} from explicitly branching on r_i with some special techniques.

The special techniques arise from observing the combinations of problem (22). Notice from problem (22) that each denominator (r_i) appears in the capacity constraint set. Therefore, it relates to capacity of each facility. Actually, it is the amount of resource used from each facility. If it is fixed at a certain value, there will be two advantages. The first advantage is the problem will be reduced from the minimizing sum of *m* fractions to sum of *m* functions. And then, the reduced problem, which is minimizing concave QIP, can be solved using the efficient specific method. Another advantage is that the problem will be subject to equality constraint ($r_i = \sum_{j=1}^{n} z_{ij} w_j$) and the constraint set becomes a set of balanced transportation constraints, which is easier than solving the problem with inequality constraints. Therefore, the problem (22) with fixing each denominator at r_i becomes the following model.

$$\text{P1: } \textit{Minimize} \quad f(z) = -\sum_{i=1}^{m} \frac{1}{r_i}\left[(z_i^T wa)^2 + (z_i^T wb)^2 \right] \equiv -\sum_{i=1}^{m} \frac{1}{r_i}\left[z_i^T H z_i \right] \equiv \frac{1}{2} z^T G z \qquad (23)$$

$$\textit{subject to} \quad \sum_{i=1}^{m} z_{ij} = 1 \quad ; \; j = 1,...,n$$

$$\sum_{j=1}^{n} z_{ij} w_j = r_i \quad ; \; i = 1,...,m$$

$$z_{ij} \in \{0,1\} \quad ; \; i = 1,...,m \; \textit{and} \; j = 1,...,n$$

To use this nice property of the problem, the proposed exact algorithm starts with determining the lower bound and upper bound of denominators. After that, iterative procedure of fixing value of denominators and solving minimizing concave QIP problem will be done. The algorithm will stop when all possible combinations of denominators are visited. Finally, the solution which provides the minimum objective function value is an optimal solution. It seems to be an easy idea but it is really hard to do if there is no well-designed technique in each step. Therefore, before going on to the summarized flow chart of this algorithm, the special techniques and methods will be expressed elaborately as follows.

2.1.1 Determination of Lower and Upper Bound of Denominators

Lower bound and upper bound of each denominator is very essential for this algorithm because it determines number of combinations to be visited. If the range of each denominator determined is too loose (or wide), the number of combinations will be larger than it should be. On contrast, if the range of each denominator is smaller than the actual range, there are some possible combinations ignored. If range of a denominator, let say r_1, increases by one, number of additional branches will increase by $\prod_{i=2}^{m} (\textit{upper bound of } r_i - \textit{lower bound of } r_i)$.

The easiest and least computational effort way to determine the lower bound an upper bound on r_i is using algebra with the demand constraints as

follows. The upper bound on r_i is obviously s_i. From the demand constraints, $\sum_{i=1}^{m} z_{ij} = 1$, multiply by w_j to both sides. The equation becomes

$$\begin{aligned} w_j \sum_{i=1}^{m} z_{ij} &= w_j \\ \sum_{j=1}^{n}\sum_{i=1}^{m} z_{ij} w_j &= \sum_{j=1}^{n} w_j \\ \sum_{i=1}^{m} r_i &= \sum_{j=1}^{n} w_j \\ r_i + \sum_{\substack{k=1 \\ k\neq i}}^{m} r_k &= \sum_{j=1}^{n} w_j \end{aligned} \tag{24}$$

And therefore, the minimum r_i occurs when $\sum_{\substack{k=1 \\ k\neq i}}^{m} r_k$ is maximum, which can occur when each r_k is set at s_k. And, from demand constraints and assumptions of non redundant facility, a customer has to receive products from a facility and a facility has to provide products to at least one customer. Thus, the lower bound on r_i is $\max\left\{ \min(w_j : j = 1,...,n), \sum_{j=1}^{n} w_j - \sum_{\substack{k=1 \\ k\neq i}}^{m} r_k \right\}$.

Although it is easier to use algebraic method to find the corresponding lower bound and upper bound on r_i, the obtained range of r_i is too loose. So, the proposed special technique will provide the tighter range of r_i with a little bit higher computational effort as follows. Finding for lower bound and upper bound on r_i is independent to finding minimum and maximum values of r_i respectively subject to the constraints of problem (22). They can be represented by the following mathematical models. Model (25) is used for finding the lower bound on r_i for each *i*.

$$\begin{aligned} &\textit{Minimize} && \sum_{j=1}^{n} z_{ij} w_j && && (25)\\ &\textit{subject to} && \sum_{k=1}^{m} z_{kj} = 1 && ;\ j = 1,...,n \\ & && \sum_{j=1}^{n} z_{kj} w_j \le s_i && ;\ k = 1,...,m \\ & && z_{kj} \in \{0,1\} && ;\ k = 1,...,m \textit{ and } j = 1,...,n \end{aligned}$$

And, solving model (26), an upper bound on r_i for each *i* can be obtained.

$$\begin{aligned} &\textit{Maximize} && \sum_{j=1}^{n} z_{ij} w_j && && (26)\\ &\textit{subject to} && \sum_{k=1}^{m} z_{kj} = 1 && ;\ j = 1,...,n \\ & && \sum_{j=1}^{n} z_{kj} w_j \le s_i && ;\ k = 1,...,m \\ & && z_{kj} \in \{0,1\} && ;\ k = 1,...,m \textit{ and } j = 1,...,n \end{aligned}$$

Using this technique minimum and maximum values of r_i will be independently considered in the feasible region of the problem. Therefore, the provided lower bound and upper bound on r_i is the tightest lower bound and upper bound on r_i.

2.1.2 Branching Strategy

Owing to being discrete problem of the studied problem, minimum changing in value of each r_i is one unit. After the lower bound and upper bound of each component r_i is found, matrix containing these values of each r_i can be written as the following matrices.

$$\begin{bmatrix} LBr_1 & , & UBr_1 \\ LBr_2 & , & UBr_2 \\ : & , & : \\ LBr_m & , & UBr_m \end{bmatrix} \quad or \quad \begin{bmatrix} \underline{r_1} & , & \overline{r_1} \\ \underline{r_2} & , & \overline{r_2} \\ : & , & : \\ \underline{r_m} & , & \overline{r_m} \end{bmatrix}$$

, where LBr_i and UBr_i represent lower bound and upper bound on r_i respectively. Recognizing that all facilities are identical, the combination that arises from alternate position of same set of values of r_i is equivalent. The position of component r_i in the matrix is then arranged in order of non-decreasing of UBr_i so as to be easy for checking the repeat of the vector r when the branching mechanism goes on. This means that $UBr_1 < UBr_2 < ... < UBr_m$.

The first combination of vector r is to set each component of vector r, r_i for every $1 \le i \le m$, at the minimum value. This minimum value means the minimum value of r_i that will not make other lower level components $r_k; \forall i < k \le m$ out of their upper bound. And, the last combination of vector r is to set each component of vector r, r_i for every $1 \le i \le m$, at the maximum value. Likewise, this maximum value means the maximum value of r_i that will not make other lower level components $r_k; \forall i < k \le m$ out of their lower bound. And, the minimum and maximum values of r_i can change when the iterative branching procedure goes on.

The determination of minimum and maximum values of component r_i is carefully done in order that the vector r that makes surely infeasible solution (due to the outbound component r_k) will be eradicated from iterative procedure and then the computational time will decrease numerously. The technique to calculate minimum and maximum values of r_i is based on the demand constraints set again. From equation (24), $\sum_{i=1}^{m} r_i = \sum_{j=1}^{n} w_j \rightarrow \sum_{l=1}^{i-1} r_l + r_i + \sum_{k=i+1}^{m} r_k = \sum_{j=1}^{n} w_j$. Therefore, the minimum and maximum values of r_i are as follows.

$$\text{minimum } r_i = \max\left\{LBr_i,\ \sum_{j=1}^{n} w_j - \sum_{l=1}^{i-1} r_l - \sum_{k=i+1}^{m} \overline{r_k}\right\} \tag{27}$$

$$\text{maximum } r_i = \min\left\{UBr_i,\ \sum_{j=1}^{n} w_j - \sum_{l=1}^{i-1} r_l - \sum_{k=i+1}^{m} \underline{r_k}\right\} \tag{28}$$

, *where* r_i is the i^{th} component, which is being considered.

r_l is the l^{th} component, where $l < i$. It is the component, whose value has been fixed at the considered iteration.

r_k is the k^{th} component, where $k > i$. It is the component, whose value is free to move from LBr_k to UBr_k.

r_m ,which is the lowest level component, will be fixed at $r_m = \sum_{j=1}^{n} w_j - \sum_{l=1}^{m-1} r_l$

When branching procedure goes on, these values are also changed. Branching procedure starts at the $(m-1)^{th}$ level by adding its minimum value by one while other components are fixed and the procedure will stop at the first set of iterations when r_{m-1} reaches its maximum value. After that, $(m-2)^{th}$ level is added by one and again r_{m-1} will vary from its minimum to its maximum value before another one unit will be added to r_{m-2}.

This iterative procedure will be done in the same way until the first level of component is fixed at its maximum value and the lower level components vary from their minimum to maximum. To understand the concept clearly, the following numerical example of branching strategy is shown.

Example of Branching Procedure: Assume $\sum_{j=1}^{n} w_j$ = 25 and matrix of range of vector *r* obtained from method described in section 2.1.1 is as follows.

$$r = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} \in \begin{bmatrix} 5 & , & 10 \\ 9 & , & 12 \\ 6 & , & 9 \end{bmatrix} \xrightarrow[\text{Sort } UBr]{} r \in \begin{bmatrix} 6 & , & 9 \\ 5 & , & 10 \\ 9 & , & 12 \end{bmatrix}$$

$\min r_1 = \max\{6, 25-10-12\} = 6 \qquad \max r_1 = \min\{9, 25-5-9\} = 9$

At $r_1 = 6$ $\quad \min r_2 = \max\{5, 25-6-12\} = 7 \qquad \max r_2 = \min\{10, 25-6-9\} = 10$

Combinations at this iteration are as follows.

$$\begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} 6 \\ 7 \\ 25-7-6=12 \end{bmatrix}, \begin{bmatrix} 6 \\ 8 \\ 11 \end{bmatrix}, \begin{bmatrix} 6 \\ 9 \\ 10 \end{bmatrix}, \begin{bmatrix} 6 \\ 10 \\ 9 \end{bmatrix}$$

At $r_1 = 7$ $\min r_2 = \max\{5, 25-7-12\} = 6$ $\max r_2 = \min\{10, 25-7-9\} = 9$

Combinations at this iteration are as follows.

$$\begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 6 \\ 12 \end{bmatrix}, \begin{bmatrix} 7 \\ 7 \\ 11 \end{bmatrix}, \begin{bmatrix} 7 \\ 8 \\ 10 \end{bmatrix}, \begin{bmatrix} 7 \\ 9 \\ 9 \end{bmatrix}$$

At $r_1 = 8$ $\min r_2 = \max\{5, 25-8-12\} = 5$ $\max r_2 = \min\{10, 25-8-9\} = 8$

Combinations at this iteration are as follows.

$$\begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} 8 \\ 5 \\ 12 \end{bmatrix}, \begin{bmatrix} 8 \\ 6 \\ 11 \end{bmatrix}, \begin{bmatrix} 8 \\ 7 \\ 10 \end{bmatrix}, \begin{bmatrix} 8 \\ 8 \\ 9 \end{bmatrix}$$

At $r_1 = 9$ $\min r_2 = \max\{5, 25-9-12\} = 5$ $\max r_2 = \min\{10, 25-9-9\} = 7$

Combinations at this iteration are as follows.

$$\begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} 9 \\ 5 \\ 11 \end{bmatrix}, \begin{bmatrix} 9 \\ 6 \\ 10 \end{bmatrix}, \begin{bmatrix} 9 \\ 7 \\ 9 \end{bmatrix}$$

Observe that some branches such as $r_1 = 6, r_2 = 6, r_3 = 25-6-6=13$, which is infeasible combination due to outbound r_3, does not appear if this technique is used. So, using this branching technique can get rid of the infeasible branches, which are usually large when number of *m* increases, and then can be expected to reduce the total computational effort.

2.1.3 Finding the Solution to Minimizing Concave QIP Problem

After replacing vector r at a certain value according to branching strategy into problem (22), the problem becomes problem (23), which is minimizing

concave QIP problem. The exact method for such problem is using reformulated linearization techniques (RLT) mentioned in the literature review part. This method provides the optimal solution but with unacceptable large computational effort. To reduce the computational time, another method, which is Extreme Point Ranking (EPR) method, is proposed.

This method uses the same basic idea as extreme point ranking method for QIP proposed by Gupta *et al.* (1996) but with the different technique. The provided solutions can be tested for optimality using algebraic method. To avoid completely visiting all possible extreme points in the worst case, some additional stopping rules are added. Therefore, this method may provide some near-optimal solutions. Thus, before using EPR method at each branch of vector *r*, it will be carefully tested and validated with linearization method that it can provide solution with 0% of error on at least 99% confidence interval. Therefore, the algorithm of combining EPR method with other techniques in other steps is called the empirical exact algorithm (EEA) instead of exact algorithm, which is obtained from combining linearization method with other techniques in other steps. These two methods (EPR method and linearization method) can be theoretical illustrated as follows.

I. Extreme Point Ranking Method

Because of 0-1 variables and balanced transportation constraints of problem (23), preprocessing method by using logic based method and exchanging method will be used to reduce the computational time with retaining the same quality of solutions. These methods used in the proposed EPR method can be described as follows.

A. Theoretical Approach of EPR Method

The basic idea of extreme point ranking is to rank the vertices of the polytope defining the feasible region in order of importance regarding the global solution. Starting from one of the vertices of the polytope, the nearby vertices

are ranked using an extreme point approach. This provides a new vertex to move to and the process continues until no adjacent vertices can be found with a decreasing objective function value. At each step, linear integer programming problem P2 shown below is solved to provide a lower bound on the objective function value of the quadratic integer programming problem P1 (Gupta *et al.*, 1996), while the upper bound can easily be calculated by substituting this solution in $f(z)$.

$$
\begin{aligned}
P2: Minimize \quad & g(z) = \frac{1}{2}Uz \\
subject\ to \quad & \sum_{i=1}^{m} z_{ij} = 1 \qquad ; \ j = 1,...,n \\
& \sum_{j=1}^{n} z_{ij} w_j = r_i \qquad ; \ i = 1,...,m \\
& z_{ij} \in \{0,1\} \qquad ; \ i = 1,...,m \ and \ j = 1,...,n
\end{aligned}
\tag{29}
$$

and, U_c which is the c^{th} column of matrix *U* can be found by solving

$$
\begin{aligned}
U_c = Minimize \quad & z^T G_c \\
subject\ to \quad & \sum_{i=1}^{m} z_{ij} = 1 \qquad ; \ j = 1,...,n \\
& \sum_{j=1}^{n} z_{ij} w_j = r_i \qquad ; \ i = 1,...,m \\
& z_{ij} \in \{0,1\} \qquad ; \ i = 1,...,m \ and \ j = 1,...,n \\
& , where \ G_c \ is \ the \ c\text{-}th \ column \ of \ G.
\end{aligned}
\tag{30}
$$

Proposition 1 proved below explains how the solution of P2 can provide a lower bound on objective function P1.

Proposition 1: The solution of problem P2 provides a lower bound on objective function of P1.

Proof: Let *S* be the convex set of constraints: $\sum_{i=1}^{m} z_{ij} = 1$, $\sum_{j=1}^{n} z_{ij} w_j = r_i$, and $z_{ij} \in \{0,1\}$; $i = 1,...,m \, and \, j = 1,...,n$. Since each U_c is obtained from optimal solution, then $U_c = \min z^T G_c \le z^T G_c; \forall c = 1,...,m \times n$. Multiplying $\frac{1}{2} \times z$ to both sizes, the equation becomes

$\frac{1}{2}U_c z \le \frac{1}{2}z^T G_c z \rightarrow\ g(z)\ \le f(z), \forall z \in S.$ This equation shows that all feasible solutions in S will not makes the objective function value of P2 over that of P1. Thus, P2 is the lower bound of P1. ■

To solve problem P1 and P2, Hessian matrix (*G*) has to be developed. And, to construct the Hessian matrix, partial derivative corresponding to $z_{ij}, \forall i = 1,...,m; j = 1,...,n$ has to be calculated. For problem with *m* facilities *n* customers, there are up to $(m \times n)^2$ partial derivative to be calculated, which will take a long time and then impede the efficiency of this method when *m* and *n* are large. To reduce time used to construct the Hessian matrix theoretically developed by calculus we proposed another method, which uses only algebraic skill yet provides the equivalent Hessian matrix. The algebraic method arises from finding the closed form solution of the partial derivative terms corresponding to each $z_{ij}, \forall i = 1,...,m; j = 1,...,n$. The closed form solutions are as follows.

$$\frac{\partial f(z)}{\partial z_{ij}^2} = \frac{2w_j^2(a_j^2 + b_j^2)}{r_i}$$

$$\frac{\partial f(z)}{\partial z_{ij}\partial z_{ik}} = \frac{2w_j w_k(a_j a_k + b_j b_k)}{r_i}$$

Plugging these closed form solutions into their related position in the original Hessian matrix shown below, the equivalent Hessian matrix will be provided.

$$G = \begin{bmatrix} \frac{\partial^2 f(z)}{\partial z_{11}{}^2} & \frac{\partial^2 f(z)}{\partial z_{11}\partial z_{21}} & \cdots & \frac{\partial^2 f(z)}{\partial z_{11}\partial z_{mn}} \\ \frac{\partial^2 f(z)}{\partial z_{21}\partial z_{11}} & \frac{\partial^2 f(z)}{\partial z_{21}{}^2} & \cdots & \frac{\partial^2 f(z)}{\partial z_{21}\partial z_{mn}} \\ \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial^2 f(z)}{\partial z_{mn}\partial z_{11}} & \frac{\partial^2 f(z)}{\partial z_{mn}\partial z_{21}} & \cdots & \frac{\partial^2 f(z)}{\partial z_{mn}{}^2} \end{bmatrix}$$

To obtain solutions for U_c, $g(z)$, and $f(z)$ logic based method shown in the following section B will be combined to reduce the number of variables. And then, the next adjacent vertices will be explored using exchanging method, which will be shown in the following section C, and be ranked using extreme point ranking approach. Proposition 2, proved below, explains how ranking feasible solutions of P2 can provide the optimal solution for P1.

Proposition 2: Let z_p be the p^{th} extreme points ranked ascendingly by the objective function value of $g(z)$ and T^k be the set of all extreme points collected from the 1st to the k^{th} set of adjacent extreme points. If $g(z_p) \geq \min\{ f(z) : z \in T^k \} = f(z^*)$, then z^* is an optimal solution of P1.

Proof: Since $g(z_p)$ is the value of $g(z)$ at the p^{th} best integer feasible solution of P2, then $g(z_v) > g(z_p)$; $\forall v \geq p+1$. Also, $f(z_v) \geq g(z_v) > g(z_p) \geq \min\{ f(z) : z \in T^k \} = f(z^*)$ That is $f(z_v) > f(z^*)$; $\forall v \geq p+1$, which means that $f(z^*)$ is the least among the values of $f(z)$ at all the integer feasible solution in *S*. Thus, z^* is an optimal solution of problem P1. ■

B. Logic Based Method

The objective of using logic here is to tighten bound of variables. Since variables in the problem are 0-1 variables, tightening bound of variables is equivalent to fixing value of some variables. This means that the number of variables to be branched will be reduced. Therefore, the computational time will be reduced. The logic using to fix variable here is fixing $z_{ij} = 0$, if $w_j > r_i$. Observe that this logic will work only when some facilities have capacity limitation (r_i in this case) less than the maximum amount of products shipped: $r_i < \max\{ w_j ; j = 1, ..., n \}$

C. Exchanging Method

Although the existing method for finding the next adjacent extreme points in quadratic integer programming problem is cutting plane method (Gupta *et al.*, 1996), it was shown to be non-converge due to cycling or an infinite sequence of cutting planes by Zwart (1973). The reason for the cycling behavior as well as non-convergence of this approach lies in the fact that although the approach generates cones during the algorithm, it fails to explicitly incorporate these cones into the remaining steps. This is essential to avoid the reemergence of vertices that have already been considered (Floudas and Visweswaran, 1995). To avoid the cycling, the exchanging method is proposed here. The problem here can be classified into clustering problem with balanced transportation constraints. Changing value of allocation variables between zero and one, which means changing cluster of a customer, affects both demand and supply constraints. To conserve the balance of the constraints a customer can move from a current facility to another facility only when there is another customer requiring the same amount of products or a group of some customers whose summation of amount of products equal to that of leaving customers to exchange with. The exchanging method proposed here exchanges customers only one pair of facilities at a time and does not consider crossing of the pair to avoid exponentially growth of computational time. The customers to be exchanged will be considered in order of appearing in the vector of variables. Therefore, no cycling emerges. The exchanging method can be summarized as follows. Let *t* be number of customers considered to be move out at a time. At the k^{th} adjacent extreme point, *t* customers running from one to *k* customers of a current facility will be exchanged with *k* customers of the other facility whose summation of amount of products equal to that of the *t* customers. Observing that the maximum value of *k* is the maximum number of customers assigned to each facility.

In summary, the proposed EPR method can be summarized as follows.

