The Linear Constrained Maximum Capacity Path Problem

Peerayuth Charnsethikul
Krit Virojsailee

Operations Research and Management Science Units
Industrial Engineering Department, Kasetsart University
Bangkok 10903, Thailand

Abstract

The multi-linear constrained maximum capacity path problem is to search a directed path P^* with maximal capacity $C(P^*) = \min\{c_{ij} \mid (i,j) \in A(P^*)\}$ where (i,j) is an arc on the path represented by the set of sequential arcs $A(P^*)$ with c_{ij} as the arc capacity and moreover, the path has to satisfy additional constraints represented as $t_l(P^0) = \sum_{\{(i,j):\,(i,j) \in A(P^*)\}} t_{ijl} \le T_l$, $l = 1,2,..,a$ where t_{ijl} is the arc parameter according to constraint l with T_l as the given upper bound and, a, as the number of constraints. The purpose of this study is to develop an exact algorithm for solving this problem with proof of correctness and complexity. The proposed technique is based on the concept of applying the modified K shortest path as an iteration process to enumerate and search for a feasible path with optimal capacity. Some preliminary computational study has been conducted using four structural network types: Transshipment, Equipment Replacement, Sparse and Fully Dense Networks. The results indicate that the algorithm performs well on average. More than 85% has been exactly solved within reasonable time. All factors derived from theoretical complexity analysis also illustrate significance influences on computation time.

1. Introduction

Everywhere in our daily lives, networks are apparent. Electrical and power networks bring lighting and entertainment into our homes. Telephone networks permit us to communicate with each other almost effortlessly within our local communities and across regional and international borders. National highway systems, rail networks, and airline service networks provide the means to cross great geographical distances to accomplish work. Computer networks, such as airline reservation systems, have changed the way to share information and conduct business and personal lives. In all of these problem domains, and in many more, the entity (electricity, a consumer product, a person or a vehicle, a message) is moved from one point to another in an underlying network according to the objectives (such as minimum cost flow, shortest path, maximum flow) and not exceeding the constraint limits. There are numerous ways to solve these network flow problems. Although the effective technique has been used to solve, those problems, however, there are still some encountered problems, such that, the total computation time is larger than available time. Consequently, researchers have to consider improving the running time behavior of algorithm as a significant factor in network flow problems.

Traditionally, the problems can be solved by simply enumerating the set of possible solution and choose the best one. Unfortunately this approach is not practical since the number of possible alternatives can be very large. So instead, the algorithm whose running time is guaranteed not to grow very fast as the underlying network becomes larger, is devised. One approach has been used in order to measure the computation time is worst-case analysis which measure the increasing rate of computation time when the size of problem is large.

Generally, a network consists of a set of nodes and a set of arcs connecting among nodes. The nodes also referred to as vertices or points. The arcs are also called edges, links, lines or branches. Nodes of a network can represent highway intersections, power solutions, railroad, airline terminals etc. In common, a node can represent a point where a kind of flow is originated or terminated. For this reason, a node can be viewed as a branching point in a typical network. Arcs of a network can represent roads, power lines, airline route etc. In some instances, the arcs have no physical meaning but serve to direct flow in a logical sequence, or to maintain a specified precedence relationship.

The maximum capacity path is to send as much flow as possible between two special nodes, a source node and a sink node within a network without multi-arcs and loop, without exceeding the capacity of any arc. In the application, however, the sole maximum capacity path problem is not always significant since the additional constraint would be considered such as time constraint. Thus, Thienpaitoon Nopparat (1997) developed the algorithm, The Constrained Maximum Capacity Path Problem (CMCP), to find maximum capacity through a network within a time constraint and it can be solved within the polynomial time.

From the above, CMCP algorithm can solve the maximum capacity problem only one constraint. In some cases, it requires searching the maximum capacity through a network within the constraints, which is more than one constraint. For the multi-linear constrained maximum capacity path problem, there is a directed network G(N,A) which for every directed arc(i,j) $\in$ A, there is an arc capacity C_{ij} and arc for each constraint and we denote A(P) is a set of all

arcs in P. It intends to seek for the directed path P^* from the start node to the destination node that its path $c(P^*) = \min \{c_{ij} : (i,j) \in A(P^*)\}$ is maximum and each path constraint $t_l(P^*) = \sum \{(i,j): (i,j) \in A(P^*)\} t_{ijl}$ is less than or equal to the available value of constraint (T_l), l =1,2,..a, where a is the number of constraints. The objective of the research is to develop an efficient algorithm, which solves The Multi-Linear Constrained Maximum Capacity Path Problem. Then analysis on both the worst-case bound and empirical studies of the algorithm will be presented.

2. Scope

In this research, directed network consists of the arc capacity, which is more than zero, and the arc constrained which is more than or equal to zero. After an algorithm is constructed, the algorithm with 4 types of network which each type of network has different characteristics such as number of arcs are tested. And the parameter is adjusted for checking how these characteristics are related to the running time. The types of network can be described as follows.

