Journal of Computer Science 3 (8): 583-591, 2007
ISSN 1549-3636

An Efficient Algorithm for Capacitated Multifacility Location Problems

Chansiri Singhtaun and Peerayuth Charnsethikul
Operations Research and Management Science Units, Industrial Engineering Department
Kasetsart University, Bangkok 10903, Thailand

Abstract: In this paper, a squared-Euclidean distance multifacility location problem with inseparable demands under balanced transportation constraints is analyzed. Using calculus to project the problem onto the space of allocation variables, the problem becomes minimizing concave quadratic integer programming problem. The algorithm based on extreme point ranking method combining with logical techniques is developed. The numerical experiments are randomly generated to test efficiency of the proposed algorithm compared with a linearization algorithm. The results show that the proposed algorithm provides a better solution on average with less processing time for all various sizes of problems.

Keywords: Multifacility location problem, Minimizing concave function, Quadratic integer programming Problem, Extreme point ranking method

INTRODUCTION

Facility location problem, searching for the optimum locations of facilities to attain minimum cost or maximum profit, is a significant problem that almost every organization has to face with. The appropriate locations for the facilities, such as warehouses, factories, service centers, outlets, clinics, and so on, to provide the minimum total distances between facilities and customers are completely required to enhance the organization's performance. Conversely, if the facilities are not located in appropriate locations, the company will not only be suffered from large investment but it will be suffered from low manufacturing and service performance for a long period as well. Thus, it is undeniable that the excellent decision on facility location is indispensable for all types of organization to enhance the core competence of the company. As a consequence, this decision problem has been studied continuously for many decades in both wider and deeper area of research.

Capacitated version of multifacility location-allocation problem (CMLP), 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, is a specific class of facility location problem proven to be NP-hard. This means that it is hard or impossible to be solved by the exact methods when the problem is large. Unsurprisingly, most of the developed algorithms for CMLP are heuristic algorithms, which although cannot provide the best solution, they give good solutions with much less computational effort than exact algorithms. CMLP has been studied in wild diverse versions, which can be classified as follows. The classical CMLP studied by Nauss[1], Sa[2], Akino and Khumawala[3] considered the problem on network and the products are separable or can be fraction. The p-median problem studied by Mulvey and Beck[4], Koskosidis and Powell[5], Lorena and Seene[6] considered the CMLP on network with inseparable products. The last class which was studied by Sherali and Tuncbilek[7] is to consider CMLP on plane with separable products. Therefore, the heuristic algorithms for CMLP have wild varieties depending on defining the problem.

In this paper, an efficient algorithm for another specific class of CMLP, which considers the problem on plane with inseparable product, is developed under balanced transportation constraints. The objective is to minimize total distance measured by squared-Euclidean metric between the facilities and customers. It will be hereafter called Capacitated Multifacility Location Clustering Problem (CMLCP). The applications of CMLCP can be usually found in computer network or electronics system setup such as finding the appropriate locations of computer servers and allocation of the clients to these servers in order to minimize losses due to distance between servers and clients.

Corresponding Author: Peerayuth Charnsethikul, Operations Research and Management Science Units, Industrial Engineering Department, Kasetsart University, Bangkok 10903, Thailand

MATERIALS AND METHOD

This part describes the proposed algorithm to solve CMLCP which can be divided into 3 sections as follows.

Define the problem and formulate mathematical model: The studied problem is described as follows. There are $m > 1$ new facilities to be located on the continuous plane with a certain capacity. 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 the total distance measured in squared-Euclidean metric with respect to facility capacity. The problem can be mathematically formulated as follows.

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

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

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

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

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

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

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

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

and $x_i, y_i \in R^m$

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 limitation are respect. 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 (1) to x_i and y_i at the solution

$$x_i = \frac{\sum_{j=1}^{n} z_{ij} w_j a_j}{\sum_{j=1}^{n} z_{ij} w_j} \text{ and } y_i = \frac{\sum_{j=1}^{n} z_{ij} w_j b_j}{\sum_{j=1}^{n} z_{ij} w_j} \; ; i = 1,...,m \qquad (2)$$

Substituting (2) into the objective function of (1), the objective function can be projected onto the

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

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

space of z_{ij} variables as follows.