INPUT: Locations (a_j, b_j) of customers j on continuous plane; $j = 1,...,n$

Demands w_j of customer j ; $j = 1,...,n$

Capacity r_i of facility i ; $i = 1,...,m$

Convex set *S* is a constraint set of $\sum_{i=1}^{m} z_{ij} = 1, \forall\ j$ and $\sum_{j=1}^{n} z_{ij} w_j = r_i, \forall\ i$.

Step 1: Find the initial solution or initial extreme point z_0 by solving problem P2. Take $f_l = g(z_0)$ as a lower bound and $f_u = f(z_0)$ as an upper bound on f^* (by proposition 1). Take z_0 as the "current best solution" to P1. If $f_u = f_l$, the current best solution is an optimal solution (by proposition 2) and then stop. Otherwise, go to step 2.

Step 2: Search for the new "current best solution", which will be the best incumbent solution to be searched for its next adjacent vertices, $z_0^* = \arg\min f(z_c)$, $\forall c = 1,...,m \times n$ where z_c is the optimal solution of U_c. This step is done in order to accelerate process of moving to a peak of a function $f(z)$ explained as follows.

Observing U_c, we will find that it is equivalent to finding the minimum value of gradient function of $f(z)$, which is

$$U_c = \textit{Minimize}\ \ z^T G_c = \nabla f(z),\ \ \forall c = 1,...,m \times n$$

If the problem is an unconstrained quadratic function, U_c should be zero if $z = z_c$ is an optimal solution. But, problem P1 is a constrained quadratic function subject to many-facet constraints leading to nonzero U_c. Therefore, z_c should be considered as high priority extreme points to be explored because it is close to the peak of the function $f(z)$. Also, the adjacent extreme points around the best incumbent extreme point, which has the steepest improvement rate of the initial solution, have higher possibilities to provide the optimal solution.

Take $\max\{g(z_c), \forall c = 1,...,m \times n\}$ and $\min\{f(z_c), \forall c = 1,...,m \times n\}$ as a new lower bound and a new upper bound respectively.

Step 3: Find the next adjacent extreme points using exchanging method proposed in section C. Set k =1. If $g(z_p) \geq f_u; z \in T^k$, then stop. The current best solution is an optimal solution to P1 (by proposition 2).

According to the exchanging method proposed in section C, not all possible (but some high possibility to be an optimal solution) extreme points are considered. Therefore, this optimal condition may not be satisfied completely and then two additional stopping rules are constructed as follows.

- No more possible exchanging pairs of customers exist.
- There is no improvement on f_u within q = two consecutive sets of adjacent extreme points. Note that q can be more than two. But, the higher value of q is, the more computational time is required.

If at least one of these additional stopping rules is satisfied, the existing current best solution is the final solution. And, $f^* = f_u$. If $g(z_p) < f_u$ and the additional stopping rules are not satisfied, then replace f_l by $g(z_p)$.

Step 4: If $f(z_p) < f_u$, then replace f_u by $f(z_p)$ and replace the current best solution to P1 by z_p. Otherwise, set $k = k+1$ and return to step 3 without changing f_u or the current best solution.

II. Linearization Method

To verify and validate the EPR method the difference of solutions and processing time of the EPR method and linearization method, which is

usually used for quadratic integer programming problem, will be analyzed. The linearization method used here is the method that was proposed by Glover and Woolsey (1974). To enhance the efficiency, logic based method shown in section 2.1.3.I.B will be combined. The objective function (19) can be linearized as follows.

$$\begin{aligned}
\textit{Minimize}\quad \phi &= -\sum_{i=1}^{m}\frac{1}{r_i}\left[\left(\sum_{j=1}^{n} z_{ij}w_j a_j\right)^2 + \left(\sum_{j=1}^{n} z_{ij}w_j b_j\right)^2\right] + \sum_{j=1}^{n} w_j(a_j{}^2 + b_j{}^2) \\
&= -\sum_{i=1}^{m}\frac{1}{r_i}\left[\sum_{j=1}^{n} w_j^2\left(a_j^2+b_j^2\right)z_{ij}^2 + 2\sum_{j=1}^{n-1}\sum_{k>j}^{n} w_j w_k\left(a_j a_k + b_j b_k\right)\underbrace{z_{ij}z_{ik}}_{z_{ijk}}\right] + \sum_{j=1}^{n} w_j(a_j{}^2 + b_j{}^2) \\
&= -\sum_{i=1}^{m}\frac{1}{r_i}\left[\sum_{j=1}^{n} w_j^2\left(a_j^2+b_j^2\right)z_{ij} + 2\sum_{j=1}^{n-1}\sum_{k>j}^{n} w_j w_k\left(a_j a_k + b_j b_k\right)z_{ijk}\right] + \sum_{j=1}^{n} w_j(a_j{}^2 + b_j{}^2)
\end{aligned}$$

Therefore, the linearized problem of problem (19) can be written as follows.

$$\textit{Minimize}\quad \phi = -\sum_{i=1}^{m}\frac{1}{r_i}\left[\sum_{j=1}^{n} w_j^2\left(a_j^2+b_j^2\right)z_{ij} + 2\sum_{j=1}^{n-1}\sum_{k>j}^{n} w_j w_k\left(a_j a_k + b_j b_k\right)z_{ijk}\right] \tag{31}$$

$$\textit{subject to}\quad \sum_{i=1}^{m} z_{ij} = 1 \qquad ;\ j = 1,\ldots,n \tag{31.1}$$

$$\sum_{j=1}^{n} z_{ij}w_j = r_i \qquad ;\ i = 1,\ldots,m \tag{31.2}$$

$$\left.\begin{aligned}
-z_{ij} + z_{ijk} &\le 0 \\
-z_{ik} + z_{ijk} &\le 0 \\
\left(z_{ij} + z_{ik}\right) - z_{ijk} &\le 1
\end{aligned}\right\} \quad ;\ i = 1,\ldots,m\ ,\ j = 1,\ldots,n-1, k = 2,3,\ldots,n, \text{and } k > j \tag{31.3}$$

(31.4)

(31.5)

$$z_{ij}, z_{ik}, z_{ijk} \in \{0,1\} \tag{31.6}$$

, where

$$z_{ij} = \begin{cases} 1, \text{if customer } j \text{ is assigned to facility } i \\ 0, \text{otherwise} \end{cases}$$

r_i is capacity of facility i ; $i = 1,\ldots,m$

w_j is demand of customer j ; $j = 1,\ldots,n$

$(a_j,\ b_j)$ is known co-ordinate of customer on plane.

$z_{ijk} = z_{ij}z_{ik}$; $i = 1,\ldots,m$, $j = 1,\ldots,n-1, k = 2,3,\ldots,n,$ and $k > j$

Observe from problem (31) that constraint set (31.5) is the least effective constraint. The experiments for solving the problems with and without the constraint set (31.5) were done with various numbers of facilities and customers and the results show that the solutions to both of them are equivalent. Therefore, the last constraints can be ignored to reduce constraints size when the problem is solved by using command "bintprog" in MATLAB. This constraint is constructed to force z_{ijk} to be one when both z_{ij} and z_{ik} are one. The value of z_{ijk} should be set at one if there is no restriction on it to gain a better objective function value. Therefore, if both z_{ij} and z_{ik} are one that allows z_{ijk} to be one, the value of z_{ijk} will be automatically one. As a result, the constraint (31.5) will be cut off and the constraints to control value of z_{ijk} will remain only constraint (31.3) and (31.4). Finally, problem (31) becomes:

$$\textit{Minimize} \quad \phi = -\sum_{i=1}^{m}\frac{1}{r_i}\left[\sum_{j=1}^{n} w_j^2\left(a_j^2+b_j^2\right)z_{ij} + 2\sum_{j=1}^{n-1}\sum_{k>j}^{n} w_j w_k\left(a_j a_k + b_j b_k\right)z_{ijk}\right] \tag{32}$$

$$\textit{subject to} \quad \sum_{i=1}^{m} z_{ij} = 1 \qquad ;\ j = 1,...,n$$

$$\sum_{j=1}^{n} z_{ij} w_j = r_i \qquad ;\ i = 1,...,m$$

$$\left.\begin{aligned} -z_{ij} + z_{ijk} &\le 0 \\ -z_{ik} + z_{ijk} &\le 0 \end{aligned}\right\} \qquad ;\ i = 1,...,m\ ,\ j = 1,...,n-1, k = 2,3,...,n, \textit{and}\ k > j$$

$$z_{ij}, z_{ik}, z_{ijk} \in \{0,1\}$$

$$\textit{, where} \quad z_{ij} = \begin{cases} 1, \text{if customer } j \text{ is assigned to facility } i \\ 0, \text{otherwise} \end{cases}$$

r_i is capacity of facility i ; $i = 1,...,m$

w_j is demand of customer j ; $j = 1,...,n$

(a_j, b_j) is known co-ordinate of customer on plane.

$z_{ijk} = z_{ij} z_{ik}$; $i = 1,...,m$, $j = 1,...,n-1, k = 2,3,...,n,$ and $k > j$

There are additional $mn(n-1)/2$ variables compared with problem (23) solved by the EPR method. The problem (32) with $mn(n+1)/2$ variables will be solved under branch-and-bound approach using command "bintprog" in MATLAB. The solutions

obtained from this method will be sequentially compared with ones obtained from the EPR method.

The Exact Algorithm (EA) and Empirical Exact Algorithm (EEA), of which flow diagram is shown in Figure 1, can be summarized step by step as follows.

Step 0: Set the list of vector *r* to be an empty set.

Step 1: Calculate lower bound and upper bound on r_i using technique shown in 2.1.1.

Step 2: Fix value of r_i at the first combination according to branching strategy described in section 2.1.2.

Step 3: Check if the current vector *r* repeats the existing vector *r* in lists: infeasible solution list and feasible solution list. If the current vector *r* repeats vector *r* in the lists, go to step 6. Otherwise, go to step 4.

Step 4: Solve QIP problem (32) or (23) by replacing all corresponding r_i's with the current vector *r* and using linearization method (shown in section 2.1.3.II) or EPR method (shown in section 2.1.3.I) in case of the algorithm is exact or empirical exact algorithm respectively.

Step 5: Keep vector *r* in feasible solution list or infeasible solution list if solution from step 4 is feasible or infeasible respectively. Keep the objective function value of the feasible solution in another list.

Step 6: Check if all possible combinations of vector *r* are considered. If yes, stop and the solution that gives the minimum value of objective function value is the optimal solution. Otherwise, move to the next combination of vector *r* and go back to step 3.

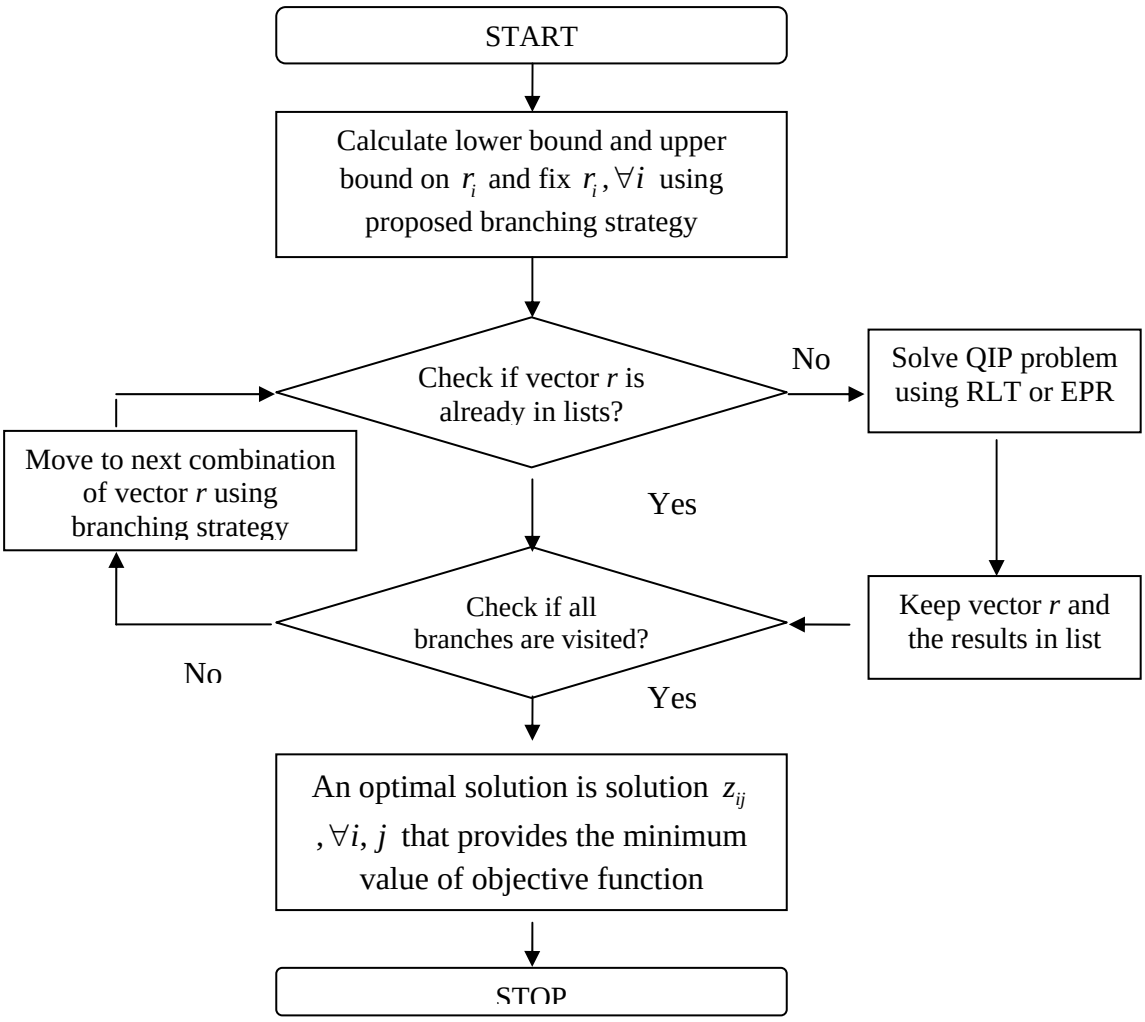


Figure 1 Flow diagram of Exact and Empirical Exact Algorithms

2.2 Heuristic Algorithm for CMLCP

The proposed heuristic method for problem (22) is called Heuristic Branch-and-Bound Algorithm (HBBA). It bases on the branch-and-bound approach for sum-of-ratios problem. Generally, branch-and-bound algorithm for sum-of-ratios problem starts with finding range of fractions or denominators to be branched. This range will be partitioned corresponding to branching strategy, which is usually rectangular partition. In each branch, the well-developed lower bound function will be solved and its solutions will be used to define which ratio will be cut or partitioned next. The next branch will be considered according to the branching rule, depth first

or best bound. Branching will be finished when lower bound is close to upper bound, which is obtained by replacing the solution to the lower bound function and to the original function. Finally, the branch-and-bound algorithm will stop if all determined branches are visited.

Under the same concept of branch-and-bound algorithm for sum of ratios problem, the HBBA starts with determining the range of the denominators. After that the iterative procedure of partitioning range of a certain denominator, solving problem (22) on the specific range, and updating the range of denominator to be considered next will proceed. And then, the algorithm will stop at the determined width of remaining range of denominators. Therefore, the best of all local optimal solutions will be the final solution. To compute the optimal solutions to problem (22) under global optimization approach, the lower bound function of problem (22) is constructed and then minimized by EPR method. The upper bound will be given by trust-region method provided the initial solution by EPR method from minimizing the lower bound function. The upper bound proposed here does not just come from replacing the solution of the lower bound function to the original objective function as it is usually done in general branch-and-bound algorithm for sum-of-ratios problem, so as to expedite the upper bound improvement rate. Using trust-region method, the local optimal solution corresponding to the initial solutions will be obtained. Observe that, the initial solutions to the trust-region method move when range of denominator is cut, and the upper bound is solved under the original range. The concept under this algorithm is to find the best local optimal solution in original range when considering the further neighborhood of the preceding initial solutions. If the further neighborhood is considered but the local optimal solution provided from trust-region method is still unchanged, it is believable that this local optimal solution may be the global one. The HBBA under this approach uses five main techniques as follows.

2.2.1 Determination of Initial Range of Denominators

This step is done in the same manner of finding lower bound and upper bound of $r_i; \forall i = 1,...,m$ described in section 2.1.1. The lower bound and upper

bound can be obtained from solving problem (25) and (26) respectively. Since this initial range is continuously cut, then the matrix of the range will be added some symbols as follows.

$$\begin{bmatrix} LBr_1^{(t)} & , & UBr_1^{(t)} \\ LBr_2^{(t)} & , & UBr_2^{(t)} \\ \vdots & & \vdots \\ LBr_m^{(t)} & , & UBr_m^{(t)} \end{bmatrix} \quad or \quad \begin{bmatrix} \underline{r_1}^{(t)} & , & \overline{r_1}^{(t)} \\ \underline{r_2}^{(t)} & , & \overline{r_2}^{(t)} \\ \vdots & , & \vdots \\ \underline{r_m}^{(t)} & , & \overline{r_m}^{(t)} \end{bmatrix}$$

, where $LBr_i^{(t)}$ and $UBr_i^{(t)}$ represent lower bound and upper bound on r_i at the t^{th} considered iteration respectively. And, $\underline{r_i}^{(t)} \le r_i \le \overline{r_i}^{(t)}$. For initial range, *t* is set at 0.

2.2.2 Branching Strategy

General method for branching variables is rectangular partitioning, which can be shown in Figure 2. Using this partitioning algorithm, range of one variable (which is component r_i) is cut at a time. In here, the component, say r_l, whose range will be cut is one with the maximum width of range at the considered iteration or $l = \arg\max_{i=1,\ldots,m} \left\{ \overline{r_i}^{(t-1)} - \underline{r_i}^{(t-1)} \right\}$. The reason behind this criterion is because of the convergence of rectangular partition. If the criterion is to cut one with the minimum width of range, the same component will be cut in every iteration and then only the range of that component will be converge.

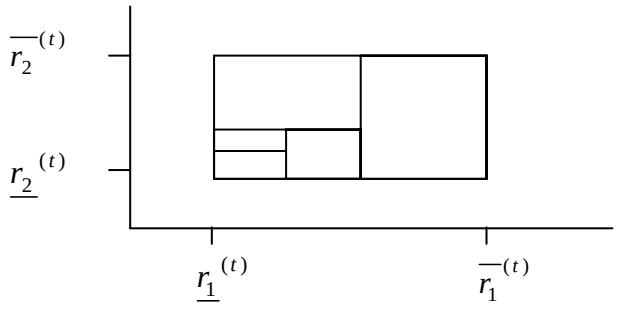


Figure 2 Rectangular partition

After the range of r_l is cut, the feasible range of vector r will be partitioned by plane $r_l^{(t)} = \frac{1}{2}(\underline{r_l}^{(t-1)} + \overline{r_l}^{(t-1)})$ into two mutually exclusive feasible ranges shown in matrix form below.

$$r^{(t,1)} \in \begin{bmatrix} \underline{r_1}^{(t-1)} & , & \overline{r_1}^{(t-1)} \\ : & & : \\ \underline{r_l}^{(t-1)} & , & \left\lfloor \frac{1}{2}\left(\underline{r_l}^{(t-1)} + \overline{r_l}^{(t-1)} \right) \right. \\ : & & : \\ \underline{r_m}^{(t-1)} & , & \overline{r_m}^{(t-1)} \end{bmatrix} \quad \text{and} \quad r^{(t,2)} \in \begin{bmatrix} \underline{r_1}^{(t-1)} & , & \overline{r_1}^{(t-1)} \\ : & & : \\ 1+\left\lfloor \frac{1}{2}\left(\underline{r_l}^{(t-1)} + \overline{r_l}^{(t-1)} \right) \right., & & \overline{r_l}^{(t-1)} \\ : & & : \\ \underline{r_m}^{(t-1)} & , & \overline{r_m}^{(t-1)} \end{bmatrix}$$

For feasible range $r^{(t,1)}$ the lower bound on component $r_i; \forall i = 1,...,m$ and $i \neq l$ will be tightened by using property of constraint set shown in equation (24). From equation (24), $\sum_{i=1}^{m} r_i = \sum_{j=1}^{n} w_j \rightarrow r_i + \sum_{\substack{k=1 \\ k \neq i}}^{m} r_k = \sum_{j=1}^{n} w_j$. Therefore, the lower bound on component r_i are revised to $\max\left\{ \underline{r_i}^{(t-1)}, \sum_{j=1}^{n} w_j - \sum_{\substack{k=1 \\ k \neq i}}^{m} \overline{r_k}^{(t-1)} \right\}$. This tightening is done because in partitioned feasible range $r^{(t,1)}$ the upper bound of r_l is cut. It affects to the lower bound of the other components $r_i; \forall i = 1,...,m$ and $i \neq l$, which sequentially affects the lower bound function of problem (22) shown in the next section. Since in feasible range $r^{(t,2)}$ the lower bound of r_l is cut which does not affect the lower bound on other components, then the lower bound of the other r_i' s will not be tightened.

2.2.3 Finding the Solution to Problem (22) on the Considered Range of *r*

According to the proof shown below, the objective function of the studied problem is neither pseudoconvex nor pseudoconcave function. As a result, the problem has multiple local minima, many of which fail to be globally optimal and no vertex of polytope S might provide a globally optimal solution to problem (22).