2.1 Transshipment Network

The network is arranged as a matrix, with a number of rows and columns. The leftmost node is the source node and the rightmost node is the destination node. Among these nodes, there are directed paths connected to the other nodes in the next column. Figure 1 gives an example of Transshipment Network. For this network, N = {1,2,3,4,5,6} and A = {(1,2), (1,3), (2,4), (2,5), (3,4), (3,5), (4,6), (5,6)}.

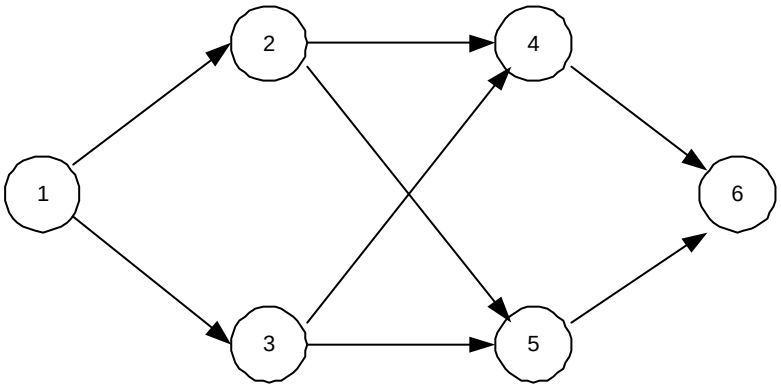
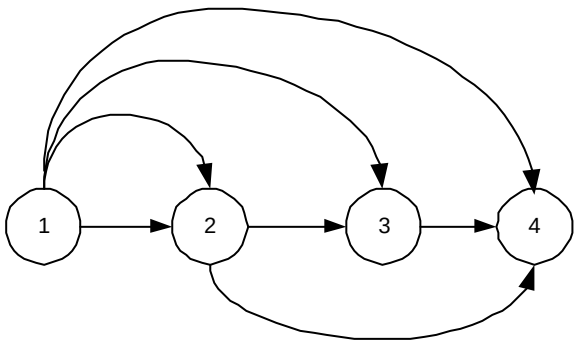


Figure 1 Transshipment Network

2.2 Equipment Replacement Network

The nodes are arranged in sequence. The lower order node has the directed arcs to every higher order nodes. But the higher order node has no directed arc to the lower order node. Figure 2 gives an example of Equipment



Replacement Network for this network, N = {1,2,3,4} and A = {(1,2), (1,3), (1,4), (2,3), (2,4), (3,4)}.

Figure 2 Equipment Replacement Network

2.3 Fully Dense Network

Every node in the network has the directed arcs connected to all other node in the network. Figure 3 gives an example of Fully Dense network. For this network,
N = {1,2,3} and A = {(1,2), (1,3), (2,1), (2,3), (3,1), (3,2)}

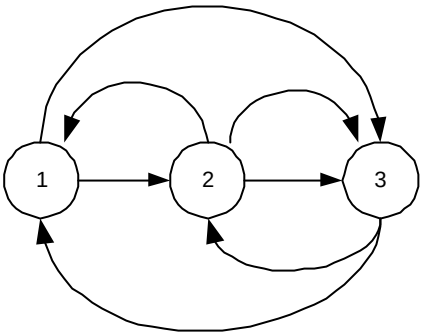


Figure 3 Fully Dense Network

2.4 Sparse Network

The network is defined from the number of nodes and arcs. The directed path will be connected to all other node in the network but the number of arcs is less than number of arcs in Fully Dense Network. Figure 4 gives an example of Sparse Network. For this network, N = {1,2,3,4,5,6} and A = {(1,2), (1,3), (2,3), (2,4), (2,5), (3,1), (3,4), (4,6), (5,4), (6,5)}.

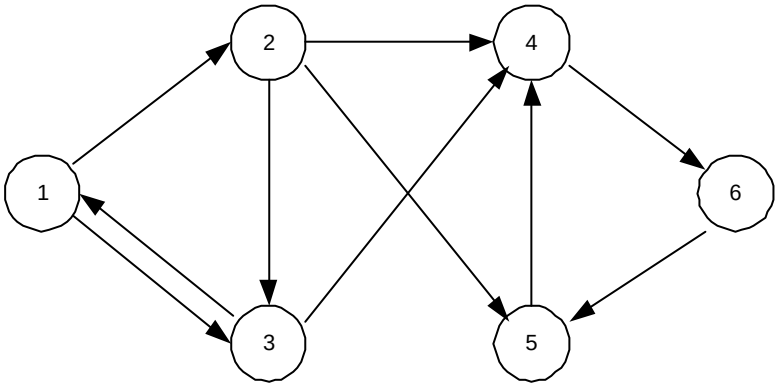


Figure 4 Sparse Network

3. Literature Reviews

3.1 Shortest Path Algorithms

The network flow literature typically classifies algorithmic approaches for solving shortest path problems into two groups: label setting and label correcting. Label setting algorithm designates one label as permanent (optimal) in each iteration. In contrast, labels correcting algorithm consider all labels as temporary until the final step, when they all become permanent.