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

$$= -\sum_{i=1}^{m} \frac{1}{\sum_{j=1}^{n} z_{ij} w_j} \left[\left(\sum_{j=1}^{n} z_{ij} w_j a_j \right)^2 + \left(\sum_{j=1}^{n} z_{ij} w_j b_j \right)^2 \right] + \sum_{j=1}^{n} w_j (a_j^2 + b_j^2) \qquad (3)$$

The equation (3), which is the reduced objective function (1), is equivalent to the following equation

$$\text{Minimize} -\sum_{i=1}^{m} \frac{1}{\sum_{j=1}^{n} z_{ij} w_j} \left[\left(\sum_{j=1}^{n} z_{ij} w_j a_j \right)^2 + \left(\sum_{j=1}^{n} z_{ij} w_j b_j \right)^2 \right] \qquad (4)$$

or,

$$\text{Maximize} \quad \sum_{i=1}^{m} \frac{1}{\sum_{j=1}^{n} z_{ij} w_j} \left[\left(\sum_{j=1}^{n} z_{ij} w_j a_j \right)^2 + \left(\sum_{j=1}^{n} z_{ij} w_j b_j \right)^2 \right] \qquad (5)$$

The problem (1) with objective function (4) can be written in the matrix form as follows.

$$P1: \underset{z \in S}{\text{Minimize}} \; f(z) = -\left\{ \sum_{i=1}^{m} \frac{1}{s_i} \left[(z_i^t wa)^2 + (z_i^t wb)^2 \right] \right\} \qquad (6)$$

$$\equiv -\sum_{i=1}^{m} \frac{1}{s_i} \left[z_i^t H z_i \right] \equiv \frac{1}{2} z^t G z$$

$$\text{,where } S = \left\{ z_{ij} \in \{0,1\}, \sum_{i=1}^{m} z_{ij} = 1, \text{and} \sum_{j=1}^{n} z_{ij} w_j = s_i \right\}$$

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

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

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

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

$G =$ a negative semi-definite symmetric matrix

$=$ Hessian Matrix of objective function (4)

Algorithms development: Observing that problem (6) is minimizing concave function or maximizing convex function subject to convex set of constraints, the optimal solutions occur at the extreme point of the convex set[8]. Therefore, the proposed algorithm is based upon extreme point ranking method, which is a specific algorithm for this kind of problem. Moreover, since the problem is 0-1 quadratic integer programming, then logic based method will be combined to reduce the number of variables by fixing value of some variables. After that, the next adjacent vertices will be explored by exchanging cluster of customers corresponding to the balanced transportation constraints and then be ranked using extreme point ranking approach. These methods and techniques added into the algorithm can be described as follows.

Extreme Point Ranking Method: The basic idea of extreme point ranking is ranking 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. The initial vertex is found using the same method as Gupta et al.'s method[9], while the next adjacent vertices are found using the proposed techniques. At each step, linear integer programming problem P2 shown below is solved to provide lower bounds on the objective function values of the quadratic integer programming problem P1 while the upper bound can be easily updated by substituting this solution in $f(z)$.

$$P2: \underset{z\in S}{\text{Minimize}}\ g(z) = \frac{1}{2}Uz \qquad (7)$$

U_c (the cth column of matrix U) is found by solving

$$U_c = \underset{z\in S}{\text{Minimize}}\ z^t G_c \qquad (8)$$

,where G_c *is the c - th column of* G.

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

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

Proof: Let $S = \left\{ z_{ij} \in \{0,1\}, \sum_{i=1}^{m} z_{ij} = 1, and \ \sum_{j=1}^{n} z_{ij} w_j = s_i \right\}$

Since each U_c is obtained as an optimal solution, then $U_c = \min z^t G_c \le z^t G_c$ .Multiplying $\frac{1}{2}\times z$ to both sizes, the equation becomes

$$\frac{1}{2}U_c z \le \frac{1}{2} z^t G_c z \rightarrow g(z) \le f(z); \forall z \in S$$

The above equation shows that, for all feasible solutions in *S*, *P2* will not give the objective function value over that of *P1*. Thus, *P2* is the lower bound of *P1*.

Proposition 2, proven below, explains how ranking feasible solutions for *P2* can provide the optimal solution to *P1*.

Proposition 2: Let z_r be rth extreme points ranked in ascending order of the objective function value of $g(z)$ and T^k be the set of all extreme points collected from the 1st to the kth set of adjacent extreme points. If $g(z_r) \ge \min\{f(z): z \in T^k\} = f(z^*)$, then z^* is an optimal solution of *P1*.