Proposition 3: The objective function of problem (22) is neither pseudoconvex nor pseudoconcave function.

Proof: Objective function value of problem (22) can be linearized as equation (31). So, each ratio can be reduced to ratio of two linear functions, which is pseudomonotonic function. According to Kuno (2005), the sum of pseudomonotonic (which is pseudoconvex or pseudoconcave) functions is neither pseudoconvex nor pseudoconcave function. Therefore, by reduction method, the proof is complete. ■

After passing the branching step, the original range will be cut into two mutually exclusive feasible regions. And then, the problem (22) corresponding to the partitioned range shown below will be solved.

$$\textit{Minimize} \quad f(r, z_{ij}) = -\sum_{i=1}^{m} \frac{1}{r_i}\left[(z_i{}^T wa)^2 + (z_i{}^T wb)^2 \right] = \frac{1}{2} z^T G z \tag{33}$$

$$\textit{subject to} \quad \sum_{i=1}^{m} z_{ij} = 1$$

$$\underline{r_i}^{(t)} \le r_i = \sum_{j=1}^{n} z_{ij} w_j \le \overline{r_i^{(t)}}$$

$$\textit{, where} \quad wa = [w_1 a_1, ..., w_n a_n]^T \; ; \forall\, j = 1, ..., n$$

$$wb = [w_1 b_1, ..., w_n b_n]^T \; ; \forall\, j = 1, ..., n$$

$$z_i = [z_{i1}, z_{i2}, ..., z_{in}]^T \; ; \forall\, i = 1, ..., m$$

$\underline{r_i}^{(t)}$ is lower bound of r_i on the considered range; $\forall\, i = 1, ..., m$

$\overline{r_i}^{(t)}$ is upper bound of r_i on the considered range; $\forall\, i = 1, ..., m$

Due to multiple locally optimal solutions of the problem, concept of global optimization is applied. To compute the optimal solution to the problem (33), the solutions to lower bound function shown below will be found.

$$\textit{Minimize} \quad \underline{q}(r, z_{ij}) = -\sum_{i=1}^{m} \frac{1}{\underline{r_i}^{(t)}}\left[(z_i{}^T wa)^2 + (z_i{}^T wb)^2 \right] \equiv \frac{1}{2} z^T \underline{G} z \tag{34}$$

$$\textit{subject to} \quad \sum_{i=1}^{m} z_{ij} = 1$$

$$\underline{r_i}^{(t)} \le \sum_{j=1}^{n} z_{ij} w_j \le \overline{r_i^{(t)}}$$

, *where* $wa = [w_1 a_1, ..., w_n a_n]^T \ ; \forall\ j = 1, ..., n$

$wb = [w_1 b_1, ..., w_n b_n]^T \ ; \forall\ j = 1, ..., n$

$z_i = [z_{i1}, z_{i2}, ..., z_{in}]^T \ ; \forall\ i = 1, ..., m$

$\underline{r_i}^{(t)}$ is lower bound of r_i on the considered range; $\forall\ i = 1, ..., m$

$\overline{r_i}^{(t)}$ is upper bound of r_i on the considered range; $\forall\ i = 1, ..., m$

Proof of proposition 4 shown below explains how problem (34) can provide a lower bound on problem (33).

Proposition 4: Problem (34) provides lower bound on problem (33) for all value of vector r in the considered range.

Proof: Recognize "for all" in the statement, "Choose method" is used to choose any vector *r* lying on the considered range. The remaining is to show that with this chosen vector *r*, the objective function value of problem (34) is less than the objective function value of problem (33).

Plugging the chosen vector *r* in the objective function of problem (33), the objective function will become $f(r, z_{ij}) = -\sum_{i=1}^{m} \frac{1}{r_i} \left[(z_i{}^T wa)^2 + (z_i{}^T wb)^2 \right]$. Subtracting $f(r, z_{ij})$ by $\underline{q}(r, z_{ij})$, the remaining is

$$f(r, z_{ij}) - \underline{q}(r, z_{ij}) = -\sum_{i=1}^{m} \frac{1}{r_i} \left[(z_i{}^T wa)^2 + (z_i{}^T wb)^2 \right] + \sum_{i=1}^{m} \frac{1}{\underline{r_i}} \left[(z_i{}^T wa)^2 + (z_i{}^T wb)^2 \right]$$

$$= \sum_{i=1}^{m} \left(-\frac{1}{r_i} + \frac{1}{\underline{r_i}} \right) \left[(z_i{}^T wa)^2 + (z_i{}^T wb)^2 \right]$$

$\left[(z_i{}^T wa)^2 + (z_i{}^T wb)^2 \right] > 0$, because all matrix, $z_i, wa, and\ wb$, compose of positive variables. According to the supply constraint set,

$$0 < \underline{r_i} \le r_i$$
$$-\underline{r_i} \ge -r_i$$
$$\frac{-1}{\underline{r_i}} \le \frac{-1}{r_i}$$
$$\frac{-1}{r_i} + \frac{1}{\underline{r_i}} \ge 0$$

Therefore, $f(r,z_{ij}) - \underline{q}(r,z_{ij}) \ge 0 \rightarrow f(r,z_{ij}) \ge \underline{q}(r,z_{ij})$ and then the proof is complete. ■

This lower bound will be solved by the proposed EPR method described in section 2.1.3.I. Since only a lower bound is required, then the EPR method is used to provide only the incumbent solution (exchanging method is excluded). Therefore, the aim is to solve the following related LIP problem of problem (34).

$$\begin{aligned} &\textit{Minimize} && \underline{g}(r,z) = \frac{1}{2}\underline{U}z && && (35)\\ &\textit{subject to} && \sum_{i=1}^{m} z_{ij} = 1 && ;\ j = 1,...,n \\ & && \underline{r_i}^{(t)} \le \sum_{j=1}^{n} z_{ij} w_j \le \overline{r_i}^{(t)} && ;\ i = 1,...,m \\ & && z_{ij} \in \{0,1\} && ;\ i = 1,...,m \text{ and } j = 1,...,n \end{aligned}$$

and, $\underline{U_c}$, which is the c^{th} column of matrix $\underline{U}$, can be found by solving

$$\begin{aligned} &\underline{U_c} = \textit{Minimize} && z^T \underline{G_c} && && (36)\\ &\textit{subject to} && \sum_{i=1}^{m} z_{ij} = 1 && ;\ j = 1,...,n \\ & && \underline{r_i}^{(t)} \le \sum_{j=1}^{n} z_{ij} w_j \le \overline{r_i}^{(t)} && ;\ i = 1,...,m \\ & && z_{ij} \in \{0,1\} && ;\ i = 1,...,m \text{ and } j = 1,...,n \end{aligned}$$

where, $\underline{G_c}$ *is the* c^{th} *column of* $\underline{G}$

$\underline{G}$ = Hessian matrix of objective function (34)

The logic based method can be also used in this case as follows. The z_{ij} is fixed to be zero, if $\bar{r}_i^{(t)} < w_j, \forall i, j$. The solutions to $\underline{g}(r, z_{ij})$ and $\underline{U_c}$ are then used as initial solutions to find upper bound by using trust-region method with the original problem (22). This method is called Multistart Trust-Region Method (MSTR) because the initial solutions for trust-region method will be changed when the procedure goes on. The current best solution is the solution from MSTR that provides the minimum objective function value. Observe that, the initial current best solution is obtained from solving problem (22) provided initial solutions by solving problem (34) at $t = 0$.

To understand the trust-region approach to optimization, consider the unconstrained minimization problem $f(x)$, where the function takes vector arguments and return scalars. Suppose the starting point is at a point x in n-space and there is a need to improve, i.e., move to a point with a lower function value. The basic idea is to approximate f with a simpler function q, which reasonably reflects the behavior of function f in a neighborhood, let say N, around the point x. This neighborhood is the trust region. A trial step s is computed by minimizing (or approximately minimizing) over N. This is the trust-region subproblem, $\min_s \{q(s); s \in N\}$. The current point is updated to be $x+s$, if $f(x+s) < f(x)$, otherwise, the current point remains unchanged and N, the region of trust, is shrunk and the trial step computation is repeated. In the standard trust-region method, the quadratic approximation q is defined by the first two terms of the Taylor approximation to f at x; the neighborhood N is usually spherical or ellipsoidal in shape. Mathematically the trust-region subproblem is typically stated

$$\min \left\{ \frac{1}{2} s^T H s + s^T g \text{ such that } \| Ds \| \leq \Delta \right\} \tag{37}$$

, where g is the gradient of f at the current point x, H is the Hessian matrix, D is a diagonal scaling matrix, Δ is a positive scalar, and $\|.\|$ is the 2-norm. Some efficient and effective algorithms exist for solving equation (37) typically involves the computation of a full eigensystem and a Newton based process can be applied to the

secular equation $\frac{1}{\Delta}-\frac{1}{s}=0$. Such algorithms provide an accurate solution to equation (37). However, they require time proportional to several factorizations of *H*. Therefore, for large-scale problems a different approach is needed. Several approximation and heuristic strategies, based on equation (37) have been proposed in many literatures. The optimization toolbox installed in MATLAB, which is applied in this research, is to restrict the trust-region subproblem to a two-dimensional subspace *S*. Once the subspace *S* has been computed, the work to solve equation (37) is trivial even if full eigenvalue/ eigenvector information is needed (since in the subspace, the problem is only two-dimensional). The two-dimensional subspace *S* is determined with the aid of a preconditioned conjugate gradient process described below. The toolbox in MATLAB assigns $S=(s_1,s_2)$, where s_1 is in the direction of the gradient g and s_2 is either an approximate Newton direction, i.e., a solution to $H\cdot s_2=-g$ or a direction of negative curvature, $s_2^T\cdot H\cdot s_2<0$. The philosophy behind this choice of S is to force global convergence by the steepest descent direction or negative curvature direction and achieve fast local convergence by the Newton step, when it exists (MATLAB manual, 2006).

The trust-region method used in this research is based on the method installed in MATLAB under "fmincon" command. Since this command is created to find a minimum of constrained nonlinear multivariable function, the provided (local) optimal solution may not be integer even the initial solution is integer. Thus, the additional nonlinear constraints to force z_{ij} to be 0 and 1, $z_{ij}(1-z_{ij})\le 0;\forall i,\forall j$, are added if the first trial of using command "fmincon" without these nonlinear constraints fail to give $z_{ij}(1-z_{ij})\le 10^{-6};\forall i,\forall j$. After upper bound and lower bound is solved, the range of r which provides a less objective function value of problem (22) is considered next. And, the other will be ignored.

2.2.4 Stopping Rules

The HBBA stops if at least one of these stopping criteria is satisfied.

- The convergence condition of rectangular partition. In other words, there is no feasible r in the considered range due to the tightness of range.
- There are no improvement on current best solution (the solution that provides minimum value of equation (22)) within q = 6 consecutive iterations.

The last criterion arises from the approach of HBBA described previously. Sequential cutting the feasible range of vector *r* and solving problem (33) with MSTR method is to provide the high quality further initial solution (or further neighborhood) from the ones of the preceding range for the trust-region method. If the current best solution remains the same even the initial solution has been further changed, it has a high possibility that the current best solution is an optimal solution. Therefore, to reduce the computational time, this stopping rule is set up.

2.2.5 Additional Options

Observe that if the HBBA is used, problem (33) with a certain changed feasible region of r has to be solved at least three times corresponding to three regions: original interval, and partitioned regions from the first cut. Moreover, problem (33) is subject to inequality constraints not equality constraints. This factor is a large disadvantage when *n* is a little bit larger than *m*. The worst case is $n-m=1$, which means that only one customer is left after the others are matched with *m* facilities. The range of *r* is too tight to waste the time cutting and solving problem (33). So, considering all possible branches of *r* provides a reduced computational time. Therefore, HBBA will be switched to EEA for this special case.

Another special situation comes from the assumption of non-redundant facility. This assumption means if the amount of the facility with a certain capacity is less than *m*, the demand of all *n* customers cannot be satisfied. Therefore, the m^{th} facility is a necessity. This special case is the problem, whose $\frac{n-m}{m-1}<1$. This

equation can be described as follows. One customer is allocated to one facility and there are $n-m$ remaining customers. These customers will control the width of feasible range *r* of $m-1$ facilities. So, the range of *r* is too tight and then it takes more time to sequentially cut range of *r*. However, the range is too loose to use EEA. Thus in this case only the initial current best solution will be found.

The HBBA can be summarized step-by-step as follows. The flow diagram of this algorithm is also expressed in Figure 3 below.

Step 1: If $n-m=1$, switch to EEA. Otherwise, set iteration index *t*=0 and determine the initial range of r_i using technique in section 2.2.1.

Step 2: Find the initial current best solution by solving problem (22) with MSTR on the initial range of *r* (range of *r* at *t*=0). If $\frac{n-m}{m-1}<1$, stop the final solution is the initial current best solution. Otherwise, go to step 3.

Step 3: Set $t = t+1$. Partition range of *r* using bisection method at component $l=\arg\max_{i=1,\ldots,m}\left\{\overline{r}_i^{(t-1)}-\underline{r}_i^{(t-1)}\right\}$ by plane $r_l^{(t)}=\frac{1}{2}(\underline{r}_l^{(t-1)}+\overline{r}_l^{(t-1)})$.

Step 4: Find solution to problem (22) corresponding to the partitioned range of $r^{(t)}$ using MSTR method.

Step 5: If one of the obtained upper bound is lower than the current best solution, the new current best solution is changed to the solution providing the minimum upper bound. If upper bound from solving problem (22) on range $r^{(t,1)}$ ($UB^{(t,1)}$) is lower than that from solving problem on range $r^{(t,2)}$ ($UB^{(t,2)}$), range $r^{(t,1)}$ is selected to be considered next. Otherwise, range $r^{(t,2)}$ is selected to be considered next.

Step 6: If at least one of the stopping rules is not satisfied, go back to step 3. Otherwise, stop. The current best solution is a final solution.

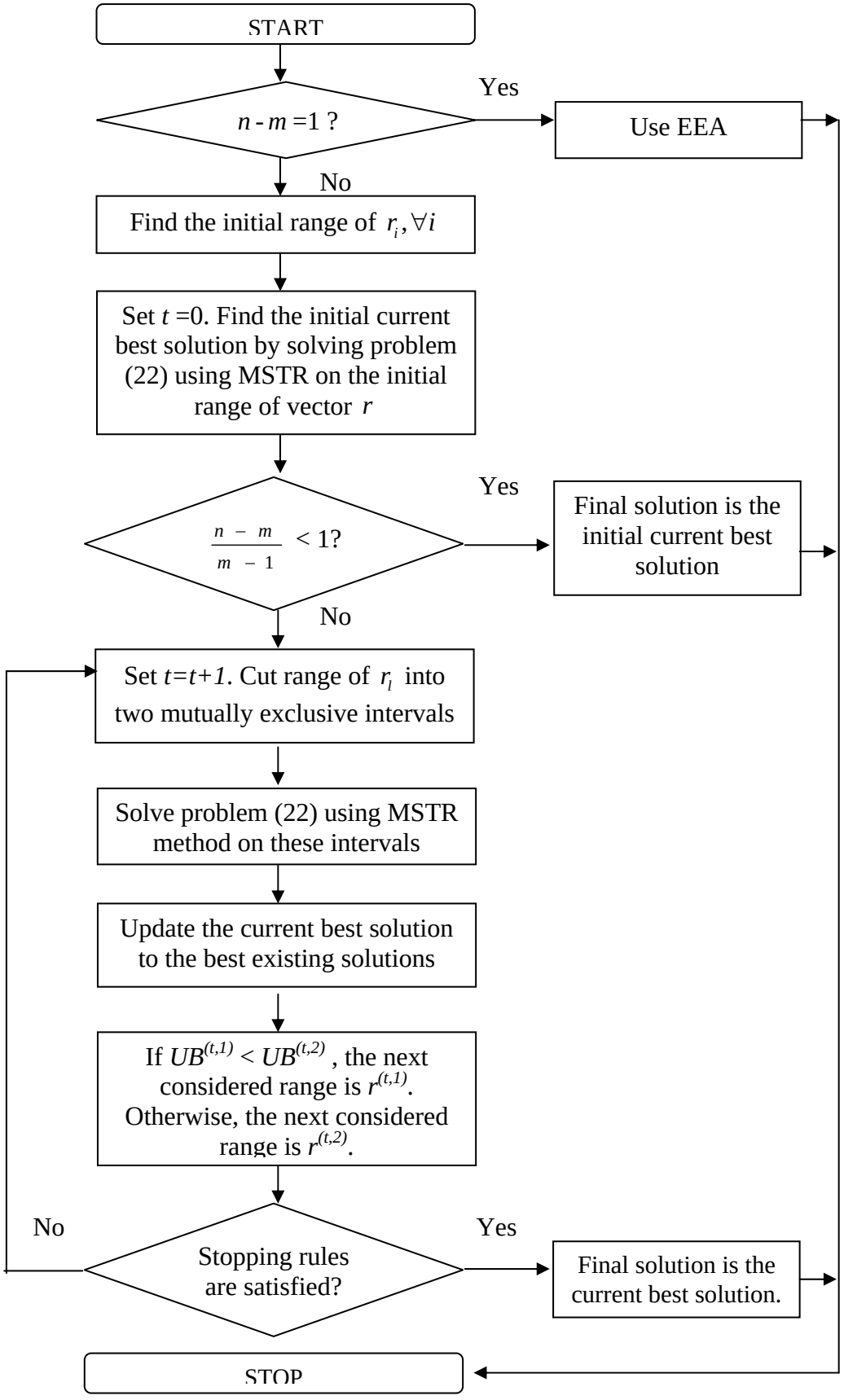


Figure 3 Flow Diagram of Branch-and-Bound Based Heuristics Algorithm

3. Numerical Experiments

The numerical experiments mainly compose of two sets. The first set is done to evaluate the EPR method, which will be applied in EEA to provide the reference solution for HBBA. The other is done to evaluate the proposed heuristic algorithm, HBBA, for CMLCP. They are well designed as follows.

3.1 Design of Numerical Experiments for Evaluating EPR Method

This set of numerical experiment is very important because the EPR method will be later used in EEA to provide the reference solution. So, the main objective of this numerical experiment is to statistically ensure that EPR method will provide the accurate solutions in any problem size of problem (23). The numerical experiments will be done on various problem sizes each of which composes of 10 to 100 data sets depending on the restriction of linearization method. And then, the solutions of EPR method will be evaluated by the solutions of linearization method explained in section 2.1.3.II by analyzing the confidence interval on the percentage of error calculated as follows.

$$\% \text{ of error} = \frac{\text{OFV of EPR method - OFV of linearization method}}{\text{OFV of linearization method}} \times 100$$

,where the word “OFV” abbreviates from objective function value.

EPR method and MSTR method involve re-computing Hessian matrix. For the large problem, Hessian matrix construction using calculus consumes very large computational time. To reduce the large computational time, the algebraic method to construct Hessian matrix mentioned in section 2.1.3.I.A is proposed. So, before doing the experiments to evaluate EPR method, another small set of experiments to test the effect of Hessian construction time is conducted. Because only roughly approximation of trend of Hessian construction time is required, there are thirteen problem sizes generated, each of which composes of five sets of data.

The method to generate input data such as $w_j, a_j, b_j, and\ r_i$, which are the demand for products, the *x*-coordinate, the *y*-coordinate of customers, and the capacity of facility (because in this case $r_i = s_i$) can be summarized as follows.

3.1.1 Generating Data of Customers

The Customer data w_j, a_j, and b_j are randomly generated using MATLAB program. Under the command "rand" in MATLAB program with certain seed numbers, the customer data will be generated in uniform distribution pattern, which will be then rounded up to be an integer. These seed numbers can vary to reflect the more realistic problem. For example, if the customers' locations can be covered by 100x200 units^2 plane, the seed numbers for generating a_j and b_j should be 100 and 200 respectively. Likewise, if the customers' demands vary not over 50 lots, the seed number for generating w_j, will be set at 50.

3.1.2 Generating Capacity of Facility

Value of $r_i; \forall i = 1,...,m$ has to be carefully generated because of the clustering problem. Using only command "rand" may derive the infeasible problem. We proposed the method to generate this data as follows. The procedure to generate r_i is based upon the fact that r_i arises from clustering the customers. Firstly, the amount of customers assigned for each facility will be randomly generated. Then, customers will be randomly picked up until the amount of customers in the facility is reached. Therefore, r_i is the summation of all w_j in cluster *i*.

3.2 Design of Numerical Experiments for Evaluating HBBA

This set of experiments is done to evaluate HBBA with EEA. EEA can be viewed as iteratively using EPR method to solve problem (23), which the capacity is changed according to the branching strategy. There are 31 problem sizes created and

each of which composes of 5 to 50 cases depending on EEA computational effort. The customers' input data for this set of experiments is generated in the same way as that of the experiments in 3.1.1. The main difference is to generate the facility capacity s_i .