Ford (1962): The generic label correcting algorithm maintains a set of distance labels d(˙) at every stage. The label d(j) is either ∞, indicating that a directed path from the source to node j has been yet discovered, or it is the length of some directed path from the source to node j. For each node j, also maintain a processor index, pred(j), which records the node prior to node j in the current directed path of length d(j). At termination, the processor indices allow us to trace

the shortest path from procedure for successively updating the distance labels until satisfy the shortest path optimality conditions.

Dijkstra (1959): Dijkstra's algorithm find the shortest paths from the source node s to all other nodes in the network with nonnegative arc lengths. Dijkstra's algorithm maintain a distance label d(i) with each node i, which is an upper bound on the shortest path length to node i. At any intermediate step, the algorithm divides the nodes into two groups: permanent and temporary. The distance label to any permanent node represents the shortest distance from the source to that node. For any temporary node, the distance label is an upper bound on the shortest path distance to that node. The basic idea of the algorithm is to fan out from node s and permanently label nodes in the order of their distances from node s. Initially, assign node s as a permanent label of zero, and each other node j a temporary label equal to ∞. At each iteration, the label of node i is its shortest distance from the source node along a path whose internal nodes are all permanent. The algorithm selects a node i with the minimum temporary, make it permanent, and reaches out from the node that is scanned arc A(i) to update the distance labels of adjacent nodes. The algorithm terminates when it has designated all nodes as permanent.

3.2 The Maximum Capacity Through a Network

Pollack (1960) defined the maximum capacity route to be route between any two given cities (nodes) that allows the greatest flow. It differs from the maximum capacity problem in that only the single best route is desired. In the maximum capacity problem, the objective is to find the maximum minimum flow between two nodes using as many different routes as needed. The computational algorithm for solving the maximum capacity route can be modified from the shortest route algorithm.

3.3 A Dual Algorithm for the Constrained Shortest Path Problem

Handler and Zang (1980) develop a Lagrangian relaxation algorithm for the problem of finding a shortest path between two nodes in a network, subject to a knapsack-type constraint. The problem is generic to a class of problems that arise in the solution of integer linear program and discrete stage deterministic dynamic program.

3.4 The K Shortest Path problem

In the communication and transportation networks, it is sometimes desirable to have knowledge of several shortest paths, arranged in increasing order according to their lengths. Dreyfus (see reference in Phillips and Garcia-Diaz (1981)) proposed the K shortest path algorithm to provide an alternative approach for planning when the best solution is not practical.

The general version of the K shortest path problem admits cycles in the paths and ties between the path lengths. Several relaxation of the problem is possible in order to provide alternative solution procedures to specific network problems of interest. The new method exploits a fairly strong analogy between the solution to the general K shortest path problem and the solution of a system of ordinary linear equations. It is known as the double sweep method, simultaneously calculates the K shortest path lengths from a particular source node to all other nodes in the network.

3.5 The Constrained Maximum Capacity Path Problem

Theinpaitoon Nopparat (1997) proposed algorithm, Constrained Maximum Capacity Path (CMCP)" for solving the maximum capacity problem within time constraint. The method starts by seeking for the shortest path by Dijkstra algorithm from start node to destination node. If the shortest path is less than the upper bound, a problem has feasible solutions then it seeks for the maximum capacity from the start node to the destination node by the maximum capacity path algorithm and starts calculation at the first iteration. At each iteration b, the algorithm starts from seeking for the capacity constrained shortest path from the start node to the destination node, which is the directed path that has the least path time with path capacity C_b. If the value of time constraint is less than the upper bound, then the algorithm terminates and we get the optimal capacity within time constraint but if not the algorithm will find next capacity that $C_{b+1} = \max\{c_{ij} : (i,j) \in A, c_{ij} < C_b\}$ and proceed at iteration b+1 until the stopping condition will be detected. The worst case complexity of algorithm CMCP is $O(mn^2)$ where m is the number of arcs and n is number of nodes.

4. Methodology

In practice, there are several steps in order to develop The Multi-linear Constrained Maximum Capacity Path algorithm. The detail of these steps can be described as follows

4.1 Mathematical Model Formulation

4.1.1 Notation

The following notation are used for model formulation

N	The number of nodes
N_s^-	Set of nodes leaving from the start node

N^-_k Set of nodes leaving from node k

N_k Set of nodes entering to node k

N^+_d = Set of nodes entering to the destination node

C_{ij} = Arc capacity from node i to node j

t_{ijl} = Path constraint value on the arc connecting from node i to node j for l^{th} constraint

T_l = The upper bound value of l^{th} constraint

X_{ij} = { 1 if link i,j is included in the path connected from the start node to the destination node

0 otherwise }

4.1.2 Formulation

The basic mathematical formulation for the Multi-Linear Constrained Maximum Capacity Path Problem is an integer program with the objective as follows.