Proof: Since $g(z_r)$ is the value of $g(z)$ at the rth best integer feasible solution of *P2*, then $g(z_v) > g(z_r)$; $\forall v \ge r+1$. Therefore,

$$f(z_v) \ge g(z_v) > g(z_r) \ge \min\{f(z): z \in T^k\} = f(z^*)$$

That is $f(z_v) > f(z^*)$; $\forall v \ge r+1$. This means that $f(z^*)$ is the least among the values of $f(z)$ at all of the integer feasible solutions in S. Thus, z^* is an optimal solution of problem *P1*.

To find U_c and $g(z)$, Hessian matrix generally found by calculus, is needed. In this paper, the algebraic method is used to construct the closed form solution to each component of the Hessian matrix. The algebraic method arises from finding the closed from solution of the partial derivative terms corresponding to each $z_{ij}, \forall i = 1,...,m; j = 1,...,n$. The closed form solutions are $\frac{\partial f(z)}{\partial z_{ij}^2} = \frac{2w_j^2(a_j^2 + b_j^2)}{s_i}$ and $\frac{\partial f(z)}{\partial z_{ij}\partial z_{ik}} = \frac{2w_j w_k(a_j a_k + b_j b_k)}{s_i}$. Then, plugging these closed form solutions into their related position in the original Hessian matrix shown below.

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

Logic Based Method: The objective of using logic here is to tighten bound of some variables. Regarding to 0-1 quadratic problem, tightening bound of variables is equivalent to fix value of variables. This means that the number of variables to branch will be reduced. Therefore, the computational time is expected to be decreased. The logic used here is fixing $z_{ij} = 0$, if $w_j > s_i$. Observe that this logic will work only when some facilities have capacity limitation less than maximum amount of products shipped: $s_i < \max\{w_j; j = 1,...,n\}$.

Exchanging Method: Although the existing method for finding the next adjacent extreme points in quadratic integer programming problem is cutting plane method[9], it was shown to be non-convergent due to cycling or need an infinite sequence of cutting planes by Zwart[10]. The reason for the cycling behavior as well as non-convergence of these approaches lies in the fact that although the approaches generate cones during the algorithm, they failed to explicitly incorporate these cones into the remaining step. This is essential to avoid the reemergence of vertices that have already been considered. 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 0 and l, which means changing cluster of customers, affects both demand and supply constrains. To conserve the balance of the constraints a customer can move from a current facility to other 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 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 vector of variable. Therefore, no cycling emerges. The exchanging method can be summarized as follows. Let t be number of customers considered to be moved out at a time. At the k[th] adjacent extreme point, t customers running from 1 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.

The proposed heuristics, called EPR algorithm, can be summarized step by step as follows.

INPUT: Locations (a_j, b_j) of customers j on continuous plane, capacity s_i of facility i; $i = 1,...,m$ and demands w_j of customer j.

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^*. 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.
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 upper bound respectively.

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

According to the exchanging method, not all possible (but some high possibility to be an optimal solution) extreme points are considered. Therefore, this optimal condition

may not be satisfied and then 2 following additional stopping rules are constructed.

- No more possible exchanging pairs of customers exist.
- There is no improvement of f_u within q=2 consecutive sets of adjacent extreme points. Note that q can be more than 2. But, the higher value of q, the more computational effort is required.

If at least one of these 2 additional stopping rules is satisfied, the existing current best solution is the final solution. And, $f^* = f_u$. If $g(z_r) < f_u$ and the additional stopping rules are not satisfied, then replace f_l by $g(z_r)$.

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

Verification and Validation of the Solutions: To verify and validate the proposed algorithm the numerical experiments are done. With MATLAB, both the proposed algorithm and exact algorithm will be applied to these sets of data. The solutions and computing time will be compared. The selected exact algorithm is linearization algorithm proposed by Glover and Woolsey[11]. To enhance the efficiency of exact algorithm, logic based method will be combined. With linearized objective function of (3), the problem becomes

$$\text{Minimize } -\sum_{i=1}^{m}\frac{1}{s_i}\left[\sum_{j=1}^{n} w_j^2\left(a_j^2+b_j^2\right)z_{ij} + 2\sum_{j=1}^{n-1}\sum_{k>j}^{n} w_j w_k\left(a_j a_k + b_j b_k\right)z_{ijk}\right] \quad (9)$$