In this set of experiments, $r_i \le s_i$ has to be also carefully generated because of the clustering problem. To generate s_i , r_i constructed in the same way as shown in 3.1.2 is used as a lower bound on s_i to ensure the feasible capacity. The feasible capacity means the capacity of facility which does not make customers' product to be separated. The upper bound on s_i under non-redundant facility assumption can be found by maximizing $\sum_{i=1}^{m} s_i$, with all $m-1$ combinations of s_i are not over the sum of customers' demand ($\sum_{j=1}^{n} w_j$) in order that the m^{th} facility is required or non-redundant facility. In other words, it can be written in mathematical model as follows.

$$
\begin{aligned}
&\textit{Maximize} \quad \sum_{i=1}^{m} s_i \\
&\textit{subject to} \quad \sum_{\substack{k=1 \\ k \neq i}}^{m} s_k \le \sum_{j=1}^{n} w_j - 1 \quad ; \; i = 1, \ldots, m \\
&\qquad\qquad s_i \ge r_i \qquad\qquad ; \; i = 1, \ldots, m
\end{aligned}
\tag{38}
$$

Now feasible range of capacity s_i , which is the value between r_i and the solution s_i obtained from solving problem (38) , is constructed. The feasible s_i will be randomly selected from this range. That is $s_i = r_i + \delta_i$, where δ_i is random integer number running from one to (s_i obtained from solving problem (38) $- r_i$).

4. Validation of the Solutions

To validate the solutions from HBBA, the confidence interval of percentage of error calculated as follows will be analyzed.

$$\% \text{ of error} = \frac{\text{OFV of HBBA - OFV of EEA}}{\text{OFV of EEA}} \times 100$$

The word "OFV" abbreviates from the objective function value. The confidence interval (CI) is provided by SPSS. It can be calculated as follows. A 100(1-α)% CI on the mean is

$$\bar{x} \pm t_{\frac{\alpha}{2},n-1} \frac{S}{\sqrt{n}}$$

,where S = standard deviation of sample

n = number of sample

$t_{\frac{\alpha}{2},n-1}$ = the percentage point of the *t distribution* with n-1 degrees of freedom such that $P\left\{t_{n-1} \geq t_{\frac{\alpha}{2},n-1}\right\} = \frac{\alpha}{2}$

The above formula will be used to analyze in all problem sizes and to study the effect of problem size on percentage of error. Also, it will be used to analyze the effectiveness of HBBA in all cases.

RESULTS AND DISCUSSION

This part shows the results of numerical experiments and discussion on these results. It composes of two sections as follows. To ensure EPR method before being applied in EEA, the first section expresses the results of and discussion on EPR method. The effect of the Hessian construction time on processing time is also illustrated. The confidence interval of percentage of error of EPR method compared with linearization method are also shown and analyzed. For the next section, the results of HBBA, processing time and confidence interval of percentage of error of HBBA compared with EEA, and discussion on these results are shown.

1. Results of and Discussion on EPR method

This part begins with reporting the effect of Hessian matrix construction time on processing time. And then, the remaining section reports the efficiency and effectiveness of EPR method compared with the linearization method.

1.1 Effect of Hessian Matrix Construction Time on Processing Time

Firstly, EPR method using original calculus to construct Hessian matrix G is analyzed. The 13 problem sizes each of which composes of five data sets of problem (23), which is a specific problem of problem (22) when $r_i = s_i$ or subject to balanced transportation constrains, with $m = 2$ and n varies from 5 to 120 are randomly generated. They are generated using the method described in section 3.1.1 and 3.1.2. The processing time, time used for the whole process, is measured by separating into two parts. The first part represents Hessian construction time. The other represents execution time, which is time used to solve problem (23). The processing time can be shown as the following table.

Observe from Table 1 that time used to construct Hessian matrix is still less than execution time if the number of variables is less than 150 variables. For

problems whose number of variables is more than or equal 150 variables, most of process time are devoted to constructing Hessian matrix.

Table 1 Effect of Hessian matrix construction time on processing time.

Problem no.	Number of variables = mxn	<1> Hessian Construction Time (sec)	< 2 > Execution Time (sec)	Processing Time = <1> + <2> (sec)
1	10	0.078	0.238	0.316
2	20	0.216	1.822	2.038
3	30	0.450	4.853	5.303
4	40	0.931	7.303	8.234
5	50	1.828	9.070	10.898
6	80	15.563	57.898	73.461
7	100	42.703	83.602	126.305
8	130	144.797	316.313	461.109
9	150	274.469	225.625	500.094
10	180	547.766	419.609	967.375
11	200	928.359	294.241	1222.600
12	220	1445.600	534.800	1980.400
13	240	2075.900	565.300	2641.200

Later, the EPR method using algebraic method stated in section 2.1.3 I.A is used with the same set of input data and the processing time measured as above method is recorded. The results show that the same solutions are provided with less processing time. Hessian construction time for all problem sizes is reduced to zero. With benefits from the proposed algebraic method, most of the processing time is devoted to solving the problem and then the processing time is equivalent to execution time. The relationship of the processing time of EPR method using both calculus (reported in Table 1) and algebraic method to construct Hessian matrix and number of variables $(m \times n)$ are shown in Figure 4.

In Figure 4, the first three labels represent processing time, Hessian construction time, and execution time of EPR method using calculus provided by command “jacobian” in MATLAB to construct Hessian matrix. The other represent the processing time of EPR method using the proposed method to construct Hessian matrix. Since the Hessian construction time using this method is much less, then most of processing time is execution time. Therefore, processing time is merely presented.

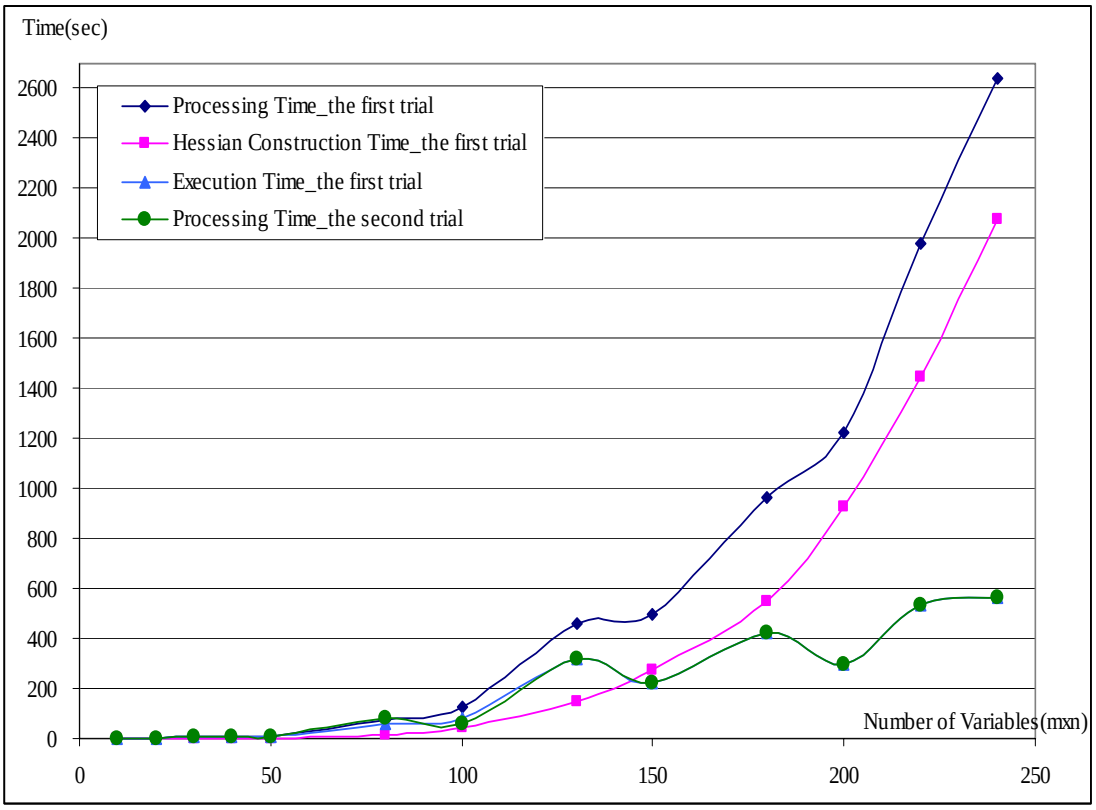


Figure 4 Time comparisons of using two methods to construct Hessian matrix.

Figure 4 shows that even the problem size grows, time to construct Hessian matrix by using algebraic method does not appear. Thus, the main processing time is used only for solving problem and it seems to be equal to execution time of using calculus to construct Hessian matrix. On contrary, time to construct Hessian matrix by using calculus increases rapidly as the problem size grows. Using algebraic method to construct Hessian matrix is very efficient. As a result, this approach will be used to develop Hessian matrix from now on.

1.2 The Results of Comparing EPR Method with Linearization Method

To evaluate EPR method the following 41 problem sizes, of which amount of variables $m \times n$ is not over 1000 variables, of problem (23) are generated.

Table 2 Problem sizes of numerical experiments used to evaluate EPR method.

Problem No.	Problem Size		Problem No.	Problem Size		Problem No.	Problem Size	
	m	n		m	n		m	n
1	2	5	15	3	17	29	5	20
2	2	10	16	3	20	30	5	200
3	2	15	17	3	25	31	6	7
4	2	20	18	4	6	32	6	9
5	2	30	19	4	8	33	6	12
6	2	40	20	4	9	34	6	15
7	2	50	21	4	10	35	7	9
8	2	60	22	4	15	36	7	11
9	2	200	23	4	20	37	7	12
10	2	500	24	4	25	38	8	10
11	3	5	25	5	8	39	8	11
12	3	8	26	5	9	40	8	12
13	3	10	27	5	10	41	20	50
14	3	11	28	5	15			

They vary from $m = 2$ to 20 and $n = 5$ to 500. For each problem size, 10 - 100 sets of data are created. The amount of data sets depends on the computational time of linearization method. And then, these problems are solved by EPR method and linearization method, both of which are coded by MATLAB program and use command “bintprog” at solving step.

Before summarizing the results, two examples of using EPR method are shown as follows to illustrate principle of this method in practice. The first example shows the detail of calculation while the second example shows the mechanism in ranking the solution to problem P2 to obtain the solution to problem P1.

Example 1 A randomly generated problem: $m = 2,\ n = 5$ has the following input data

$$
\begin{aligned}
w_j &= 10,\ 25,\ 27,\ 30,\ 4 \quad ;\ j = 1,...,5 \\
a_j &= 2,\ 13,\ 41,\ 21,\ 33 \quad ;\ j = 1,...,5 \\
b_j &= 42,\ 35,\ 1,\ 24,\ 34 \quad ;\ j = 1,...,5 \\
r_i &= 35,\ 61 \qquad\qquad\quad ;\ i = 1\ and\ 2
\end{aligned}
$$

These input data are used to formulate the following model.

$$
\text{P1: } \underset{z_{ij} \in S}{Minimize}\ f(z) = -\left\{ \sum_{i=1}^{2} \frac{1}{r_i} \left[(z_i{}^T wa)^2 + (z_i{}^T wb)^2 \right] \right\} \equiv -\sum_{i=1}^{2} \frac{1}{r_i} \left[z_i{}^T H z_i \right] \equiv \frac{1}{2} z^T G z
$$

$$, where \ \ S = \left\{ \ \sum_{i=1}^{2} z_{ij} = 1, \ \sum_{j=1}^{5} z_{ij} w_j = r_i, z_{ij} \in \{0,1\}, i = 1 \ and \ \ 2, \ and \ j = 1,...,5 \ \right\}$$

$$z = \begin{bmatrix} z_{11} & z_{21} & z_{12} & z_{22} & z_{13} & z_{23} & z_{14} & z_{24} & z_{15} & z_{25} \end{bmatrix}^T$$

And, the symmetry matrix *G* obtained by algebraic method is as follows.

$$G = -10^4 \times \begin{bmatrix} 1.0103 & 0 & 2.1371 & 0 & 0.1913 & 0 & 1.8000 & 0 & 0.3415 & 0 \\ 0 & 0.5797 & 0 & 1.2262 & 0 & 0.1098 & 0 & 1.0328 & 0 & 0.1959 \\ 2.1371 & 0 & 4.9786 & 0 & 2.1909 & 0 & 4.7700 & 0 & 0.9251 & 0 \\ 0 & 1.2262 & 0 & 2.8566 & 0 & 1.2570 & 0 & 2.7369 & 0 & 0.5308 \\ 0.1913 & 0 & 2.1909 & 0 & 7.0067 & 0 & 4.0963 & 0 & 0.8560 & 0 \\ 0 & 0.1098 & 0 & 1.2570 & 0 & 4.0203 & 0 & 2.3503 & 0 & 0.4911 \\ 1.8000 & 0 & 4.7700 & 0 & 4.0963 & 0 & 5.2303 & 0 & 1.0347 & 0 \\ 0 & 1.0328 & 0 & 2.7369 & 0 & 2.3503 & 0 & 3.0010 & 0 & 0.5937 \\ 0.3415 & 0 & 0.9251 & 0 & 0.8560 & 0 & 1.0347 & 0 & 0.2053 & 0 \\ 0 & 0.1959 & 0 & 0.5308 & 0 & 0.4911 & 0 & 0.5937 & 0 & 0.1178 \end{bmatrix}$$

$U_1 = \min_{z_{ij} \in S} z^T G_1$, where $G_1 = -10^4 \times \begin{bmatrix} 1.0103 & 0 & 2.1371 & 0 & 0.1913 & 0 & 1.8000 & 0 & 0.3415 & 0 \end{bmatrix}^T$ = -3.1474x10^4.

Likewise, $U_2, ..., U_{10}$ x10^4 can be found as follows.

$U_2 = \min_{z_{ij} \in S} z^T G_2$ =-1.3385 $\quad$ $U_5 = \min_{z_{ij} \in S} z^T G_5$ =-2.3822 $\quad$ $U_8 = \min_{z_{ij} \in S} z^T G_8$ = -5.9450

$U_3 = \min_{z_{ij} \in S} z^T G_3$ = -7.1157 $\quad$ $U_6 = \min_{z_{ij} \in S} z^T G_6$ =-6.8617 $\quad$ $U_9 = \min_{z_{ij} \in S} z^T G_9$ = -1.2666

$U_4 = \min_{z_{ij} \in S} z^T G_4$ = -4.5248 $\quad$ $U_7 = \min_{z_{ij} \in S} z^T G_7$ =-6.5700 $\quad$ $U_{10} = \min_{z_{ij} \in S} z^T G_{10}$ = -1.2026

And then the matrix *U* is as follows.

$U = -10^4 \times [3.1474 \quad 1.3385 \quad 7.1157 \quad 4.5248 \quad 2.3822 \quad 6.8617 \quad 6.5700 \quad 5.9450 \quad 1.2666 \quad 1.2026]$.

Therefore, the optimal solution for $P2: \underset{z_{ij} \in S}{Minimize} \ g(z) = \frac{1}{2} Uz$, which is the lower bound function of problem P1, is z_0 = [1 0 1 0 0 1 0 1 0 1]^T with $g_0(z) = -1.2136 \times 10^5$. After replacing z_0 in $f(z)$, the upper bound is obtained, which is $f(z_0) = -1.2136 \times 10^5$.

Observe that $g_0(z) \geq f(z_0)$, then z_0 is an optimal solution by proposition 2. The optimal solution z_0 can be decomposed by facility as $z_1 = [1\ 1\ 0\ 0\ 0]^T$, and $z_2 = [0\ 0\ 1\ 1\ 1]^T$. This means that the optimal allocation of customer is as follows. There are two customers, who are customer 1 and 2, served by facility 1. And, there are three customers, who are customer 3, 4 and 5, served by facility 2. The optimal location of facility 1 and 2 can be calculated by using single facility location problem as follows.

Optimal location of facility 1: $x_1 = \dfrac{\sum_{j=1}^{5} z_{1j} w_j a_j}{\sum_{j=1}^{5} z_{1j} w_j} = 9.8571$, $y_1 = \dfrac{\sum_{j=1}^{5} z_{1j} w_j b_j}{\sum_{j=1}^{5} z_{1j} w_j} = 37.0000$

Optimal location of facility 2: $x_2 = \dfrac{\sum_{j=1}^{5} z_{2j} w_j a_j}{\sum_{j=1}^{5} z_{2j} w_j} = 30.6393$, $y_2 = \dfrac{\sum_{j=1}^{5} z_{2j} w_j b_j}{\sum_{j=1}^{5} z_{2j} w_j} = 14.4754$

$$\text{The total transporation distance} = f(z) + \text{Constant term} = f(z) + \sum_{j=1}^{n} w_j (a_j^{\,2} + b_j^{\,2})$$

$$= -1.2136 \text{x} 10^5 + 1.37432 \text{x} 10^5 = 1.6072 \text{x} 10^4$$

The optimal locations of the two facilities and the optimal allocation of five customers to these facilities can be shown as the following figure.

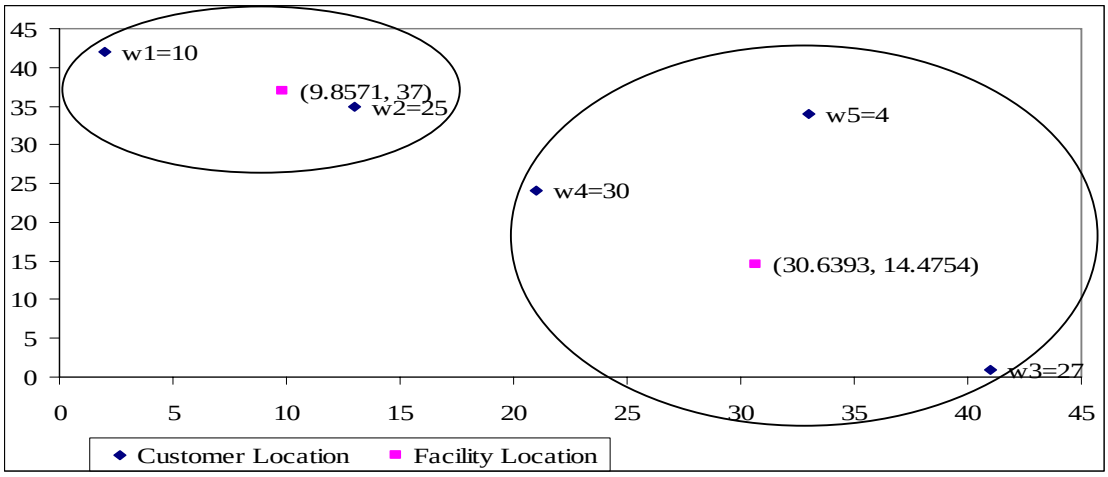


Figure 5 The optimal locations and allocations to the problem in Example 1.

Example 2 A problem m=2, n=10, which is used to verify proposition 2 and illustrate the proposed ranking method, is specially constructed. The location data is randomly generated while the weight is intentionally created to have a lot of adjacent vertices to show the ranking procedure step by step. The input data is as follows.

w_j = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ; $j = 1,...,10$

a_j = 54, 37, 10, 119, 84, 59, 87, 118, 23, 55 ; $j = 1,...,10$

b_j = 85, 53, 8, 64, 55, 6, 10, 19, 140, 146 ; $j = 1,...,10$

r_i = 10, 45 ; $i = 1 \; and \; 2$

It can be mathematically formulated as following model.

$$\text{P1: } \underset{z_{ij} \in S}{Minimize}\ f(z) = -\left\{ \sum_{i=1}^{2} \frac{1}{r_i} \left[(z_i{}^T wa)^2 + (z_i{}^T wb)^2 \right] \right\} \equiv -\sum_{i=1}^{2} \frac{1}{r_i} \left[z_i{}^T H z_i \right] \equiv \frac{1}{2} z^T G z$$

$$, where \; S = \left\{ \sum_{i=1}^{2} z_{ij} = 1, \; \sum_{j=1}^{10} z_{ij} w_j = r_i, z_{ij} \in \{0,1\}, i = 1 \; and \; 2, \; and \; j = 1,...,10 \right\}$$

$$z = \left[z_{11} \;\; z_{21} \;\; z_{12} \;\; z_{22} \;\; z_{13} \;\; z_{23} \;\; z_{14} \;\; z_{24} \;\; z_{15} \;\; z_{25} \;\; z_{16} \;\; z_{26} \;\; z_{17} \;\; z_{27} \;\; z_{18} \;\; z_{28} \;\; z_{19} \;\; z_{29} \;\; z_{1,10} \;\; z_{2,10} \right]^T$$

The symmetry matrix G and also row vector U of problem $P2: \underset{z_{ij} \in S}{Minimize}\ g(z) = \frac{1}{2} Uz$ can be obtained by the same method shown in Example 1. The provided initial solution gives $g_0(z) = -6.55850 \times 10^5$ and $f(z_0) = -5.80 \times 10^5$, which $g_0(z) < f(z_0)$. Thus, the vertices around the initial solution have to be explored to find the incumbent solution, which is the vertex that provides the minimum objective function value. After exploring, the incumbent solution is the vertex that gives $g(z_0^*) = -5.8302 \times 10^5$ and $f(z_0^*) = -5.80 \times 10^5$. Observe that $g(z_0^*) < f(z_0^*)$. The next adjacent vertices around the incumbent solution will be explored next. The value of $g(z)$ and $f(z)$ for each adjacent extreme points are listed in non-decreasing order of $g(z)$ and shown in Table 3. The current best solution or current upper bound is $f_u = f(z_0^*) = -5.80 \times 10^5$ and current lower bound is $f_l = g(z_0^*) = -5.8302 \times 10^5$. The optimal solution is seen to be extreme point 9 because $f_l = g(z_9) = -5.76 \times 10^5 > -5.80 \times 10^5 = f_u$, which is the optimal condition according to proposition 2. The solution, which is decomposed by

facility is z_1=[0 0 0 0 0 0 0 0 0 1]^T and z_2=[1 1 1 1 1 1 1 1 1 0]^T with $f(z) = -5.80 \times 10^5$. The total transportation distance is then -5.80x10^5 + constant term = 1.75720 x10^5. The solution shows that facility 1 has only one customer, who is customer 10, while the other customers are served by facility 2. The optimal location of facility 1 is obviously location of customer 10: $(x_1, y_1) = (55,146)$ and the optimal location of facility 2 calculated by the same method as shown in Example 1 is $(x_2, y_2) = (70.4, 50.311)$.