Maximum C (1)

Subject to

$$C \leq C_{ij} X_{ij}, \forall_{ij} \in A \quad (2)$$

$$\sum_{\forall_j \in N^-_s} X_{sj} = 1 \quad (3)$$

$$\sum_{\forall_i \in N^+_k} X_{il} - \sum_{\forall_j \in N^-_k} X_{lj} = 0, \forall l \in N \quad (4)$$

$$\sum_{\forall_j \in N^+_d} X_{jd} = 1 \quad (5)$$

$$\sum \sum t_{ijl} X_{ij} \leq T_l, \forall l \quad (6) \qquad \forall_{ij} \in A$$

In the above formulation

- Equation (1) shows the objective function of maximizing the capacity (C).
- Constraint (2) ensures that C is the maximum capacity on the selected path.
- Constraint (3) ensures that only one path can be left from start node.
- Constraint (4) ensures that number of incoming path equals to number of entering path.
- Constraint (5) ensures that only one path can be entered to destination node.
- Constraint (6) ensures that the value of each constraint must be less or equal to the available constraint values.

4.2. Description of the Proposed Algorithm

Two key elements of the proposed algorithm are MCP algorithm and K shortest paths algorithm. MCP algorithm is used for finding the maximum capacity from the source node to the other nodes in the network. Separate it into two sets: S contains node that has found the maximum capacity path and T contains node that has found pseudo maximum capacity path, the maximum capacity path from source node s within set S. At each iteration, it will have at least one node in set T that is the pseudo maximum capacity path to be the maximum capacity path in set S. Then the algorithm will find the pseudo maximum capacity in set T until every node in set T is contained in set S.

Algorithm MCP
begin
S := f ; T := {1,2,..,n};
pred(i) := 0 and c(i) := 0 for each node i Î T ;
c(s) := ∞ ;
while T ≠ ϕ do
begin
if s ∈ T then i := s else
let i ∈ T be a node for which c(i) = max{c(j) : j ∈ T, pred(j) ≠ 0};
S:= S ∪ {i};

```
        T:= T – {i};
    for each (i,j) when j ∈ T and (i,j) ∈ A(i) do
      if c(j) ≤ min{c(i), c_ij} then
        c(j) : = min{c(i),c_ij};
          pred(j) := i;
  end;
end;
```

When the algorithm terminated, it illustrates the value the maximum capacity path from source node to destination node. And the maximum capacity path of any node i can be traced from array pred(i), predecessor node i, which has the maximum capacity equal to c(i). From MCP algorithm, assume that network must have path from source node s emanates to other node at least one path otherwise the algorithm will terminate when every node in set T has array pred(i) = 0. Therefore only node in set S can find the maximum capacity path but node in set T has no any direction from source node S to these nodes, thus it cannot find the maximum capacity path in the network.

The MCP 's algorithm can be allocated to three basic operations:

1. Beginning: define $\forall i \in T$, pred(i) := 0, c(i) := 0 and c(s) := ∞
2. Node selection: select $i \in T$ that give maximum c(i) and pred(i) $\neq 0$ into set S
3. Capacity update: node which is contained in set S will update c(j) for node $j \in T$ when $(i,j) \in A(j)$. If $c(j) \leq \min\{c(i),c_{ij}\}$, adjust $c(j) := \min\{c(j),c_{ij}\}$ and pred(j) := i

At each iteration k, it show the direct out tree called pseudo maximum capacity path tree $t_k = (N_{tk}, A_{tk})$ that has root at source node s. Node $i \in N_{ik}$ when pred(i) $\neq 0$ (except node s) and arc $(i,j) \in A_{tk}$ when pred(j) = i . Node in tree t_k can be allocated into permanent node and temporary node. Permanent node is a node $i \in S$ except node s in t_k which pred(i) $\neq 0$. Temporary node is a node $i \in T$ which pred(i) $\neq 0$. For the nodes that lie outside t_k are nodes in set T which pred(i) = 0. These nodes do not have any pseudo path from node s in set S. So there is no pseudo maximum capacity path. The MCP algorithm will terminate when node d is selected into set S then the path from node s to node d is the maximum capacity path. From the above three basic operations, the computation time can be analyzed as follows:

1. Beginning: perform the operation n time for giving all c(i) = 0 and pred(i) = 0. For c(s) = ∞, operate O(1) time. Therefore, the total operation is O(n) + O(n) + O(1) = O(n)
2. Node Selection: finding node $i \in T$, which has maximum c(i). Each operating time is O(n) and total iteration is n times, so the operation time for node selecting is O(n)*n = $O(n^2)$.
3. Capacity update: The algorithm performs this operation $|A(i)|$ times for node i. Overall, the algorithm performs this operation $\sum_{i \in N} |A(i)|$ m times. Since each capacity updating operation requires O(1) time, the algorithm requires O(m) total time for updating all capacity labels . From the above result, MCP algorithm solves the maximum capacity path in $O(n^2)$ time where n is the number of nodes.

Consider a directed network with nodes number one through n. Suppose that for each node there is a vector K Shortest Path lengths from a given source node. Under the assumption that the initial calculation does not underestimate the path lengths, the double sweep method successively reduces the computation time until the optimal vector of path length is achieved in a finite number of iterations.

Each iteration consists of two operations. In the forward operation, the node are considered in ascending order (i.e., j = 1,2,..,n). After identifying the list of nodes i incident to node j, such that i < j, the K shortest path lengths from the source to node j are successively examined to verify if shorter path lengths are possible through the incident nodes. If such path lengths exist, they will be used as new estimates in further iterations. A similar procedure is performed during the backward operation of the algorithm, but in this case the node are considered in descending order (i.e., j = n, n-1,…,1) and only node i > j are investigated.