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

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

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

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

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

$z_{ijk} = z_{ij} z_{ik}$

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 on plane

Observe from problem (9) that the last constraint is the least effective constraint. The experiments for solving the problems with and without the last constraints were done with various numbers of facilities and customers and the results show that the solutions for both of them are equivalent. Therefore, the last constraints will be ignored to reduce size of constraints when the problem is solved with "bintprog". 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 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 last constraint will be cut off and the constraints control value of z_{ijk} will remain only $-z_{ij} + z_{ijk} \le 0$ and $-z_{ik} + z_{ijk} \le 0$.

There are $mn(n-1)/2$ additional variables compared with problem (6) solved by the proposed algorithm. Problem (9) with $mn(n+1)/2$ variables will be solved under branch and bound approach using command "bintprog" of MATLAB. The solutions obtained from this algorithm will be sequentially compared with ones obtained from the proposed algorithm.

RESULTS AND DISCUSSION

This part shows the results of the numerical experiments and discussion on these results. It is organized as follows. The first section analyzes the effect of Hessian construction time on processing time in order to ensure that most of processing time is devoted to solving the quadratic integer programming problem not to developing Hessian matrix. For second section, the results of numerical experiments with various problem sizes will be shown and discussed.

The effect of Hessian construction time on processing time: The 5 sets of input data of 13 problem sizes are randomly generated by fixing number of factory at m = 2 factories and varying number of customers (n). And, the number of variables, which equal the multiplication of m and n, runs from 10 to 240 variables. The measure of processing time will be divided into two parts: Hessian construction time and execution time. These times corresponding to problem sizes are shown in Table 1.

Table 1: Effect of Hessian construction time

Problem no.	Number of variables = mxn	<1> Hessian Construction Time (sec)	< 2 > Execution Time (sec)	Process 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

Observe from Table 1 that time used to construct Hessian matrix is still less than execution time if number of variables are not over 150 variables. For problems whose number of variables is not less than 150 variables, most of process time are devoted to constructing Hessian matrix. The Hessian matrix is theoretically developed by using calculus method, which is doing double partial derivative corresponding to each variable. Unsurprisingly, it takes much longer time developing Hessian matrix when even small number of variables increase. Time used for developing Hessian matrix can be reduced by using algebraic method instead of calculus method. To verify and measure the efficiency of constructing Hessian matrix using algebraic method, the same set of experiments will be solved by the same algorithm but new Hessian construction method. Process time using new Hessian construction method compared with the old one (from Table 1) can be shown in Fig 1.

According to Fig.1, even the problem size grows, time to construct Hessian matrix by using algebraic method does not appear. Thus, process time is used only for solving problem and it seems to be equal to execution time of using calculus method to construct Hessian matrix. On contrary, time to construct Hessian matrix by using calculus method increases rapidly when problem size grows. Using algebraic method to construct Hessian matrix is very efficient. As a result, it will be used to develop Hessian matrix from now on.

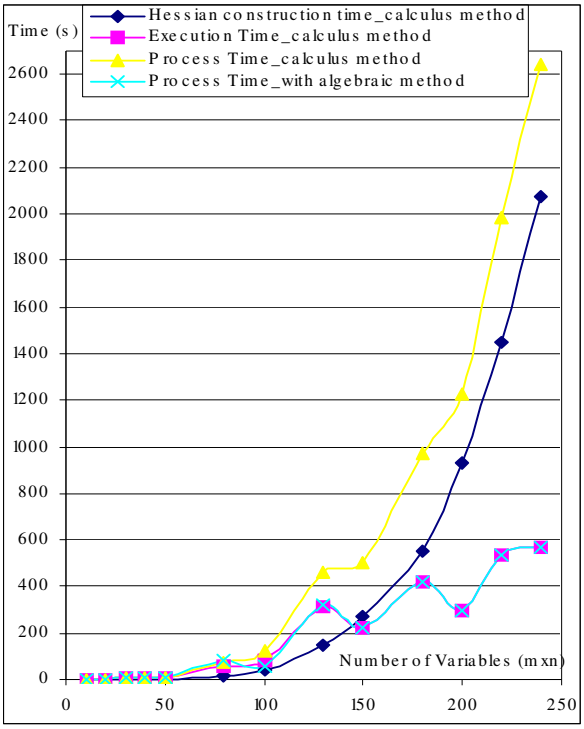

Fig. 1: Comparison of processing time of the two methods of constructing Hessian matrix