Table 3 The value of $g(z)$ and $f(z)$ for each adjacent extreme point of Example 2.

Extreme Point	$g(z)$	$f(z)$	f_l	f_u
1	-6.53E+05	-5.79E+05	-5.83E+05	-5.80E+05
2	-5.93E+05	-5.39E+05	-5.83E+05	-5.80E+05
3	-5.93E+05	-5.13E+05	-5.83E+05	-5.80E+05
4	-5.92E+05	-5.25E+05	-5.83E+05	-5.80E+05
5	-5.86E+05	-5.15E+05	-5.83E+05	-5.80E+05
6	-5.85E+05	-5.25E+05	-5.83E+05	-5.80E+05
7	-5.82E+05	-5.80E+05	-5.82E+05	-5.80E+05
8	-5.82E+05	-5.45E+05	-5.82E+05	-5.80E+05
9	-5.76E+05	-5.80E+05	-5.76E+05	-5.80E+05

The optimal location of the two facilities and the optimal allocation of 10 customers to these facilities can be shown as the following figure.

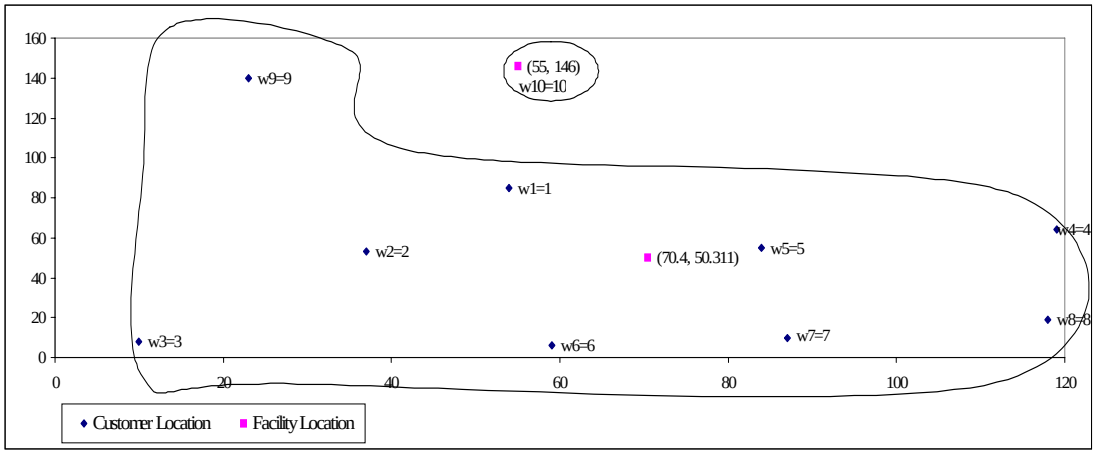


Figure 6 The optimal locations and allocations to the problem in Example 2.

This method and linearization method will be used for all problem sizes of numerical experiments. The processing time and the percentage of the difference between objective function value of EPR method and linearization method for all cases are collected, but only averages of them will be reported. The percentage of error is calculated as follows:

$$\% \text{ of error} = \frac{\text{OFV of EPR method- OFVof linearization method}}{\text{OFVof linearization method}} \times 100$$

, where OFV abbreviates from the objective function value. Since the number of variables of linearization method grows nonlinearly when *n* increases, there are some cases taking too much processing time. Therefore, the cases whose number of variable corresponding to the linearization method = $m \times n \times (n+1)/2$ is over 400 will be premature terminated by time limitation. The level of time limitation is determined as follows:

Number of variables	Premature Termination Time
$400 \le m \times n \times (n+1)/2 \le 750$	: 5 hours = 18,000 seconds
$750 < m \times n \times (n+1)/2 \le 1000$	: 8 hours = 28,800 seconds
$1000 < m \times n \times (n+1)/2$	: 24 hours = 86,400 seconds

The average processing time of both methods and average percentage of error of EPR method are summarized in Table 4. In Table 4, the blank cells represent the unsolvable cases due to number of variables over limitation of command “bintprog”, while underlined values shows that there are some premature terminated cases in these problem sizes. And 10 data sets are tested for each problem size of these cases, while 100 data sets are tested for other problem sizes. Processing time and percentage of error of EPR method are separately measured into three parts: processing time to obtain an initial solution (z_0), an incumbent solution (z_0^*), and the final solution ($z_r; z \in T^k$), so that the improvement rate of solution can be observed.

Table 4 Processing time of Linearization and EPR method and percentage of error of EPR Method.

Problem No.	Problem Size		Linearization Method		EPR Method						
						Initial Solution		Incumbent Solution		Final Solution		
	m	n	Variables mn(n+1)/2	Processing Time (sec)	No. of Variables (mxn)	Processing Time (sec)	Average Error (%)	Processing Time (sec)	Average Error (%)	Processing Time (sec)	Average Error (%)	n/m
1	2	5	30	0.17	10	0.15	0.00	0.15	0.00	0.15	0.00	2.50
2	2	10	110	28.75	20	2.39	7.38	2.39	0.00	2.39	0.00	5.00
3	2	15	240	1190.80	30	10.46	0.65	10.46	0.00	10.46	0.00	7.50
4	2	20	420	8886.67	40	19.98	-16.67	19.98	-26.16	20.01	-26.28	10.00
5	2	30	930	14601.76	60	63.24	-34.35	63.24	-41.66	63.27	-41.78	15.00
6	2	40	1640	36856.08	80	107.08	-39.52	107.09	-53.79	107.13	-53.79	20.00
7	2	50	2550	46024.80	100	617.22	-48.30	617.23	-52.28	617.41	-52.38	25.00
8	2	60	3660	-	120	902.01	-	902.04	-	903.11	-	30.00
9	2	200	40200	-	400	3045.44	-	3045.67	-	3052.12	-	100.00
10	2	500	250500	-	1000	38540.15	-	38547.10	-	38555.76	-	250.00
11	3	5	45	0.45	15	0.97	39.14	0.99	0.00	1.00	0.00	1.67
12	3	8	108	32.29	24	23.02	2.34	23.05	0.00	23.06	0.00	2.67
13	3	10	165	384.17	30	24.65	9.55	24.65	0.00	24.65	0.00	3.33
14	3	11	198	1543.61	33	31.01	26.50	31.02	7.30	31.02	0.00	3.67
15	3	17	459	9069.11	51	253.14	-38.75	253.24	-42.92	253.54	-43.54	5.67
16	3	20	630	11583.40	60	1841.88	-47.69	1841.94	-53.72	1842.87	-54.96	6.67
17	3	25	975	20469.33	75	2450.39	-32.86	2460.19	-37.90	2461.12	-38.04	8.33
18	4	6	84	2.51	24	1.01	4.50	1.02	0.00	1.02	0.00	1.50
19	4	8	144	57.54	32	15.01	2.61	15.01	0.00	15.02	0.00	2.00
20	4	9	180	239.49	36	40.45	10.22	40.45	0.00	40.45	0.00	2.25
21	4	10	220	3978.41	40	162.11	37.92	162.13	0.00	162.14	0.00	2.50
22	4	15	480	9874.46	60	2288.17	-54.80	2288.32	-56.17	2289.09	-56.17	3.75
23	4	20	840	21146.07	80	4530.30	-45.64	4530.31	-48.03	4531.73	-48.03	5.00
24	4	25	1300	49369.65	100	5471.45	-45.49	5471.98	-49.12	5473.58	-49.12	6.25

Table 4 (Continued)

Problem No.	Problem Size		Linearization Method		EPR Method							
						Initial Solution		Incumbent Solution		Final Solution		
	m	n	No. of Variables mn(n+1)/2	Processing Time (sec)	No. of Variables (mxn)	Processing Time (sec)	Average Error (%)	Processing Time (sec)	Average Error (%)	Processing Time (sec)	Average Error (%)	n/m
25	5	8	180	59.90	40	17.80	10.43	17.81	0.00	17.81	0.00	1.60
26	5	9	225	1345.10	45	220.35	0.00	220.35	0.00	221.16	0.00	1.80
27	5	10	275	1997.01	50	267.08	27.87	267.09	0.00	268.01	0.00	2.00
28	5	15	600	15577.95	75	4232.44	-43.35	4232.65	-43.99	4233.94	-43.99	3.00
29	5	20	1050	49591.39	100	6537.17	-36.44	6538.12	-37.97	6539.19	-37.97	4.00
30	5	200	100500	-	1000	28342.19	-	28344.97	-	28351.11	-	40.00
31	6	7	168	7.64	42	3.29	25.36	3.29	0.00	3.29	0.00	1.17
32	6	9	270	193.59	54	173.91	17.65	173.91	0.00	174.17	0.00	1.50
33	6	12	468	5471.55	72	859.23	0.00	859.43	0.00	859.46	0.00	2.00
34	6	15	720	15597.23	90	5711.84	-41.59	5711.85	-43.03	5711.89	-43.03	2.50
35	7	9	315	147.56	63	39.81	5.72	39.82	0.00	39.84	0.00	1.29
36	7	11	462	2146.66	77	601.27	-18.28	602.97	-18.62	603.97	-18.62	1.57
37	7	12	546	6497.42	84	742.01	-34.44	743.21	-35.21	745.07	-35.21	1.71
38	8	10	440	209.34	80	79.24	9.01	79.24	0.00	79.24	0.00	1.25
39	8	11	528	4998.91	88	1653.44	-26.99	1653.49	-26.99	1654.13	-26.99	1.38
40	8	12	624	7848.46	96	1821.29	-21.74	1825.29	-28.39	1831.97	-28.39	1.50
41	20	50	25500	-	1000	9863.45	-	9887.45	-	9923.95	-	2.50
99.99 % confidence interval of average percentage of error						-10.83 ± 17.91 %		-19.129 ± 14.42 %		-19.40 ± 14.34 %		

To obviously compare processing time of the two methods, Figure 7 is created to show the processing time for all problem sizes that both methods can solve. The processing time of EPR method in this figure is measured up to obtaining the final solution.

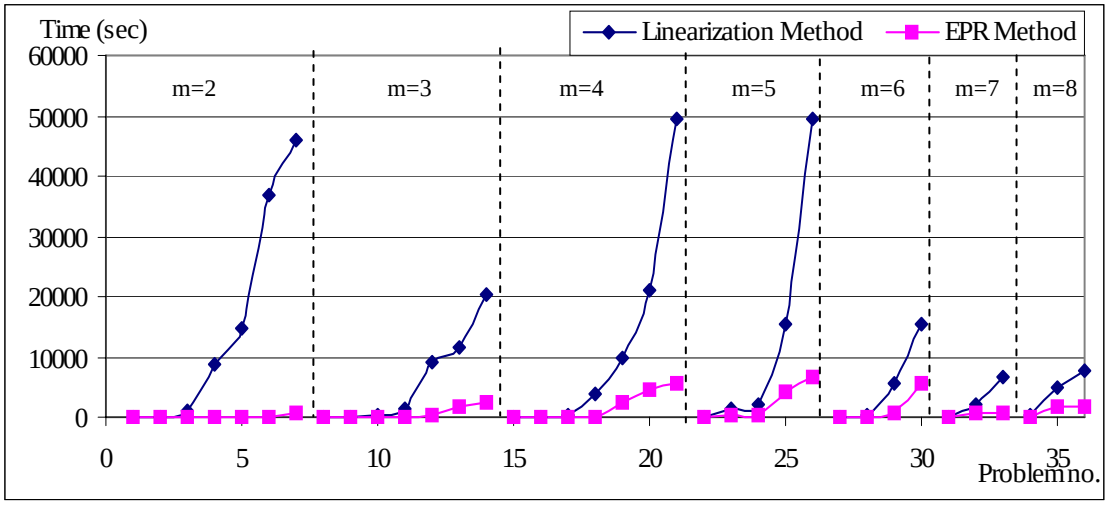


Figure 7 Comparisons of processing time between Linearization and EPR method.

From Table 4 and Figure 7, EPR method uses much less processing time than linearization method for all problem sizes. For the problem sizes that linearization method is not premature terminated, EPR method provides equally optimal solutions. For overall problem sizes including the problem sizes that linearization method requires and does not require premature termination it provides the solution with $-19.4 \pm 14.34\%$ of error within 99.99% confidence interval. This means that EPR method can be statistically ensured that it provides an accurate optimal solution when problem sizes are not more than 1000 variables. Therefore, it is reasonable to use EPR method in EEA for the remaining set of experiments because it statistically provides an optimal or better solution with much less processing time than linearization method for both small and large size problems. Observe that, for all sizes *m*, processing time of linearization method grows exponentially as *n* increases even there are premature terminated cases, while that of EPR method almost disappears when *m* is small and grows with a much smaller rate when *m* is larger.

In summary at this moment, the proposed movement mechanism in EPR method works very well. Moving from z_0 to z_0^* by examining z_c, is very efficient and effective. It can reduce a lot of percentage of error with no difference of processing time. Moving from z_0^* to final solution $z_p; z \in T^k$ and proposed additional stopping rules also work efficiently and effectively because the EPR method stops at optimal solutions or good solutions, compared with solutions from linearization method with and without premature termination respectively, with few additional processing time.

The proposed logic based method is very efficient and effective because it reduces processing time of both methods. According to the trial experiments, for linearization method, some problems that have ever stopped by premature termination can stop at optimal solution with much less time after combining logic based method. To observe the efficiency of this method, the interesting graph of processing time of EPR method corresponding to number of variables, shown in Figure 8, are studied.

Observe from Figure 8 that the beginning of graph of each *m* lies below the graph of *m*-1 at the same $m \times n$ and after passing a certain value of $m \times n$ this graph will lie above graph of *m*-1. After studying in detail, this condition can be explained as follows. At the beginning of the curve of each *m*, average number of customer in each facility = n/m is very small. Therefore, there is a high possibility to occur the cases that logic based method will be used ($\text{some } r_i < \max\{w_j; j = 1,...,n\}$). By numerical experience, comparing processing time and *n/m* of *m* and that of *m*-1 shown in Table 4, this condition usually occurs when $n/m < 2.5$. This means that the processing time of *m*-facility problem with *n/m* < 2.5 will be less than the processing time of (*m*-1)-facility problem with *n/m* $\geq$ 2.5 at the same size of problem $m \times n$. When n/m is over this value, the effect of increasing the number of facility *m* will be obviously expressed. Also, this reason supports summation derived from Figure 7 that even the increasing the number of customers leads to increasing in processing time but it has a less effect than increasing the number of facility. For linearization method increasing in the number of customers has a higher effect than increasing *m*.

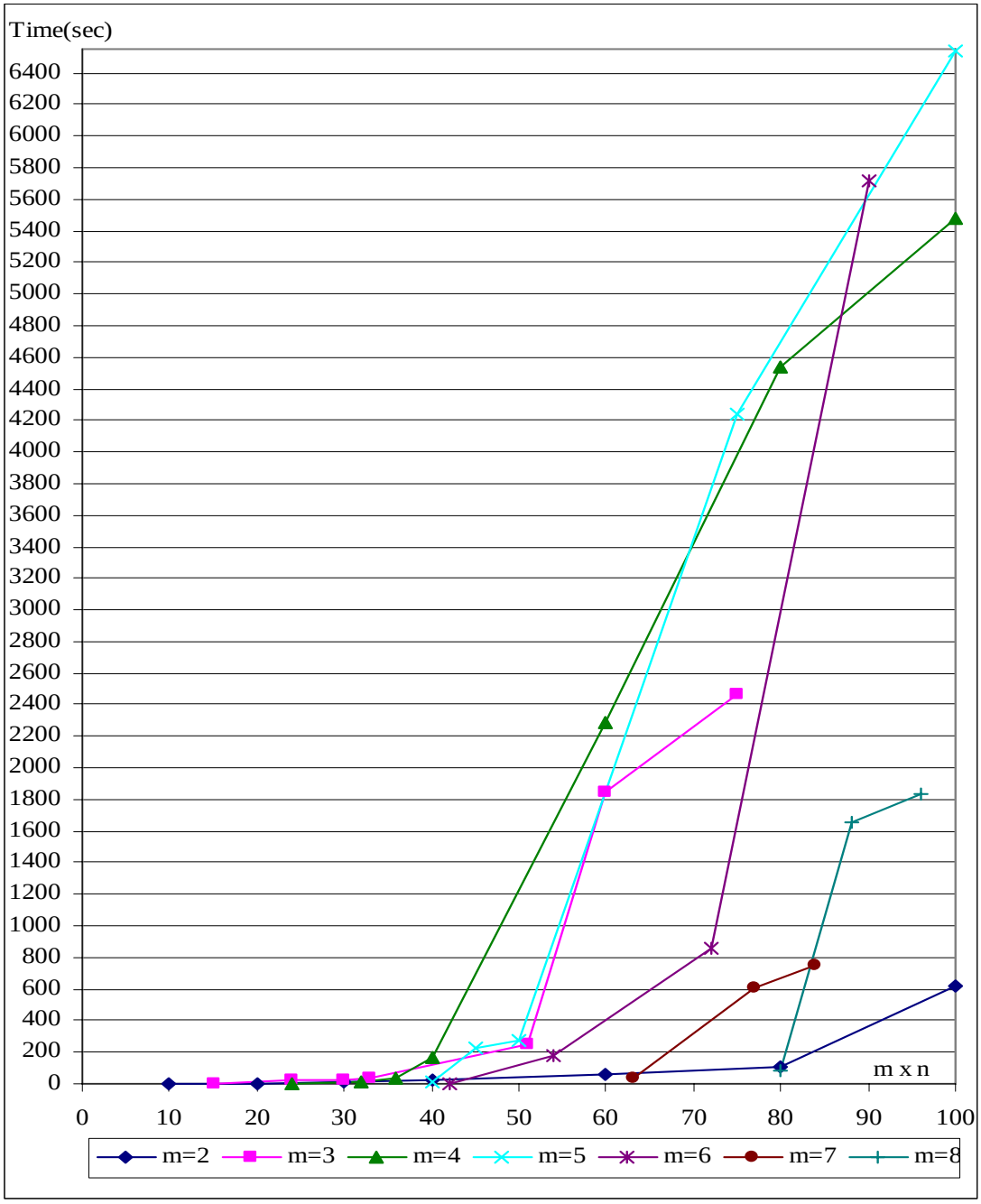


Figure 8 Processing time of EPR method corresponding to $m \times n$

2. Results of and Discussion on HBBA

To evaluate HBBA, the following 31 problem sizes, of which amount of variables are not more than 110, of problem (22) are generated. They vary from $m = 2$ to 10 and $n = 5$ to 50. For each problem size, 5-50 sets of data are generated.

Table 5 Problem sizes of numerical experiments used to evaluate HBBA.

Problem No.	Problem Size		Amount of Trial Cases	Problem No.	Problem Size		Amount of Trial Cases	Problem No.	Problem Size		Amount of Trial Cases
	m	n			m	n			m	n	
1	2	5	50	12	4	8	50	23	7	8	50
2	2	10	50	13	4	9	30	24	7	9	50
3	2	15	30	14	4	13	15	25	7	12	5
4	2	25	15	15	4	18	5	26	8	9	50
5	2	50	5	16	5	6	50	27	8	10	50
6	3	5	50	17	5	8	50	28	8	12	5
7	3	8	50	18	5	10	15	29	9	10	50
8	3	10	30	19	5	15	5	30	9	11	50
9	3	17	15	20	6	7	50	31	10	11	50
10	3	25	5	21	6	9	50				
11	4	6	50	22	6	12	5				

The amount of data set depends on the computational time of EEA. And then, these problems are solved by HBBA and EEA, both of which are coded by MATLAB. The processing time and objective function values for all cases of both algorithms are collected but only averages of them will be reported. The percentage of error is calculated as follows.

$$\% \text{ of error} = \frac{\text{OFV of HBBA - OFV of EEA}}{\text{OFV of EEA}} \times 100$$

Before summarizing the results, an example of using EEA and HBBA are shown as follows to illustrate the principle of these algorithms in practice.

Example 3 A randomly generated problem (m=2, n=15) has the following input data.

w_j = 89, 19, 2, 11, 92, 73, 58, 92, 9, 27, 13, 75, 97, 52, 19 ; $j = 1,...,15$

a_j = 87, 61, 67, 42, 47, 94, 50, 49, 77, 66, 96, 6, 74, 60, 34 ; $j = 1,...,15$

b_j = 2, 60, 199, 92, 63, 187, 98, 46, 108, 22, 24, 30, 29, 108, 55 ; $j = 1,...,15$

s_i = 361, 501 ; $i = 1$ *and* 2

According to the data, $\sum_{j=1}^{15} w_j = 728$. The problem can be mathematically formulated as the following model.

$$\textit{Minimize} \quad f(r, z_{ij}) = -\sum_{i=1}^{2} \frac{1}{r_i}\left[(z_i{}^T wa)^2 + (z_i{}^T wb)^2 \right]$$

$$\textit{subject to} \quad \sum_{i=1}^{2} z_{ij} = 1$$

$$r_i = \sum_{i=1}^{2}\sum_{j=1}^{15} z_{ij} w_j \le s_i$$

$$\textit{, where} \quad wa = [w_1 a_1, ..., w_n a_n]^T \quad ; \forall\, j = 1,...,15$$

$$wb = [w_1 b_1, ..., w_n b_n]^T \quad ; \forall\, j = 1,...,15$$

$$z_i = [z_{i1}, z_{i2}, ..., z_{in}]^T \quad ; \forall\, i = 1 \textit{ and } 2$$

$$s_i = \text{capacity of facility } i \; ; \forall\, i = 1 \textit{ and } 2$$

Firstly, this problem is solved by EEA. After that HBBA is applied with the same problem and the results of these two algorithms will be compared.