The feasible paths are the paths that has the capacity less than or equal to the maximum capacity and the summary of the each constraint value must be greater than or equal to the available value of each constraint. The algorithm will check these conditions while performing the K shortest path algorithm. If only consider the original K shortest path algorithm, the worst case condition is $O(Kn^3)$ (see Phillips and Garcia-Diaz (1981)) where K is the number of path lengths to be investigated. While Modified K Shortest Path algorithm has the number of constraints to be considered in the algorithm, thus the worst case condition is $O(Kn^3ma)$, which n is number of nodes, m is number of arcs and a is number of constraints.

The proposed algorithm referred as MLCMCP algorithm can be summarized as illustrated in figure 5 with five steps in order to get the solution as described below:

Step 1: The method starts by seeking the shortest path from node s to node d with Dijkstra's algorithm, the constraint value t_{ijl} is the path distance for checking whether the path from node s to node d that has the shortest path is less than T_l or not. If it is more than T_l, there is no feasible solution.

Step 2: Find the maximum capacity P_0 from node s to node d in the network G with MCP algorithm. The path with the maximum capacity will be the maximum capacity directed path within all directed path in network G and denote $C_1 = c(P_0)$.

Step 3: Then perform Modified K Shortest Path algorithm with the maximum

capacity. If each path has the arc capacity more than or equal to maximum
capacity and the summary of path constraint is less than or equal to the available value of path constraint for every constraint, go to step 4. Otherwise go to step 5

Step 4: Seek the constraint, which has the lowest number of K to be the selected constraint. Pick the highest K^{th} position to be the first considered and calculate the value of the remaining constraints with the selected path. If the value of every constraints is less than or equal to the available value of constraint then the algorithm terminates and get feasible path with the maximum capacity but if not pick the next lower K positions to be the selected path and start calculating at step 3 again. If we perform until the lowest K position and still do not get the solution then go to step 5.

Step 5: Find the next capacity that less than previous capacity ($C_{b+1} = \max\{c_{ij} : (i,j) \in A, c_{ij} < C_b\}$) and go to step 3.