The results of evaluating EPR algorithm: The numerical experiments are constructed with various problem sizes, which number of variables corresponding to EPR algorithm = $m \times n$ no more than 1000 variables. The problem sizes vary from (m, n) = (2, 500) to (20, 50). For each problem size, 10-100 sets of data are generated and then solved by two algorithms: EPR algorithm and linearization algorithm. Both algorithms are coded with MATLAB and use command "bintprog" at solving step. The processing time and the following % of error for every case of EPR algorithms are collected, but only average of them will be reported.

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

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

Table 2: Processing time of both algorithms and % of error of EPR algorithm

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)	error (%)	Processing Time (sec)	error (%)	Processing Time (sec)	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
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
32	6	7	168	7.64	42	3.29	25.36	3.29	0.00	3.29	0.00	1.17
33	6	9	270	193.59	54	173.91	17.65	173.91	0.00	174.17	0.00	1.50
34	6	12	468	5471.55	72	859.23	0.00	859.43	0.00	859.46	0.00	2.00
35	6	15	720	15597.23	90	5711.84	-41.59	5711.85	-43.03	5711.89	-43.03	2.50
36	7	9	315	147.56	63	39.81	5.72	39.82	0.00	39.84	0.00	1.29
37	7	11	462	2146.66	77	601.27	-18.28	602.97	-18.62	603.97	-18.62	1.57
38	7	12	546	6497.42	84	742.01	-34.44	743.21	-35.21	745.07	-35.21	1.71
39	8	10	440	209.34	80	79.24	9.01	79.24	0.00	79.24	0.00	1.25
40	8	11	528	4998.91	88	1653.44	-26.99	1653.49	-26.99	1654.13	-26.99	1.38
41	8	12	624	7848.46	96	1821.29	-21.74	1825.29	-28.39	1831.97	-28.39	1.50
42	20	50	25500	-	1000	9863.45	-	9887.45	-	9923.95	-	2.50
99.99 % Confident interval of average percentage of error						-10.83 $\pm$ 17.91 %		-19.129 $\pm$ 14.42 %		-19.40 $\pm$ 14.34 %		

Number of variables	Terminated Time
$400 \le m \times n \times (n+1)/2 \le 750$	5 hrs = 18,000 sec.
$750 < m \times n \times (n+1)/2 \le 1000$	8 hrs = 28,800 sec.
$1000 < m \times n \times (n+1)/2$	24 hrs = 86,400 sec.

The average processing time of both algorithms and average % of error of EPR algorithm are summarized in Table 2. In Table 2, 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 cases in the problem size are premature terminated. There are 10 sets of data for these both cases while 100 sets of data were generated for the other problem sizes. Processing time and % of error of EPR algorithm are separately measured into three parts: processing time to obtain initial solution (z_0), incumbent solution (z_0^*), and final solution ($z_r; z \in T^k$), so that the improvement rate of solution can be observed.

According to Table 2, EPR algorithm not only provides better solution, but also utilizes much less computational time than linearization algorithm for all problem sizes. Observe that, for all sizes *m* of facility processing time of linearization algorithm grows exponentially when *n* increases even there are premature terminated cases, while that of EPR algorithm almost disappear when *m* is small and grows linearly when *m* is larger.

The proposed movement mechanism in EPR algorithm works very well. Moving from z_0 to z_0^* by examining z_c, is very efficient and effective. It can reduces much % of error with no difference of processing time appear. Moving from z_0^* to final solution $z_r; z \in T^k$ and proposed additional stopping rules also works efficiently and effectively because the EPR algorithm stops at optimal solutions or good solutions, compared with solutions from linearization algorithm 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 algorithms. For the linearization algorithm, the premature terminated cases can stop at optimal condition with much less time after combining logic based method.