Using EEA, the algorithm starts with determining the lower bound and upper bound of every denominator by solving the equation (25) and (26) using "bintprog" command in MATLAB. The results are $LBr_1 = 227, LBr_2 = 367, UBr_1 = 361$, and $UBr_2 = 501$. It can be written in matrix form as follows.

$$r = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix} \in \begin{bmatrix} 227 \;,\; 361 \\ 367 \;,\; 501 \end{bmatrix}$$

By branching strategy of EEA, the first branch starts at $r = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix} = \begin{bmatrix} 227 \\ 728 - 227 = 501 \end{bmatrix}$

The problem (23) with this value of vector r is then solved by EPR method as shown in Example 1 and 2. The objective function value (OFV) is 1.81×10^6. The next branch is obtained by adding r_1 by 1. It is $r = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix} = \begin{bmatrix} 228 \\ 728 - 228 = 500 \end{bmatrix}$. This branching strategy proceeds until the final branch: r_1=361 and r_2=367 is reached. All 135 branches are feasible vector r. These vectors and objective function values are summarized in Table 6.

Table 6 The summarization of all vectors r and objective function values.

Branch no.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**
r_1	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253
r_2	501	500	499	498	497	496	495	494	493	492	491	490	489	488	487	486	485	484	483	482	481	480	479	478	477	476	475
OFV (x10^6)	1.81	1.24	1.44	1.23	1.40	1.19	1.82	1.22	1.23	1.35	1.19	1.39	1.46	1.35	1.21	1.38	1.17	1.49	1.30	1.73	1.32	1.51	1.37	1.48	1.33	1.55	1.44

Branch no.	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**	**48**	**49**	**50**	**51**	**52**	**53**	**54**
r_1	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280
r_2	474	473	472	471	470	469	468	467	466	465	464	463	462	461	460	459	458	457	456	455	454	453	452	451	450	449	448
OFV (x10^6)	1.31	1.47	1.27	1.47	1.50	1.43	1.61	1.45	1.46	1.56	1.42	1.76	1.59	1.65	1.44	1.62	1.40	1.59	1.51	1.74	1.53	1.37	1.68	1.33	1.61	1.66	1.58

Branch no.	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**	**64**	**65**	**66**	**67**	**68**	**69**	**70**	**71**	**72**	**73**	**74**	**75**	**76**	**77**	**78**	**79**	**80**	**81**
r_1	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303	304	305	306	307
r_2	447	446	445	444	443	442	441	440	439	438	437	436	435	434	433	432	431	430	429	428	427	426	425	424	423	422	421
OFV (x10^6)	1.52	1.73	1.48	1.32	1.69	1.28	1.73	1.30	1.72	1.42	1.69	1.77	1.64	1.40	1.28	1.36	1.24	1.75	1.36	1.70	1.39	1.56	1.35	1.53	1.31	1.73	1.33

Branch no.	**82**	**83**	**84**	**85**	**86**	**87**	**88**	**89**	**90**	**91**	**92**	**93**	**94**	**95**	**96**	**97**	**98**	**99**	**100**	**101**	**102**	**103**	**104**	**105**	**106**	**107**	**108**
r_1	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334
r_2	420	419	418	417	416	415	414	413	412	411	410	409	408	407	406	405	404	403	402	401	400	399	398	397	396	395	394
OFV (x10^6)	1.37	1.45	1.33	1.51	1.57	1.45	1.31	1.41	1.27	1.60	1.39	1.71	1.41	1.58	1.39	1.55	1.35	1.73	1.37	1.39	1.49	1.35	1.53	1.56	1.49	1.35	1.51

Branch no.	**109**	**110**	**111**	**112**	**113**	**114**	**115**	**116**	**117**	**118**	**119**	**120**	**121**	**122**	**123**	**124**	**125**	**126**	**127**	**128**	**129**	**130**	**131**	**132**	**133**	**134**	**135**
r_1	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360	361
r_2	393	392	391	390	389	388	387	386	385	384	383	382	381	380	379	378	377	376	375	374	373	372	371	370	369	368	367
OFV (x10^6)	1.31	1.61	1.43	1.69	1.45	1.62	1.49	1.58	1.45	1.76	1.56	1.43	1.58	1.39	1.56	1.56	1.52	1.60	1.54	1.56	1.58	1.52	1.68	1.62	1.62	1.52	1.64

These results can also be illustrated as the following graph provided by MATLAB.

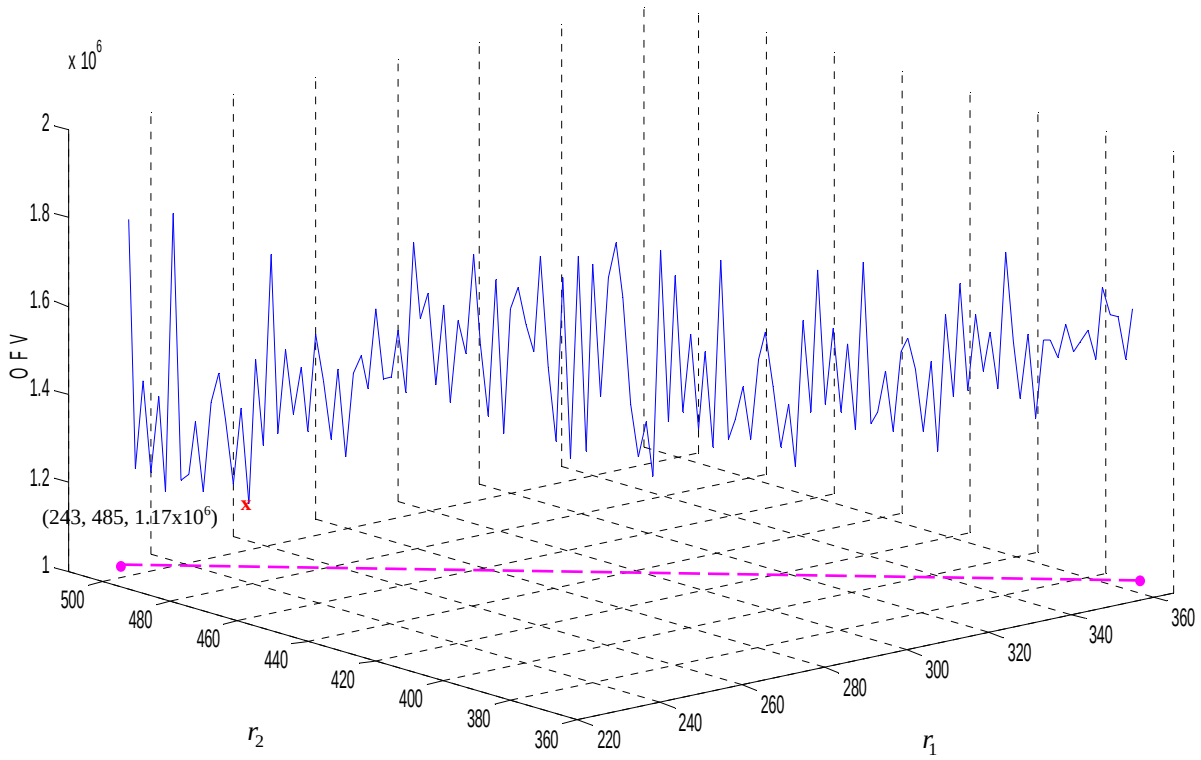


Figure 9 The relationships between vector *r* and its objective function value

In Figure 9, the dotted line lying on the plane of r_1 and r_2 shows the relationship between r_1 and r_2: $\sum_{i=1}^{2} r_i = \sum_{j=1}^{15} w_j = 728$. According to the results in Table 6 and Figure 9, the optimal solution is obtained at r_1=243 and r_2=485 because it provides the minimum value of objective function, which is 1.17x10^6. The solution decomposed by facility is as follows.

$$z_1 = [0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1]^T$$

$$z_2 = [1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0]^T$$

According to the result, customer 2,3,4,6,7,9,14 and 15 are served by facility 1 while the others are served by facility 2. The optimal locations of facility 1 and facility 2 can be obtained using the same method as shown in Example 1. They are as follows. The optimal location of facility 1 is located at $(x_1, y_1) = (65.7449, 121.4733)$ and the optimal location of facility 2 is at $(x_2, y_2) = (56.1505, 33.3505)$.

Using HBBA, the algorithm also starts with determining the lower bound and upper bound of every denominator by solving the equation (25) and (26) using "bintprog" command in MATLAB. They are $LBr^{(0)}{}_1 = 227$, $LBr_2^{(0)} = 367$, $UBr_1^{(0)} = 361$, and $UBr_2^{(0)} = 501$. The initial range of vector r to be considered is as follows

$$\begin{bmatrix} r_1^{(0)} \\ r_2^{(0)} \end{bmatrix} \in \begin{bmatrix} \underline{r_1^{(0)}} & , & \overline{r_1^{(0)}} \\ \underline{r_2^{(0)}} & , & \overline{r_2^{(0)}} \end{bmatrix} \equiv \begin{bmatrix} 227 & , & 361 \\ 367 & , & 501 \end{bmatrix}$$

The lower bound function shown in model (34) with $\underline{r_1^{(0)}} = 227$ and $\underline{r_2^{(0)}} = 367$ is solved by EPR method under this initial range. It provides the initial solution to the trust-region method and then, the trust region method provides the final solution, which is an initial current best solution, at $r_1 = 348$ and $r_2 = 380$ with the objective function value (OFV) =1.39x10^6. This method is referred as MSTR method. After this point, the iterative procedure starts and can be shown as follows.

Iteration 1: Set t=1. Since $\overline{r_1^{(0)}} - \underline{r_1^{(0)}} = \overline{r_2^{(0)}} - \underline{r_2^{(0)}} = 134$, the range of r_1 is arbitrary selected to be partitioned. $\overline{r_1^{(1,1)}} = \left\lfloor \frac{1}{2} \underline{r_1^{(0)}} + \overline{r_1^{(0)}} \right\rfloor = 294$ and $\underline{r_1^{(1,2)}} = 1 + \left\lfloor \frac{1}{2} \underline{r_1^{(0)}} + \overline{r_1^{(0)}} \right\rfloor = 295$. The two mutually exclusive intervals of vector r are

$$r^{(1,1)} \in \begin{bmatrix} 227 & , & 294 \\ 367 & , & 501 \end{bmatrix} \xrightarrow{Tigthen} \begin{bmatrix} 227 & , & 294 \\ \max(367, 728 - 294) = 434 & , & 501 \end{bmatrix} \text{ and } r^{(1,2)} \in \begin{bmatrix} 295 & , & 361 \\ 367 & , & 501 \end{bmatrix}$$

Using MSTR method, the first interval of r gives the final solution at $r_1 = 286$ and $r_2 = 442$ with OFV =1.28x10^6 and another gives the final solution at $r_1 = 348$ and $r_2 = 380$ with OFV =1.39x10^6. The current best solution is changed to the solution of the first interval because it provides the better OFV than the initial current best solution. The OFV is updated to 1.28x10^6. And the next interval of r to be considered is the first interval because it provides better OFV than the latter interval.

Iteration 2: Set *t*=2. Since $\overline{r_1^{(1,1)}} - \underline{r_1^{(1,1)}} = \overline{r_2^{(1,1)}} - \underline{r_2^{(1,1)}} = 67$, the range of r_1 is arbitrary selected to be partitioned. $\overline{r_1^{(2,1)}} = \left\lfloor \frac{1}{2}\underline{r_1^{(1,1)}} + \overline{r_1^{(1,1)}} \right\rfloor = 260$ and $\underline{r_1^{(2,2)}} = 1 + \left\lfloor \frac{1}{2}\underline{r_1^{(1,1)}} + \overline{r_1^{(1,2)}} \right\rfloor = 261$. The two mutually exclusive intervals of vector *r* are

$$r^{(2,1)} \in \begin{bmatrix} 227 & , & 260 \\ 434 & , & 501 \end{bmatrix} \xrightarrow{Tigthen} \begin{bmatrix} 227 & , & 260 \\ 468 & , & 501 \end{bmatrix} \text{ and } r^{(2,2)} \in \begin{bmatrix} 261 & , & 294 \\ 434 & , & 501 \end{bmatrix}$$

Using MSTR method, the first interval of r gives the final solution at $r_1 = 256$ and $r_2 = 472$ with OFV =1.27x10^6 and the other gives the final solution at $r_1 = 286$ and $r_2 = 442$ with OFV =1.28x10^6. The current best solution is changed to the solution of the first interval. The OFV is updated to 1.27x10^6. And, the next interval of r to be considered is the first interval.

Iteration 3: Set *t*=3. Since $\overline{r_1^{(2,1)}} - \underline{r_1^{(2,1)}} = \overline{r_2^{(2,1)}} - \underline{r_2^{(2,1)}} = 33$, the range of r_1 is arbitrary selected to be partitioned. $\overline{r_1^{(3,1)}} = \left\lfloor \frac{1}{2}\underline{r_1^{(2,1)}} + \overline{r_1^{(2,1)}} \right\rfloor = 243$ and $\underline{r_1^{(3,2)}} = 1 + \left\lfloor \frac{1}{2}\underline{r_1^{(2,1)}} + \overline{r_1^{(2,2)}} \right\rfloor = 244$. The two mutually exclusive intervals of vector *r* are

$$r^{(3,1)} \in \begin{bmatrix} 227 & , & 243 \\ 468 & , & 501 \end{bmatrix} \xrightarrow{Tigthen} \begin{bmatrix} 227 & , & 243 \\ 485 & , & 501 \end{bmatrix} \text{ and } r^{(3,2)} \in \begin{bmatrix} 244 & , & 260 \\ 468 & , & 501 \end{bmatrix}$$

Using MSTR method, the first interval of r gives the final solution at $r_1 = 243$ and $r_2 = 485$ with OFV =1.17x10^6 and the other gives the final solution at $r_1 = 256$ and $r_2 = 472$ with OFV =1.27x10^6. The current best solution is changed to the solution of the first interval. The OFV is updated to 1.17x10^6. And, the next interval of r to be considered is the first interval.

Iteration 4: Set *t*=4. Since $\overline{r_1^{(3,1)}} - \underline{r_1^{(3,1)}} = \overline{r_2^{(3,1)}} - \underline{r_2^{(3,1)}} = 16$, the range of r_1 is arbitrary selected to be partitioned. $\overline{r_1^{(4,1)}} = \left\lfloor \frac{1}{2}\underline{r_1^{(3,1)}} + \overline{r_1^{(3,1)}} \right\rfloor = 235$ and $\underline{r_1^{(4,2)}} = 1 + \left\lfloor \frac{1}{2}\underline{r_1^{(3,1)}} + \overline{r_1^{(3,2)}} \right\rfloor = 236$. The two mutually exclusive intervals of vector *r* are

$$r^{(4,1)} \in \begin{bmatrix} 227 & , & 235 \\ 485 & , & 501 \end{bmatrix} \xrightarrow{Tigthen} \begin{bmatrix} 227 & , & 235 \\ 493 & , & 501 \end{bmatrix} \text{ and } r^{(4,2)} \in \begin{bmatrix} 236 & , & 243 \\ 485 & , & 501 \end{bmatrix}$$

Using MSTR method, the first interval of r gives the final solution at $r_1 = 232$ and $r_2 = 496$ with OFV =1.19x10^6 and the other gives the final solution at $r_1 = 243$ and $r_2 = 485$ with OFV =1.17x10^6. The current best solution is unchanged because there is no new solution with lower OFV than the current OFV. And the next interval of r to be considered is the second interval because it provides better OFV than the first interval.

Iteration 5: Set *t*=5. Since $\overline{r_1^{(4,1)}} - \underline{r_1^{(4,1)}} = 7 < \overline{r_2^{(4,1)}} - \underline{r_2^{(4,1)}} = 16$, the range of r_2 is partitioned. $\overline{r_2^{(5,1)}} = \left\lfloor \frac{1}{2}\underline{r_2^{(4,1)}} + \overline{r_2^{(4,1)}} \right\rfloor = 493$ and $\underline{r_2^{(5,2)}} = 1 + \left\lfloor \frac{1}{2}\underline{r_2^{(4,1)}} + \overline{r_2^{(4,2)}} \right\rfloor = 494$. The two mutually exclusive intervals of vector *r* are

$$r^{(4,1)} \in \begin{bmatrix} 236 & , & 243 \\ 485 & , & 493 \end{bmatrix} \xrightarrow{Tigthen} \begin{bmatrix} \max(235, 236) = 236 & , & 243 \\ 485 & , & 493 \end{bmatrix} \text{ and } r^{(4,2)} \in \begin{bmatrix} 236 & , & 243 \\ 494 & , & 501 \end{bmatrix}$$

Using MSTR method, the first interval of r gives the final solution at $r_1 = 243$ and $r_2 = 485$ with OFV =1.17x10^6 and the other gives the infeasible solution. This means that there is no more feasible value of vector *r* in the second interval and then the search process is now pruned. The current best solution is unchanged because there is no new solution with lower OFV than the current OFV. And the next interval of r to be considered is the first interval.

Iteration 6: Set *t*=6. Since $\overline{r_1^{(5,1)}} - \underline{r_1^{(5,1)}} = 7 < \overline{r_2^{(5,1)}} - \underline{r_2^{(5,1)}} = 8$, the range of r_2 is partitioned. $\overline{r_2^{(6,1)}} = \left\lfloor \frac{1}{2} \underline{r_2^{(5,1)}} + \overline{r_2^{(5,1)}} \right\rfloor = 489$ and $\underline{r_2^{(6,2)}} = 1 + \left\lfloor \frac{1}{2} \underline{r_2^{(5,1)}} + \overline{r_2^{(5,2)}} \right\rfloor = 490$. The two mutually exclusive intervals of vector *r* are

$$r^{(6,1)} \in \begin{bmatrix} 236 & , & 243 \\ 485 & , & 489 \end{bmatrix} \xrightarrow{Tigthen} \begin{bmatrix} 239 & , & 243 \\ 485 & , & 489 \end{bmatrix} \text{ and } r^{(6,2)} \in \begin{bmatrix} 236 & , & 243 \\ 490 & , & 493 \end{bmatrix}$$

Using MSTR method, the first interval of r gives the final solution at $r_1 = 243$ and $r_2 = 485$ with OFV =1.17x10^6 and the other gives the final solution at $r_1 = 237$ and $r_2 = 491$ with OFV =1.19x10^6. The current best solution is unchanged because there is no new solution with lower OFV than the current OFV. And the next interval of r to be considered is the first interval because it provides better OFV than the latter interval.

Iteration 7: Set *t*=7. Since $\overline{r_1^{(6,1)}} - \underline{r_1^{(6,1)}} = 4 > \overline{r_2^{(6,1)}} - \underline{r_2^{(6,1)}} = 1$, the range of r_1 is partitioned. $\overline{r_1^{(7,1)}} = \left\lfloor \frac{1}{2} \underline{r_1^{(6,1)}} + \overline{r_1^{(6,1)}} \right\rfloor = 241$ and $\underline{r_1^{(7,2)}} = 1 + \left\lfloor \frac{1}{2} \underline{r_1^{(6,1)}} + \overline{r_1^{(6,2)}} \right\rfloor = 242$. The two mutually exclusive intervals of vector *r* are

$$r^{(7,1)} \in \begin{bmatrix} 239 & , & 241 \\ 485 & , & 489 \end{bmatrix} \xrightarrow{Tigthen} \begin{bmatrix} 239 & , & 241 \\ 487 & , & 489 \end{bmatrix} \text{ and } r^{(7,2)} \in \begin{bmatrix} 242 & , & 243 \\ 485 & , & 489 \end{bmatrix}$$

Using MSTR method, the first interval of r gives the final solution at $r_1 = 241$ and $r_2 = 487$ with OFV =1.21x10^6 and the other gives the final solution at $r_1 = 243$ and $r_2 = 485$ with OFV =1.17x10^6. The current best solution is unchanged because there is no new solution with lower OFV than the current OFV. And the next interval of r to be considered is the second interval.