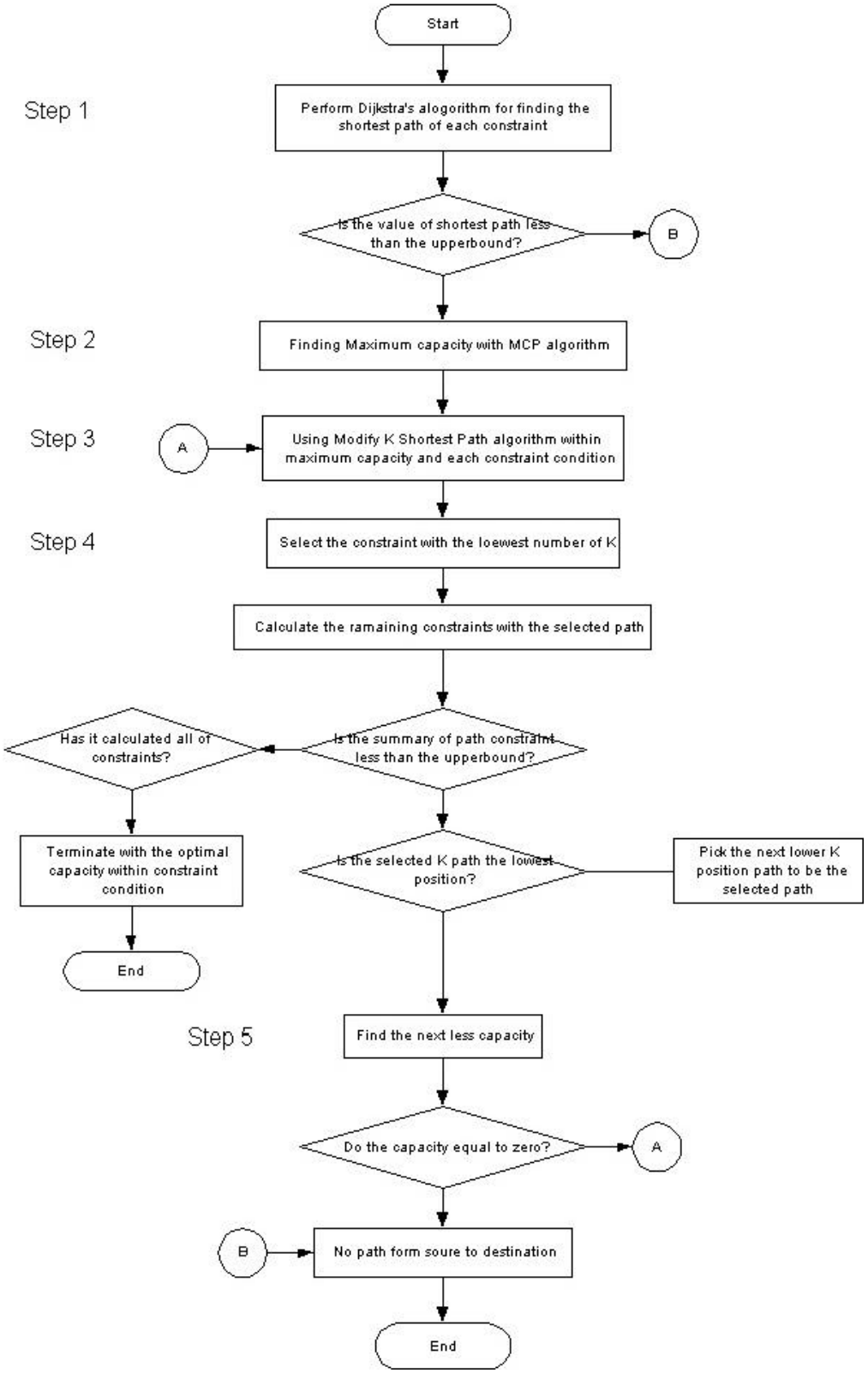


Figure 5 Flow Diagram of MLCMCP Algorithm

The MLCMCP Algorithm can be allocated into 4 parts.

1. Beginning: finding the shortest path for every constraint in the network G from node s to node d for checking whether there exists a feasible path in the network or not. If there is, find the maximum capacity path P_o with the MCP algorithm and define $C_1 = c(P_o)$.
2. Find the feasible path within the constraint condition and the capacity must greater than or equal to C_b with Modified K shortest path algorithm.
3. Find the common path that agrees upon the every constraint condition.
4. Update the new capacity C_b with $C_{b+1} = \max\{(i,j) \in A, C_{ij} < C_b\}$

The algorithm starts at the beginning part then at each iteration b it will start Modified K Shortest Path algorithm and finish at the update of the new capacity C_b. Therefore the number of iteration will not be greater than the number of arcs in the network G. Because C_b is equal to an arc parameter in the network G.

If there exist optimal solutions more than one, MLCMCP algorithm can provide only one solution. So this is a weak point of this algorithm that should be improved for further study. To verify the correctness of the proposed algorithm, the following theorem is proven as follows.

Theorem: Algorithm terminates with the optimal capacity C_b within the constraint condition.

Proof: Let consider Modified K Shortest Path algorithm. At the first iteration, the algorithm is proceeded with the capacity C_1 and constraint condition on every constraint. If there is still no common path met with every constraint conditions, the process repeats until at the b^{th} iteration, if there is a common path met with every constraint conditions while there is no such path at $(b-1)^{th}$ iteration. This common path must be the solution with the optimal capacity C_b.

The computation time can be allocated into 3 parts.

1. Beginning: Check whether the problem has the feasible solution by using Dijkstra's algorithm in the network G from node s to node d, thus time consumption is $O(n^2a)$, where n is number of nodes and a is number of constraints. Next apply the MCP algorithm to find the maximum capacity, which is $O(n^2a)$. Then this part is $O(n^2a) + O(n^2) = O(n^2a)$.
2. Seek the common path by Modified K Shortest path algorithm. For each iteration, the operating time is $O(kn^3)$. Select the path to be considered as the common path of the remaining constraint is $O(k)$ and the number of maximum iteration for evaluation is m (number of arcs). Also include number of constraints, a, in the operating time. The total operating time is $ma[O(kn^3)+O(k)] = O(kn^3ma)$.
3. Update the new capacity C_b: At each iteration, every arc capacity will be checked for finding C_b. The operating time is $O(m)$ and number of maximum iteration is m. So the total running time is $O(m)* O(m) = O(m^2)$.

Therefore, the computation time of MLCMCP algorithm is $O(n^2) + O(kn^3ma) + O(m^2) = O(kn^3ma)$. If consider Fully Dense Network type, The number of paths from node s to node d in the network is (n-2)!. Therefore the worst case, the maximum number of k, is (n-2)!. Hence, the worst-case computation time grows exponentially by size of the problem.

5. Results and Discussions

In order to evaluate the efficiency from the proposed algorithm, the Visual Basic program, was tested on four types of network, which are Transshipment, Equipment Replacement, Sparse and Fully Dense network by using Experimental Design Method with Statistica Software.

The method is experimented in these cases is 2-level design and for saving number of runs, one-half fraction of 2^k design is used. It can provide test of the main effect and two-ways interaction. The value of each parameter was randomly generated from the computer program. The detail of tests are shown below:

5.1 Transshipment Network

There are 6 parameters to be considered: number of rows, number of columns, maximum capacity value, maximum constraint value, proportional of the upper bound to minimum path constraint and number of constraints. Fractional factorial design 2^{6-1} is used for testing these parameters so there are 32 numbers of runs. Table 1 shows the value of low level and high level for six parameters.