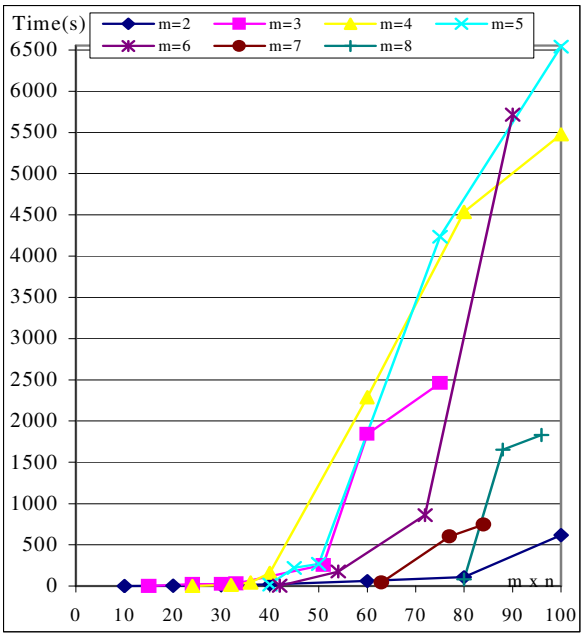

Fig. 2: Processing time of EPR algorithm

To observe the efficiency and effectiveness, the interesting graph, shown in Fig. 2, of processing time of EPR algorithm corresponding to number of variables are studied. Observe from Fig. 2 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 lies above graph of *m-1*. After studying in detail, this condition can explain 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 high possibility to occur the cases that logic based method can be used (some $s_i < \max\{w_j; j = 1,...,n\}$). By numerical experience, this condition usually occurs when. $n/m < 2.5$. When n/m over this value, the effect of increasing number of facility m will obviously expressed. Also, this reason supports summation derived from Table 2 that even the increasing number of customers makes increasing processing time but it has less effect than increasing number of facility. Conversely, for linearization algorithm, increasing n has higher effect than increasing m.

CONCLUSION

The EPR algorithm can solve CMLCP, which is maximizing concave quadratic integer programming problem with balanced transportation constraints, more efficiently and effectively than linearization algorithm. Using much less processing time it provides solutions with % error = -19.40 $\pm$ 14.34 % with 99.99% confidence. In additional, thanks to number of variable to solve of EPR algorithm is $m \times n$ not $m \times n \times (n+1)/2$ like linearization algorithm, it can solve large sized problems $(100 < m \times n \leq 1000)$ that linearization algorithm cannot solve. The EPR algorithm can improve solutions in high rate and stop at optimal solution or good solution compared with non-premature terminated or premature terminated cases of linearization algorithm 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 balanced transportation constraints. In additional, logic based method works very well to reduce the processing time of both algorithms. Thanks to processing time of EPR algorithm depends on increasing m more than n, the algorithm is appropriate to use in the realistic problem that n much more than m.

REFERENCES

1. Nauss, M.R. 1978. An Improved Algorithm for the Capacitated Facility Location Problem. J. Operational Research Society, 29(12): 1195-1201.
2. Sa,G. 1969. Branch-and-Bound and Approximate Solutions to the Capacitated Plant Location Problem. Operations Research, 17: 1005-1016.
3. Akino, U. and B.M. Khumawala. 1977. An Efficient Branch and Bound Algorithm for the Capacitated Warehouse Location Problem. Management Science, 20: 822-844.

4. Mulvey, J. and Beck, M. 1984. Solving Capacitated Clustering Problems. J. of Operations Res., 18: 339- 348.
5. Koskosidis, I. and W.B. Powell. 1992. Clustering Algorithms for Consolidation of Customer Orders Into Vehicle Shipments. Transportation Res., 26B: 365-379.
6. Lorena, L.A.N. and E.L.F. Senne. 2003. Local Search Heuristics for Capacitated p-Median Problems. Home Page: http://www.citeseer.istpsu.edu/306404.htm
7. Sherali, H.D. and Tuncbilek, C.H. 1992. A Squared-Euclidean distance Location Allocation Problem. Naval Res. Logistic, 39: 447 –469.
8. Floudas, C.A. and V. Visweswaran. 1995. Quadratic Optimization. Home Page: http://citeseer.ist.psu.edu/26184.html.
9. Gupta, R. and Bandopadhyaya.L, and Puri,.M.C. 1996. Ranking in Quadratic Integer Programming Problems. European Journal of Operational Research. 95:231-236.
10. Zwart,P.B. 1973. Nonlinear programming: Counterexamples to two global optimization algorithms. Operations Res. 21(6): 1260-1266.
11. Glover,F. and Woolsey , E. 1974. Converting the 0-1 Polynomial Programming Problem to 0-1 Linear Program. Operations Research.22: 180-182.