Iteration 8: Set t=8. Since $\overline{r_1^{(7,1)}} - \underline{r_1^{(7,1)}} = 1 < \overline{r_2^{(7,1)}} - \underline{r_2^{(7,1)}} = 4$, the range of r_2 is partitioned. $\overline{r_2^{(8,1)}} = \left\lfloor \frac{1}{2}\underline{r_2^{(7,1)}} + \overline{r_2^{(7,1)}} \right\rfloor = 487$ and $\underline{r_2^{(8,2)}} = 1 + \left\lfloor \frac{1}{2}\underline{r_2^{(7,1)}} + \overline{r_2^{(7,2)}} \right\rfloor = 488$. The two mutually exclusive intervals of vector *r* are

$$r^{(8,1)} \in \begin{bmatrix} 242 & , & 243 \\ 485 & , & 487 \end{bmatrix} \xrightarrow{Tigthen} \begin{bmatrix} 242 & , & 243 \\ 485 & , & 487 \end{bmatrix} \text{ and } r^{(8,2)} \in \begin{bmatrix} 242 & , & 243 \\ 488 & , & 489 \end{bmatrix}$$

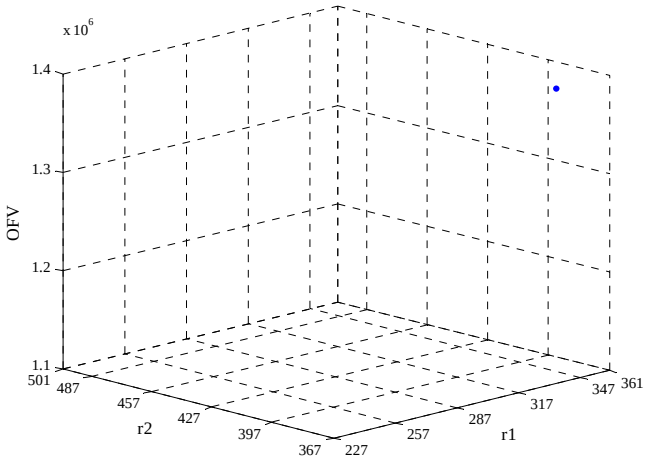
Using MSTR method, the first interval of r gives the final solution at $r_1 = 243$ and $r_2 = 485$ with OFV =1.17x10^6 and the other gives the infeasible solution. This means that there is no more feasible value of vector *r* in the second interval and then the process will be pruned. The current best solution is unchanged. And, the iterative procedure stops because of no more range of r can be partitioned.

In this case, the near optimal solution is provided at $r_1 = 243$ and $r_2 = 485$ with OFV =1.17x10^6 . The solution decomposed by facility is as follows.

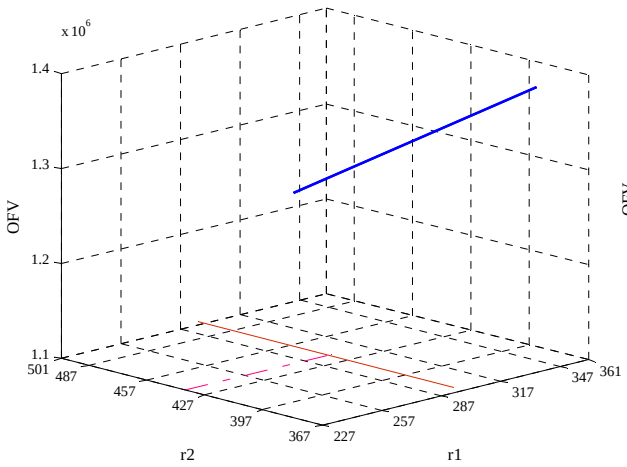
$$z_1 = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}^T$$

$$z_2 = [1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0]^T$$

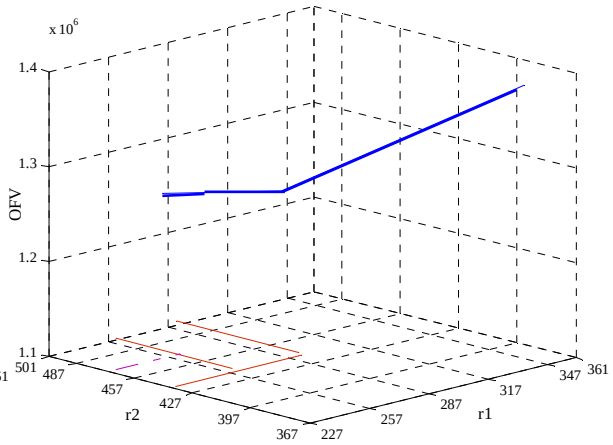
It is the same as the solution provided by EEA. Therefore, the optimal locations of both facilities are the same as that of EEA. To show the movement mechanism of HBBA, a set of graph illustrating the final solution at each iteration will be provided by MATLAB as follows.



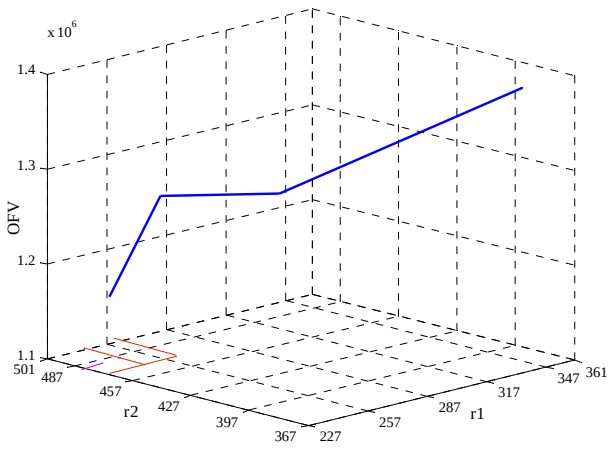
Initial Range



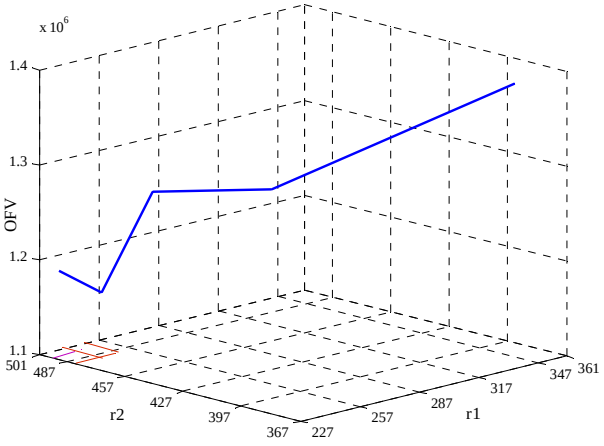
Iteration 1



Iteration 2



Iteration 3



Iteration 4

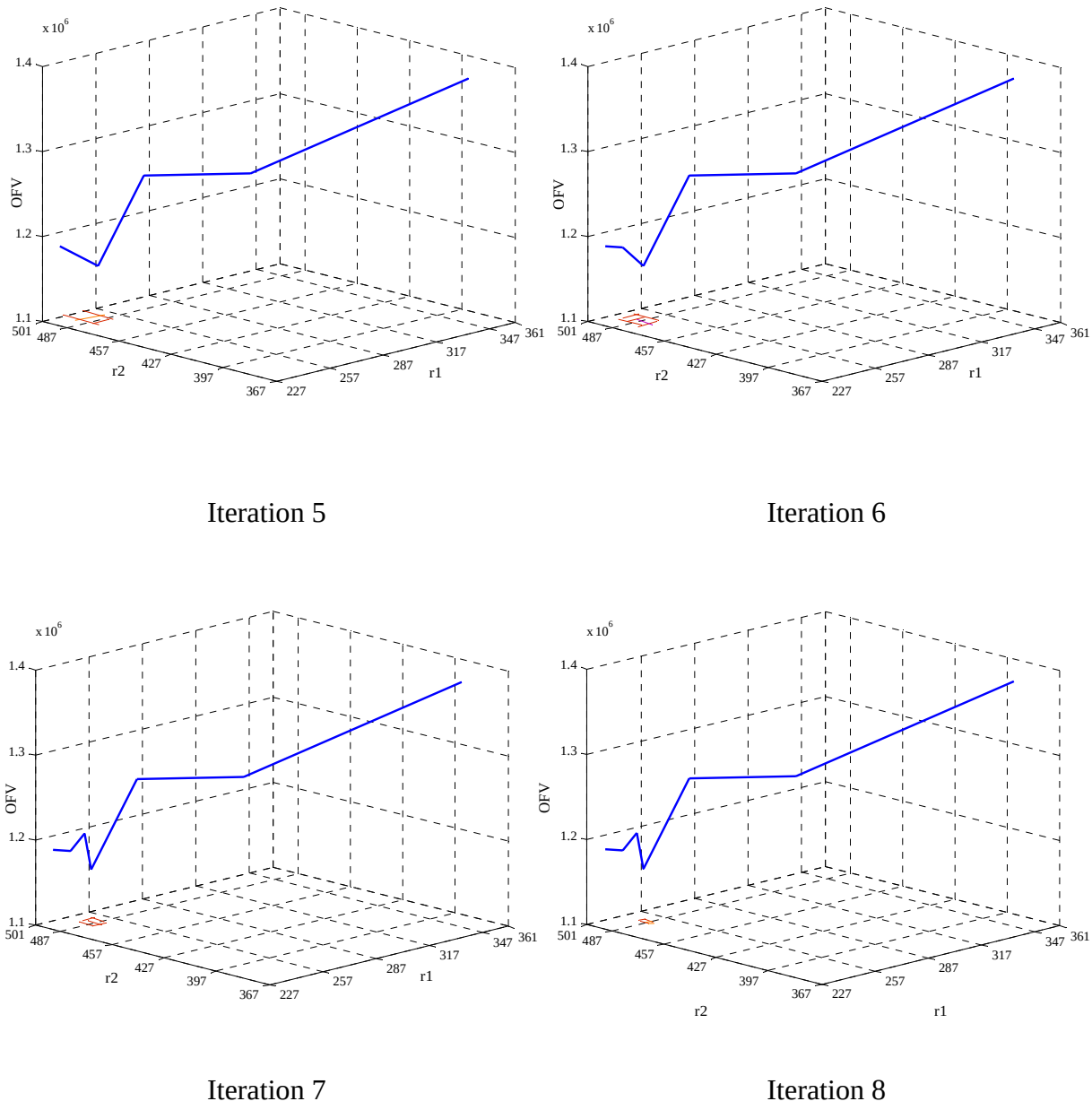


Figure 10 The movement of the solutions for all eight iterations

The results of doing the experiments using these algorithms are shown in Table 7. In the Table, average processing time of both algorithms, average percentage of error of HBBA and confidence interval of the percentage of error are reported. The underlined values in the Table mean there are some premature terminated cases by time limitation at 108,000 seconds or 30 hours in these problem sizes.

Table 7 Processing time of EEA and HBBA and percentage of error of HBBA.

Problem No.	Problem Size		No. of Variables =mn	Empirical Exact Algorithm (EEA)	Heuristic Branch-and-Bound Algorithm (HBBA)		% CI that average percentage of error is less than 3%		n/m	Amount of Trial cases
	m	n		Processing Time (sec)	Processing Time (sec)	Average Error (%)	CI range = $t_{\alpha/2}S/\sqrt{n}$ (±)	% CI (%)		
1	2	5	10	4.03	2.69	0.05	0.14	99	2.50	50
2	2	10	20	207.92	15.33	0.51	0.92	99	5.00	50
3	2	15	30	2128.84	59.30	0.01	0.01	99	7.50	30
4	2	25	50	6848.44	317.39	0.33	0.50	99	12.50	15
5	2	50	100	77139.76	2696.58	0.00	0.00	100	25.00	5
6	3	5	15	26.99	5.43	0.00	0.00	100	1.67	50
7	3	8	24	1356.68	84.89	1.83	1.17	89	2.67	50
8	3	10	30	7967.89	203.18	1.21	1.75	98	3.33	30
9	3	17	51	36956.37	4030.64	0.66	1.17	99	5.67	15
10	3	25	75	91368.40	18922.78	0.00	0.00	100	8.33	5
11	4	6	24	32.03	6.23	0.15	0.38	99	1.50	50
12	4	8	32	1727.63	255.76	0.55	1.22	99	2.00	50
13	4	9	36	4982.28	886.66	1.57	1.40	89	2.25	30
14	4	13	52	22871.74	5602.86	1.29	1.26	99	3.25	15
15	4	18	72	82214.40	17671.80	0.00	0.00	100	4.50	5
16	5	6	30	7.62	7.62	0.00	0.00	100	1.20	50
17	5	8	40	1313.87	119.23	1.95	1.00	90	1.60	50
18	5	10	50	5134.48	1037.18	1.22	1.75	89	2.00	15
19	5	15	75	98023.20	19993.26	0.00	0.00	100	3.00	5
20	6	7	42	26.82	26.82	0.00	0.00	100	1.17	50
21	6	9	54	3187.35	348.12	1.54	1.42	95	1.50	50
22	6	12	72	4908.45	1702.84	0.00	0.00	100	2.00	5
23	7	8	56	51.33	51.33	0.00	0.00	100	1.14	50
24	7	9	63	811.97	433.42	1.65	1.32	91	1.29	50
25	7	12	84	11999.63	2243.96	0.00	0.00	100	1.71	5
26	8	9	72	81.17	81.17	0.00	0.00	100	1.13	50
27	8	10	80	793.54	370.69	0.37	0.96	99	1.25	50
28	8	12	96	6423.86	1387.68	2.99	1.45	90	1.50	5
29	9	10	90	119.09	119.09	0.00	0.00	100	1.11	50
30	9	11	99	756.47	524.08	1.41	1.55	92	1.22	50
31	10	11	110	181.49	181.49	0.00	0.00	100	1.10	50

To descriptively compare processing time of the two algorithms, Figure 11 is created.

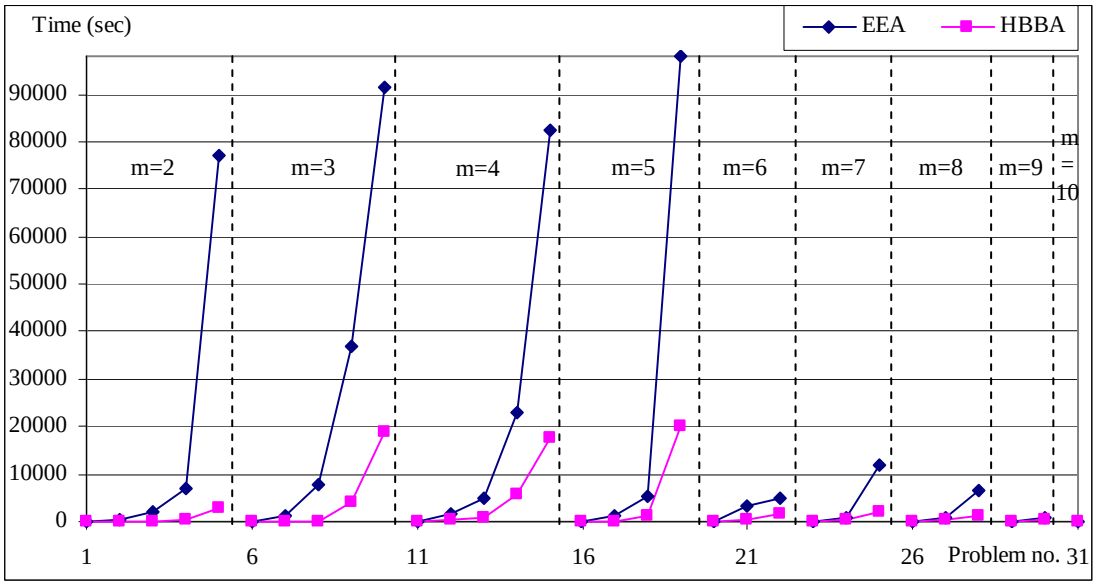


Figure 11 Comparisons of processing time between EEA and HBBA.

From Table 7 and Figure 11, observe that slope of each graph at the beginning is small. After passing a certain point of each graph, slope of a graph will be much steeper. In addition, it is obviously illustrated in case of the graph belongs to EEA. This situation can be explained as follow. Because at the beginning of the graph m and n is so close ($n-m$ is small) and also n/m is small, the width of range of vector r is small. Therefore, the number of branches of vector r in the range is also small. Owing to the small value of n/m, there are high possibilities to occur the cases that use logic based method as explained in the last section. Thus, the processing time of both algorithms increase slowly at the beginning of the graph and the difference between them is small. After passing the certain point, $n-m$ is larger. The width of range of vector r is also large, and then, there are numerous increments in number of branches. It can be observed in case of the performance graph belongs to EEA because all possible branches have to be considered during branching strategy. For HBBA, only some important ranges of vector r are considered. Even there is a wider range of vector r to be considered when $n-m$ is larger, the number of range to be

considered is much smaller than the number of branches to be considered using EEA. Unsurprisingly, when $n-m$ becomes large, HBBA can provide the solution with much less computational time than EEA. The size of m or the number of facilities has highly effects on the processing time. Observe that the increasing rate of processing time after passing some certain points is higher as m increases. The effect of logic based method depending on n/m can be observed by the following graph.

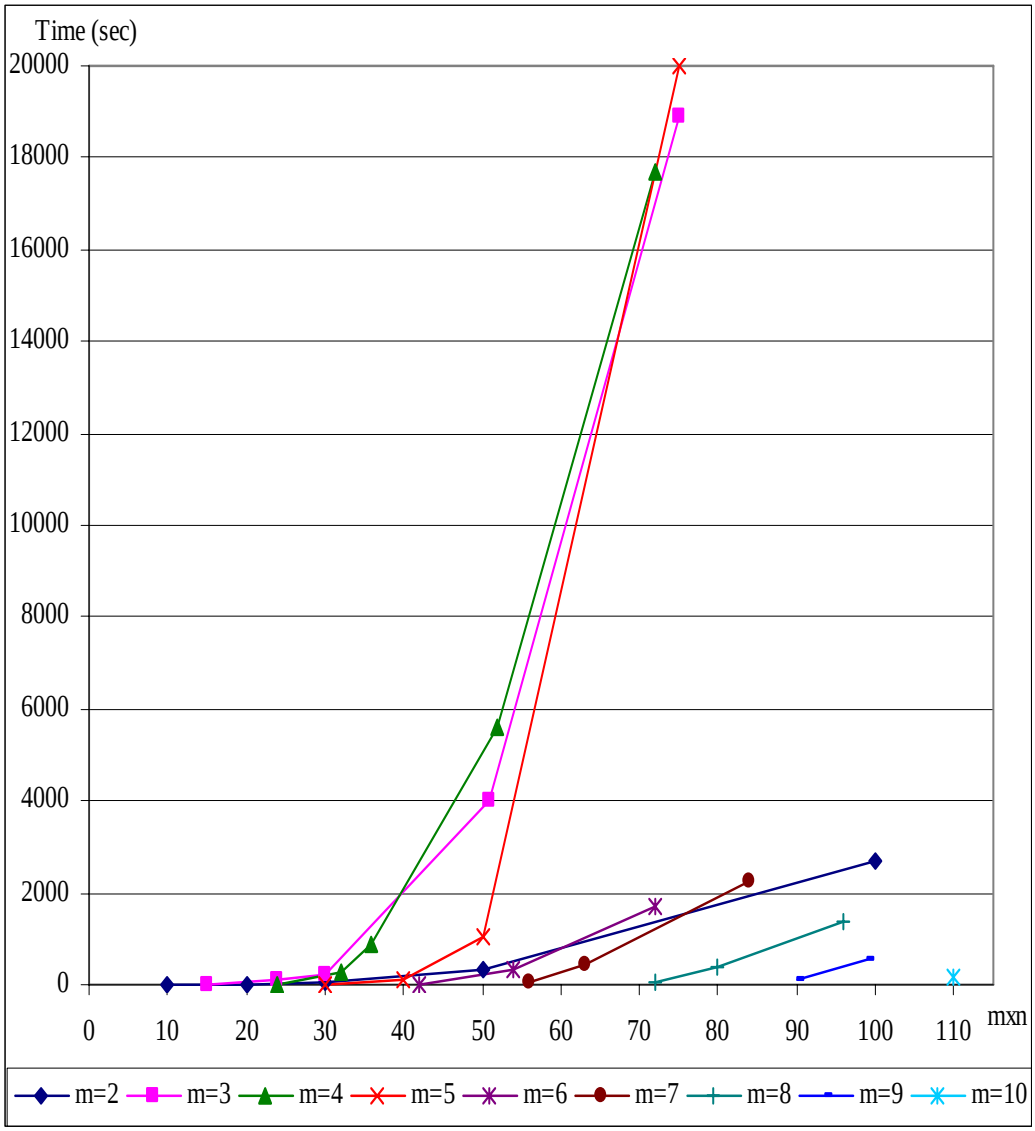


Figure 12 Processing time of HBBA corresponding to $m \times n$

From the results in Figure 12, at the beginning, graph of each m lies below the graph of $m-1$ at the same number of variables $m \times n$ and after passing a certain value of $m \times n$, this graph will lie above the graph of $m-1$. After studying in detail, this condition can be explained as follows. At the beginning of the curve of each m, the average number of customer in each facility = n/m is very small. Therefore, there is a high possibility to occur the case that logic based method is used ($\text{some } \bar{r}_i^{(t)} \text{ or } s_i < \max\{w_j;\ j=1,...,n\}$). By numerical experience, comparing processing time and *n/m* of *m* and that of *m*-1 shown in Table 7, this condition usually occurs when $n/m < 2.25$. This means that the processing time of *m*-facility problem with *n/m* < 2.25 will be less than the processing time of (*m*-1)-facility problem with *n/m* $\geq$ 2.25 at the same size of problem $m \times n$. When n/m is more than this value, the effect of increasing number of facility *m* can be clearly expressed. Also, this reason supports summations derived from Figure 11 that even the increase in the number of customers leads to the increase in processing time but it illustrates less effect than increasing the number of facility.