Table 1 Data for Transshipment Network

Parameter	Low(-1)	High(1)
Number of Columns	5	10
Number of Rows	5	10
Proportional of the upper bound to minimum path constraint	5	10
Maximum Capacity Value	50	50,000
Maximum Constraint Value	50	50,000
Number of Constraints	1	5

Number of nodes = (number of rows) x (number of columns) + 2
Number of arcs = (number of columns-1) x (number of rows)^2 + (number of rows)x 2

5.2 Equipment Replacement Network

There are 5 parameters to be considered: number of nodes, maximum capacity value, maximum constraint value, proportional of the upper bound to minimum path constraint and number of constraints. Fractional factorial design 2^{5-1} is used for testing these parameters then there are 16 numbers of runs. Table 2 shows the value of low level and high level for five parameters.

Table 2 Data for Equipment Replacement Network

Parameter	Low(-1)	High(1)
Number of Nodes	10	100
Proportional of the upper bound to minimum path constraint	5	10
Maximum Capacity Value	50	50,000
Maximum Constraint Value	50	50,000
Number of Constraints	1	5

Number of arcs = [(number of nodes) x (number of nodes – 1)]/2

5.3 Sparse Network

There are 5 parameters to be considered: number of arcs, maximum capacity value, maximum constraint value, proportional of the upper bound to minimum path constraint and number of constraints. Fractional factorial design 2^{5-1} is used for testing these parameters then there are 16 number of runs. Table 3 shows the value of low level and high level for five parameters.

Table 3 Data for Sparse Network

Parameter	Low(-1)	High(1)
Number of Arcs	250	1250
Proportional of the upper bound to minimum path constraint	5	10
Maximum Capacity Value	50	50,000
Maximum Constraint Value	50	50,000
Number of Constraints	1	5

Number of nodes = 50
Number of arcs: low level = 10% of (number of nodes)^2
High level = 50% of (number of nodes)^2

5.4 Fully Dense Network

There are 5 parameters to be considered number of nodes, maximum capacity value, maximum constraint value, proportional of the upper bound to minimum path constraint and number of constraints. Fractional factorial design 2^{5-1} is used for testing these parameters then there are 16 number of runs. Table 4 shows the value of low level and high level for five parameters.

Table 4 Data for Fully Dense Network

Parameter	Low(-1)	High(1)
Number of Nodes	10	100
Proportional of the upper bound to minimum path constraint	5	10
Maximum Capacity Value	50	50,000
Maximum Constraint Value	50	50,000
Number of Constraints	1	5

Number of arcs = (number of nodes) x (number of nodes -1)

From the running time result based on four types of network, we will show how the running time performs when comparing with worst-case analysis. Previously, the time complexity is O(kn^{3}ma) and the running time result has

the value of number of paths (k), number nodes (n), numb er of arcs (m), number of constraints (a) and running time(y). So we will transform this value into linear regression for seeking the coefficient value of x. If it is less than 3 which means MLCMCP algorithm perform calculation time less than the worst-case time complexity. Table 5 shows value of running time transformed as a function.

Time complexity is $O(kn^3ma)$

f(k,m,n,a)	$= C\, k\, n^b\, m\, n$
ln f(k,m,n,a)	= ln C+ ln k + b ln n + ln m + ln a
ln y	= ln C+ ln k + b ln n + ln m + ln a
ln y - ln k - ln m- ln a	a bx
Z	a bx

Table Running Time Transformation to Linear Regression

	Z	X	No. of Node(n)	No. of Arc(m)	No. of path(k)	No. of constraint(a)	running time (Y sec.)
1	-7.07	1.79	6	8	2000	5	0.1
2	-6.57	2.3	10	90	80	5	50
3	-6.3	3.3	27	110	1	5	1
4	-5.45	3.95	52	235	1	5	1
5	-4.8	4.18	66	528	2000	5	17
6	-3.83	4.60517	100	9900	1	5	43200
7	-3.99	4.62	102	920	1	5	107

From Table , the second and third column represents the z and x value in linear regression respectively which transform from the value of number of paths k , number nodes (n), number of arcs (m), number of constraints (a) and running time(y). Then takes this z and x to construct regression model. Figure 6 shows plot of regression model, the regression model is Y = -9.25 + 1.081X. So the coefficient of x is 1.081 which is less than 3. It can be concluded that MLCMCP algorithm performs the calculation time from designed experiments less than the worst-case time complexity.