According to Table 7, HBBA is effective. In the worst case, it can provide the solution with percentage of error less than 3% with at least 89% confidence interval for every problem size. However, this does not mean that the confidence interval of all problem sizes will always reach 3%. To avoid misunderstanding about reading the results of percentage of error, one of the results with *m*=2 *n* =5 will be used as an example. For the problem size of two facilities and five customers, HBBA provides 0.05 % of error on average. The range of average percentage of error is 0.05 $\pm$ 0.14 % or lies between -0.09 and 0.19 % with 99% confidence. Finally, the overall average percentage of error of HBBA, which is obtained by considering all average percentage of all problem sizes using SPSS, is 0.62 $\pm$ 0.42 % or lies between 0.20 and 1.04 % with 99% confidence.

CONCLUSION AND RECOMMENDATION

Conclusion

In this work, the EPR method can solve a specific case of CMLCP, which is minimizing concave quadratic integer programming problem with balanced transportation constraints, more efficiently and effectively than linearization method. For the problem sizes that linearization method is not premature terminated, EPR method provides equally optimal solutions. For overall problem sizes including the problem sizes that linearization method requires and does not require premature termination it provides the solution with $-19.4 \pm 14.34\%$ of error within 99.99% confidence interval with much less processing time than linearization method. The EPR method starts with developing the lower bound function of a quadratic integer programming problem, which is a linear integer programming problem. The lower bound function will be then solved. The next adjacent vertices around the solutions will be explored and ranked under the extreme point ranking approach. To construct the lower bound function, Hessian matrix *G* is needed. The algebraic method proposed to construct Hessian matrix reduces time to construct Hessian matrix by using calculus, and then reduces processing time yet provides the equivalent Hessian matrix. In addition, owing to the number of variables to be solved by EPR method is $m \times n$ not $m \times n \times (n+1)/2$ like linearization method, it can solve large-sized problems $(100 < m \times n \leq 1000)$ that linearization method cannot solve. The logic based method developed from supply constraints set can reduce the number of variables to be branched in both methods, and then reduce processing time and extends the problem sizes to be experimented. Moreover, the EPR method can improve solutions in high rate and stop at an optimal solution or a good solution compared with non-premature terminated or premature terminated cases of linearization method respectively because of good movement mechanisms. These mechanisms are selecting z_0^* using z_c based on gradient property reason and exchanging method based on special properties of balanced transportation constraints. Owing to the result that processing time of

EPR method is dependent on increasing m more than n, the algorithm is appropriate to use in the realistic problem with n much more than m.

For a general case of CMLCP, CMLCP with unbalanced transportation constraints, the studied problem becomes the sum-of-nonlinear ratios problem. Thus, the proposed exact, empirical exact and heuristic algorithms are based on the general algorithms of such problem, which are branch-and-bound algorithms. For exact algorithm, linearization method can be combined with the explicitly branching on range of denominators. For empirical exact algorithm, EPR method can be combined with explicitly branching on range of denominators to implicitly provide the optimal solution. Observe that only one difference between exact and empirical exact algorithm is the method used to solve the subproblem in each branch, which is a specific case of CMLCP, because they use the similar branching strategy. Therefore, the empirical exact algorithm using EPR method will be selected to be the reference algorithm because the results from the first section show that EPR method provides equally optimal solution with higher efficiency. To re-ensure the fact that the empirical exact algorithm should really be the reference algorithm, the additional numerical experiments to test its performance comparing to exact algorithm were also conducted. The results are shown in the Appendix section. Due to explicitly branching strategy, this empirical exact algorithm strongly depends on the number of customers (n). The more customers are considered the more computational effort is required because of wider range of denominators being explored. Due to the nature of EPR method that is sensitive to number of facility (m), EEA is unavoidable to be also sensitive to m.

The proposed heuristic algorithm, HBBA, reduces number of considered branch using rectangular partition. With the benefit from this branching strategy, the effect on increasing of n on HBBA is much less than that on EEA. This branching strategy is combined with multistart trust-region method (MSTR) to globally provide the best solution in the considered range efficiently and effectively. EPR method considered only incumbent solution can provide the solution to lower bound and also the initial solution to the trust-region method efficiently and effectively. The trust-

region method works well. It can provide a good solution to be then used to select the next interval of range of denominator to explore further. Moreover, it can expedite the improvement of the solution effectively. With the benefit from branching strategy and MSTR method, HBBA can provide the good solution with percentage of error 0.62 $\pm$ 0.42 % with 99% confidence with much less computational effort than EEA.

Recommendation

The recommendation for applying the proposed algorithms composes of two sections. The first section mentions about the suggestion of using the proposed algorithms with the real-world case of CMLCP. The other suggests the extensions of the proposed algorithms for the future research.

Applications of the Proposed Algorithms with the Real-world Case of CMLCP

Even EEA and HBBA can solve the problem effectively and efficiently as discussed in the last part. There are some restrictions that the ones, who would like to use these algorithms, should keep in mind. They are as follows.

1. The actual processing time can be more than that reported in Table 7 because both EEA and HBBA are sensitive to the variation of weight of customers. The higher variation of customers' weight is the higher computational effort is required due to the wider range of denominator. Obviously, EEA is sensitive to this factor much more than HBBA because of its branching strategy. The numerical experiments done in this research varies the customers' weight from 1-100 and 1-50 randomly. For the variation of demand over 1-100, the user should keep in mind that the higher of computational effort may be required.

2. The locations of facilities provided by EEA and HBBA are only a better guide solution to the decision makers; even they are optimal solutions. It may provide the location that in real-world problem is not practical because one of the assumptions

assumes the considered plane having non-allowable area. Moreover, the other factors that impact the distance function such as traffic jam are not considered in this thesis. The decision makers should combine the provided solution with their other quantitative information such as marketing information and qualitative information related to their experience so as to enhance quality of the solution and finally make a better decision.

Extensions of the Proposed Algorithms for the Future Research

The proposed algorithms can be applied for the various extended versions of CMLCP as follows. The first version is Multi-level CMLCP. It is to consider CMLCP with the multiple levels of product distribution. For example, a company needs to find the appropriate locations of new warehouses to distribute products to retail outlets by passing the dealers and to allocate these dealers to the warehouses so as to minimize the total transportation distance among warehouses, dealers and retail outlets. An algorithm for such problem may be as follows. The algorithm starts with determining the initial allocation of retail outlets to the dealers. Therefore, the system below the dealers can be shrunk into a large group with adding the cost effect at retail levels estimated by actual transportation distance or distance function. The subproblem with this cost effect can be reduced to equation (20) as usual CMLCP and solved by EEA or HBBA. After that, the iterative procedure of exchanging the specified number of assignment retail outlets from its current regional dealers to the new ones and solving the subproblem proceeds until no further improvement on the objective function can be found.

The other version is fixed-cost CMLCP. It is CMLCP regarding fixed cost corresponding to location area or size of facility. For this problem, the EEA or HBBA can provide the initial solution by ignoring fixed cost in the first stage. After that, the algorithms with defined neighborhood structure such as Love and Juel (1982)' algorithm will be proceeded by exchanging the specified number of assignments of customers from their current facilities to new ones until no further improvement on the objective function with adding fixed cost function can be found.

LITERATURE CITED

Adams, W.P. and H.D. Sherali. 1986. A tight linearization and an algorithm for 0-1 quadratic programming problems. **Management Sci.** 32(10): 1274-1290.

Akino, U. and B.M. Khumawala. 1977. An efficient branch and bound algorithm for the capacitated warehouse location problem. **Management Sci.** 20: 822-844.

Benson, H.P. 2002. Using concave envelopes to globally solve the nonlinear sum of ratios problem. **J. of Global Optimization.** 22: 343-364.

Birbil, S.I., J.B.G. Frenk and S. Zhang. 2005. **Generalized Fractional Programming with User Interaction.** Available Source: http://hdl.handle.net/1765/1325, June 30, 2007.

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

Brimberg, J. and N. Mladenovic. 1996a. Solving the continuous location-allocation problem with Tabu search. **Studies in Locational Analysis.** 8: 23–32.

_____ and _____. 1996b. A variable neighborhood algorithm for solving the continuous location-allocation problem. **Studies in Locational Analysis.** 10: 1–12.

_____, P. Hansen, N. Mladenovic and E.D. Tillard. 2000. Improvements and comparison of heuristics for solving the uncapacitated multisource weber problem. **Operations Res.** 48 (3): 444-460.

_____, R. Chen and D. Chen. 1998. Accelerating convergence in the Fermat-Weber location problem. **Operations Res. Letters.** 22: 151-157.

Cabot, A.V. and L.F. Richard. 1970. Solving certain nonconvex quadratic minimization problems by ranking the extreme points. **Operations Res.** 18: 82-86.

Charalambous, C. and J. W. Bandler. 1976. Non-linear optimization as a sequence of least *p*-th optimization with finite values of *p*. **Int. J. System Sci.** 7: 377–391.

Charnes, A. and W.W. Cooper. 1962. Programming with linear fractional functionals. **Naval Res. Logistics Quart.** 9: 181-186.

Chen, P. C., P. Hansen, B. Jaumard and H. Tuy. 1998. Solution of the multisource Weber and conditional Weber problems by d.-c. programming. **Operations Res.** 46: 548–562.

Chen, R. 1983. Solution of minisum and minimax locationallocation problems with Euclidean distances. **Naval Res. Logistics Quart.** 30: 449–459.

Cooper, L. 1964. Heuristic methods for location-allocation problems. **SIAM Rev.** 6: 37-53.

Dinkelbach, W. 1967. On nonlinear fractional programming. **Managemetn Sci.** 13: 492-498.

Drezner, Z. 1984. The planar two-center and two-median problems. **Transportation Sci.** 8: 351–361.

Falk, J.E. and K.L. Hoffman. 1976. A successive underestimating method for concave minimization problems. **Mathematics of Operations Res.** 1: 251-259.

_____ and S.W. Palocsay. 1994. Image space analysis of generalized fractional programs. **J. of Global Optimization.** 4: 63-88.

Floudas, C.A. and V. Visweswaran. 1995. **Quadratic Optimization**. Available Source: http://citeseer.ist.psu.edu/26184.html, June 30, 2007.

Frenk, J.B.G. and S. Schaible. 2004. **Fractional Programming.** Available Source: http://www. erim.eur.nl, June 30, 2007.

Freund, R.W. and F. Jarre. 1999. **Solving the Sum-of-Ratios Problem by an Interior-point Method.** Available Source: http://cm.bell-labs.com/cs/doc/99, June 30, 2007.

Gamal, M.D.H. and S.Salhi. 2003. A cellular heuristic for the multisource Weber problem. **Computers & Operations Res.** 30: 1609 –1624.

Garcia, F., B. Melian, J.A. Moreno and J.M. Moreno-Vega. 2003. **Scatter Search for Multiple Objective p-Facility Location Problems.** Available Source: http:// www.elsevier.com/locate/parco, April 24, 2005.

Glover, F. 1975. Improved linear integer programming formulations of nonlinear integer problems. **Management Sci.** 22: 455-460.

_____ and E. Woolsey. 1974. Converting the 0-1 polynomial programming problem to a 0-1 linear program. **Operations Res.** 22: 180-182.

Gupta, R., L. Bandopadhyaya and M.C. Puri. 1996. Ranking in quadratic integer programming problems. **European J. of Operational Res.** 95: 231-236.

Hansen, P., N. Mladenovic and E. Taillard. 1998. Heuristic solution of the multisource Weber problem as a p-median problem. **Operations Res. Letters.** 22: 55–62.

Houck., C.R., J.A. Joines and M.G. Kay. 1997. **Characterizing Search Spaces for Tabu Search.** Available Source: http://citeseer.ist.psu.edu/109360.htm, 2007.

Isbell, J.R. and W.H. Marlow. 1956. Attrition games. **Naval Res. Logistics Quart.** 3: 71-93.

Kleibohm, K. 1967. Remarks on the nonconvex programming problem. **Unternchmensh.** 11: 49-60.

Konno, H. 1976. A cutting plane algorithm for solving bilinear programs. **Mathematical programming.** 11: 14-27.

Koskosidis, I. and W.B. Powell. 1992. Clustering algorithms for consolidation of customer orders into vehicle shipments. **Transportation Res.** 26B: 365 -379.

Kuenne, R. E. and R. M. Soland. 1972. Exact and approximate solutions to the multisource Weber problem. **Mathematical Programming.** 3: 193–209.

Kuhn, T. 1973. **The Structure of Scientific Revolutions**, 2nd ed., University of Chicago Press, Chicago, IL.

Kuno, T. 2002. A branch-and-bound algorithm for maximizing the sum of several linear ratios. **J. of Global Optimization.** 22: 155-174.

_____. 2005. A revision of the Trapezoidal branch-and-bound algorithm for linear sum-of-ratios problems. **J. of Global Optimization.** 33: 215-234.

Levin,Y. and A.B. Israel. 2004. A heuristic method for large-scale multi-facility location problems. **Computers & Operations Res.** 31: 257-272.

Lorena, L.A.N. and E.L.F. Senne. 2003. **Local Search Heuristics for Capacitated p-median Problems.** Available Source: http://www.citeseer.istpsu.edu/306404.htm, June 30, 2007.

Love, R. F. and H. Juel. 1982. Properties and solution methods for large location-allocation problems. **J. Operational Res. Soc.** 33: 443–452.

_____ and J. G. Morris. 1975. A computational procedure for the exact solution of location-allocation problems with rectangular distances. **Naval Res. Logistics Quart.** 22: 441–453.

Martos, B. 1964. Hyperbolic programming. **Naval Res. Logistics Quart.** 9: 181-186.

Mladenovic, N. and J. Brimberg. 1995. A descent-ascent technique for solving the multisource Weber problem. **Yugoslav J. Operations Res.** 5: 211–219.

Moreno, J., C. Rodrigez and N. Jimenez. 1990. Heuristic cluster algorithm for multiple facility location-allocation problem. **RAIRO. Operations Res.** 25: 97–107.

Mulvey, J. and M. Beck. 1984. Solving capacitated clustering problems. **European J. of Operational Res.** 18: 339- 348.

Murtagh, B. A. and S.R. Niwattisyawong. 1982. An efficient method for the multi-depot location-allocation problem. **J. Operational Res. Soc.** 33: 629–634.

Murty, K. 1968. **S**olving the fixed-charge problem by ranking the extreme points. **Operations Res..** 16: 268-279.

Nauss, M.R. 1978. An improved algorithm for the capacitated facility location problem. **J. Operational Res. Soc.** 29(12): 1195-1201.

Pardalos, P.M. 1986. An algorithm for a class of nonlinear fractional problems using ranking of the vertices. **BIT.** 26: 392-395.

Phillips, A.T. 1998. **Quadratic Fractional Programming: Dinkelbach's Method.** Available Source: http://www. citeseer.ist.psu.edu, June 30, 2007.

_____ and J.B. Rosen. 1988. A parallel algorithm for constrained concave quadratic global minimization. **Mathematical Programming.** 42: 421-448.

Ritter, K. 1966. A method for solving maximum problems with a nonconcave quadratic objective function. **Z. Wahrshcinlihkeitstheoric Geb.** 4: 340-351.

_____. 1967. A parametric method for solving certain nonconcave maximization problem. **J. Computer and System Sci.** 1: 44-54.

Rosen, J.B. and P.M. Pardalos. 1986. Global minimization of large-scale constrained concave quadratic problems by separable programming. **Mathematical Programming.** 34: 163.

Rosing, K. E. 1992. An Optimal method for solving the (generalized) multi-Weber problem. **European J. of Operational Res.** 58: 414–426.

Sa, G. 1969. Branch-and-bound and approximate solutions to the capacitated plant-location problem. **Operations Res.** 17: 1005-1016.

Schaible, S. and J. Shi. 2003. Fractional programming: The sum-of-ratios case. **Optimization Methods and Software.** 18(2): 219-229.

Sherali, H.D. and C.H. Tuncbilek. 1992. A squared-Euclidean distance location-allocation problem. **Naval Res. Logistics.** 39: 447 –469.

Sullivan, P. J. and N. Peters. 1980. A flexible user oriented location-allocation algorithm. **J. Environmental Management.** 10: 181–193.

Tammer, K., Chr. Tammer and E. Ohlendorf. 1999. **Multicriterial Fractional Optimization.** Available Source: http://edoc.hu-berlin.de, June 30, 2007.

Thoai, N.V. 1998. Global optimization techniques for solving the general quadratic integer programming problem. **Computational Optimization and Applications.** 10: 149-163.

Tuy, H. 1964. Concave programming under linear constraints. **Soviet Mathematics.** 5: 1437-1440

Verma, V., M.C. Puri and H.C. Bakhshi. 1991. Constrained integer linear programming. **Asia-Pacific J. of Operational Res.** 8: 70-79.

Zizong, Y., F. Pusheng and W. Zhongping. 2004. **A branch and Cut Algorithm for Solving the Linear and Quadratic Integer Programming Problem.** Available Source:http://www.optimization-online.org/DB_HTML/2005/01/1052.html , June 30, 2007.

Zwart, P.B. 1973. Nonlinear programming: Counterexamples to two global optimization algorithms. **Operations Res.** 21(6): 1260-1266.

_____. 1974. Global maximization of a convex function with linear inequality constraints. **Operations Res.** 22: 602

APPENDIX

Evaluations of the Exact Algorithms

To make a decision on a reference algorithm to HBBA, the efficiency of Exact Algorithm (EA) and Empirical Exact Algorithm (EEA) is evaluated. Even EA can be theoretically ensured to provide the optimal solution; it trends to take unacceptably high computational time. In each branch, linearization method, which requires much higher computational time than EPR method as shown in Table 4, is applied. Therefore, EA has a higher possibility than EEA to consume unacceptably high computational time for the problem under unbalanced transportation constraints, which composes of a lot of branches. The separately numerical data of problem (22) is constructed to ensure this foreseeing. The 20 following problem sizes each of which composes of 5-10 data sets are generated.

Appendix Table 1 Problem sizes of numerical experiments used to evaluate EA.

Problem No.	Problem Size		Problem No.	Problem Size	
	m	n		m	n
1	2	5	11	5	8
2	2	10	12	5	10
3	2	15	13	6	7
4	3	5	14	6	9
5	3	8	15	7	8
6	3	10	16	7	9
7	4	6	17	8	9
8	4	8	18	8	10
9	4	9	19	9	10
10	5	6	20	9	11

The processing time of EA and EEA will be collected and the average processing time and percentage of error of EEA calculated as follows are reported in Appendix Table 2.

$$\% \text{ of error} = \frac{\text{OFV of EEA - OFV of EA}}{\text{OFV of EA}} \times 100$$

Appendix Table 2 Processing time of EA and EEA and percentage of error of EEA.

Problem No.	Problem Size		Exact Algorithm		Empirical Exact Algorithm			n/m	Amount of trial cases
	m	n	No. of variables mn(n+1)/2	Processing Time (sec)	No. of variables mxn	Processing Time (sec)	Average Error (%)		
1	2	5	30	4.66	10	3.97	0	2.50	10
2	2	10	110	2152.11	20	191.23	0	5.00	10
3	2	15	240	51954.00	30	2376.58	-0.91	7.50	5
4	3	5	45	37.45	15	23.71	0	1.67	10
5	3	8	108	5213.43	24	1238.16	0	2.67	10
6	3	10	165	87558.16	30	8329.41	0	3.33	5
7	4	6	84	101.56	24	27.99	0	1.50	10
8	4	8	144	13276.43	32	1928.46	0	2.00	10
9	4	9	180	29791.04	36	4015.35	0	2.25	5
10	5	6	105	71.87	30	9.15	0	1.20	10
11	5	8	180	4139.66	40	1452.13	0	1.60	10
12	5	10	275	45954.00	50	5965.40	0	2.00	5
13	6	7	168	96.56	42	16.28	0	1.17	10
14	6	9	270	17708.20	54	3218.40	0	1.50	5
15	7	8	252	217.00	56	54.78	0	1.14	10
16	7	9	315	16584.99	63	862.71	0	1.29	5
17	8	9	360	219.53	72	98.15	0	1.13	10
18	8	10	440	17555.82	80	1359.10	0	1.25	5
19	9	10	495	227.70	90	107.07	0	1.11	10
20	9	11	594	18386.62	99	1422.24	0	1.22	5

The underlined values in Appendix Table 2 mean that there are some cases in these problem sizes that are premature terminated by time limitation. The limitation of processing time is set at 108,000 seconds or 30 hours. EA requires the maximum average processing time at 87,558.16 seconds or 24.32 hours to solve only a small size problem $(m, n) = (3,10)$ with premature termination. Moreover, the EA can solve only problem with very small *n/m* achieving acceptable computational time and its efficiency considerably decrease when *n/m* increases. Owing to the branching strategy and the additional variables required by linearization method, EA is very sensitive to the increase in number of customer (*n*). EEA can solve all problem sizes with much less processing time and with better or the same quality of solution. For larger sizes, there is a high risk that EA cannot solve the problem with acceptable time. Conversely, the EEA has a high possibility to be extended to use in larger-size problems. Therefore, EEA is selected to provide reference solutions in this research.

CURRICULUM VITAE

NAME : Miss Chansiri Singhtaun

BIRTH DATE : September 15, 1979

BIRTH PLACE : Bangkok, Thailand

EDUCATION :

YEAR	INSTITUTION	DEGREE/DIPLOMA
2000	Kasetsart Univ.	B.Eng. Hons. (Industrial Engineering)
2003	Kasetsart Univ.	M.Eng. (Industrial Engineering)

SCHOLARSHIP : Thai Government Scholarship 2001-2007