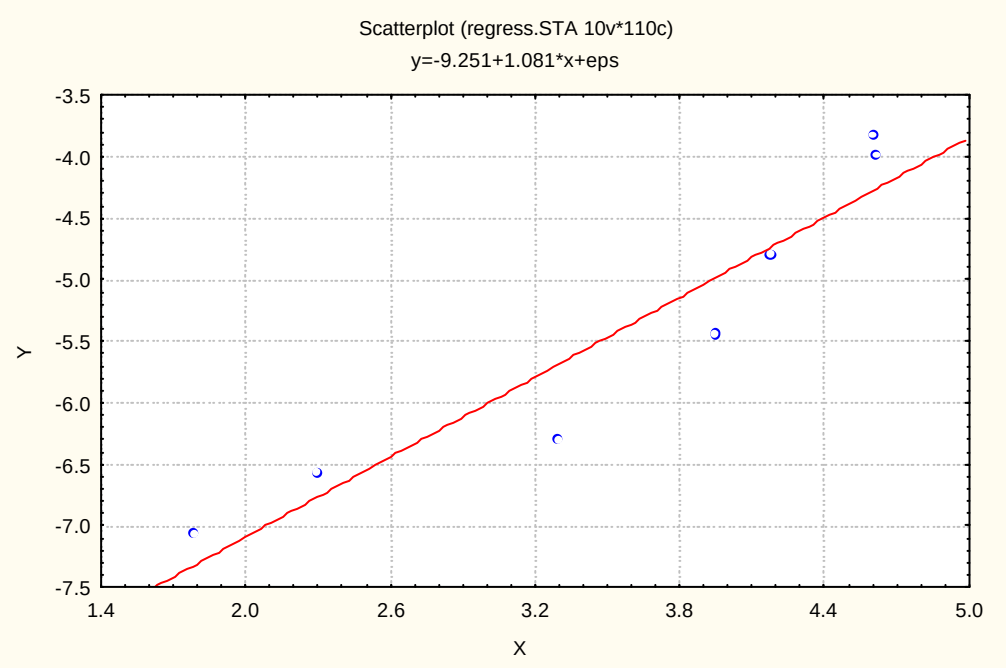


Figure 6 Regression Model

Based upon our four types of network test, in general, the proposed algorithm performs effectively in term of running time. Strengths and weaknesses of the algorithm in various perspectives are discussed as follows.

From 160 experiments (four types of network) 85% of the experiment can be terminated with optimal solution and the average running time is less than one hour. In contrast, 15% of the experiment cannot be solved within 12 hours. Due to size of problem and number of constraint is high, the running time to detect unfeasibility is high according to the fact that the algorithm continues to proceed reduction of the capacity until the least value.

At the previous section, the worst case complexity is $O(kn^3ma)$. Then when we consider the running time of MLCMCP algorithm, the usage time of Modified K Shortest Path algorithm is almost 100% of the total running time. It can be said that most of running time is spent on Modified K Shortest Path algorithm.

The ANOVA analysis illustrates that main parameter: number of nodes, number of arcs, proportional of upper bound to minimum path constraint, maximum capacity value, maximum path constraint value have significant on the running time. However, the case of two-way interaction between network sizes and number of constraints also has high priority effect to the running time.

6. Conclusion and Extensions

A new algorithm has been devised referred as The Multi-Linear Constrained Maximum Capacity Path problem (MLCMCP). The algorithm is intended to find the maximum capacity path under multi-linear upper bound constraint by combine Dijkstra's algorithm, maximum capacity path problem and modified the k shortest path algorithm to be the new algorithm for searching the feasible solution with optimal capacity. The result from 160 experiments shows that MLCMCP algorithm performs well in term of running time when size of problem and number of constraint is small. To improve the efficiency of the algorithm, there are some recommendations for further study as listed follows.

1. Develop an algorithm for find the maximum capacity path with the constraint that has greater than, less than or equal to the available value of constraints.
2. To find the optimal solution, sometimes it takes a long time to get the answer. Thus to avoid this disadvantage, developing heuristic algorithm to find the solution that give the answer close to the lower bound is encourage.
3. In some problems there are optimal solutions more than one. The MLCMCP algorithm, however, will terminate when it founds the first feasible solution. Then, to get more solutions by developing an algorithm that can enumerate all optimal solution should be adopted as an the alternative for handling multiple objectives issues

References

Ahuja, R.K., T.L. Magnanti and J.B. Orlin (1993), "Network Flows Theory Algorithm and Application", Prentice-Hall, Inc., New Jersey.

Dijkstra E. W. (1969), "Note on Two Problems in Connexion with Graphs", Numerische Mathematik 1 (1969), p. 269-271.

Fisher Marshall L. (1981), "The Lagrangian Relaxation Method for solving integer programming problems", Management Science No. 1, January 1981, p. 1-17.

Ford, L.R. and Fulkerson D.R. (1962), "Flow in Networks", 1st ed., Princeton University Press, New Jersy.

Handler Gabriel Y. and Zang Israel (1980), "A Dual Algorithm for the Constrained Shortest Path Problem", Networks, Vol. 10 (1980), p. 293-310.

Hu T. C. (1961), "The Maximum Capacity Route Problem", Operation Research Vol.9 1961, p. 898-900.

Lawler Eugene L. (1976), "Combinatorial Optimization : Networks and Matroids", New York, Holt, Rinehart and Winston.

Papaddimitriou Christos H. and Steiglitz Kenneth (1982), "Combinatorial Optimization: Algorithm and Complexity", Englewood Cliffs, N.J., Prentice-Hall, Inc.

Phillips Don T. and Garcia-Diaz Alberto (1981), "Fundamentals of Network Analysis", Englewood Cliffs, N.J., Prentice-Hall, Inc.

Pollack Maurice (1960), "The Maximum Capacity Through a Network", Operation Research 8 1960, 733-736.

Montgomery Douglas C. (1991), "Design and Analysis of Experiments", John Wiley & Sons Fourth Edition.

Thienpaitoon Nopparat (1997), "The Constrained Maximum Capacity Path Problem", Graduate School, Industrial Engineering, Kasetsart University 1997.