Exact topology discovery algorithm for the capacity and delay constrained loop network

Y.-J. Lee and M. Atiquzzaman

Abstract: We have developed an exact model and algorithm for the delay-constrained minimum cost loop problem (DC-MCLP) of finding broadcast loops from a source node. While the traditional minimum cost loop problem (MCLP) deals with only the traffic capacity constraint served by a port of source node, the DC-MCLP deals with the mean network delay and traffic capacity constraints simultaneously. The DC-MCLP consists of finding a set of minimum cost loops to link end-user nodes to a source node satisfying the traffic requirements at end-nodes and the required mean delay of the network. In the DC-MCLP, the objective function is to minimise the total link cost. We have formulated the DC-MCLP and proposed an exact algorithm for its solution. The proposed algorithm is composed of two phases: in the first phase, it generates feasible paths to satisfy the traffic capacity constraint; in the second phase it finds the exact loop topology through matching and allocating optimal link capacity to satisfy the mean delay constraint. In addition, we have derived several properties including the memory and time complexity of the proposed algorithm. Performance evaluation shows that the proposed algorithm has good efficiency for networks with less than thirty nodes and light traffic. Our proposed algorithm can be applied to find the broadcast loops for real-time multimedia traffic.

1 Introduction

Network topology is defined as the configuration of connecting computers in a network. Topology affects important factors such as communication cost and transmission speed of a network. Thus, topology discovery is an important research area in computer networks.

Modern computer networks consist of backbone networks that serve as major highways to transfer large volumes of communication traffic, and local networks that feed traffic between backbone nodes and end-user nodes connected to the backbone. Several researchers have proposed algorithms for discovering the topology of the Internet backbone [1–5]. A local area network (LAN) connected to a backbone node, can be regarded as an end-user node. A local network, therefore, consists of a backbone node and several end-user nodes hanging off a LAN. Examples of local network are campus networks or enterprise networks that are composed of gateway routers connected to outer backbone networks such as the Internet, and internal users such as host computers, switching hubs etc.

In a local network, the total traffic volume of end-user nodes that can be served by a port of the backbone node is limited. So, the local network consists of a backbone node (source) and several trees or rings that cover all end-user nodes to satisfy the constraints on traffic volume. Topology discovery is required to find all the trees or loops in a local network that satisfies the constraints.

Issues related to topology discovery for local network has been classified into two problems in the literature: capacitated minimum spanning tree problem (CMSTP) [6, 7] and minimum cost loop problem (MCLP) [8]. The CMSTP finds a set of minimal spanning trees with a capacity constraint. In the MCLP, end-user nodes are linked together by a loop that is connected to a port in the backbone node. The links connecting end-user nodes have a finite capacity and can handle a restricted amount of traffic, thus limit the number of end-user nodes that can be served by a single loop. The objective of the design is to form a collection of loops that serve all user nodes with a minimal connection cost [9, 10]. The MCLP is known to be NP-hard [11], and consists of finding a set of minimum cost loops rooted at the source node satisfying the traffic constraint only. The MCLP considers the traffic capacity constraint, but communication delay between end-user nodes deteriorates quality of service (QoS) of users as the network size grows. To solve this issue, it is necessary to consider the traffic capacity and the mean network delay constraints together. However, no previous work has been reported on the delay-constrained minimum cost loop problem (DC-MCLP). We, therefore, present the DC-MCLP in this paper.

A similar problem is the QoS routing problem, which is to select feasible paths that satisfy various QoS requirements of applications in a network. Multiple additively constrained QoS routing problem is referred to as multiple constrained path selection (MCPS) [12–14]. Delay constrained least cost (DCLC) path selection problem [15, 16] is to find a path that has the least cost among the paths from a source to a destination for which the delay remains under a given bound. To solve the above problems, network configurations such as link connectivity, link cost,

doi:10.1049/iet-com:20050063

Paper first received 4th February and in revised form 30th June 2005

Y.-J. Lee is with the Department of Technology Education, Korea National University of Education, Darak-Ri, Gangnae-Myon, Chungwon-Gun, Chungbuk, 363-791, Korea

M. Atiquzzaman is with the School of Computer Science, University of Oklahoma, 200 Felgar Street, Norman, OK 73019, USA

E-mail: lyj@knue.ac.kr

and delay should be available. Alternatively, the proposed DC-MCLP is to find the network configuration composed of several loops to satisfy the network delay constraint. Since the problem definition of DC-MCLP is different from those of MCPS and DCLC, solutions to MCPS and DCLC cannot be applied to DC-MCLP. Another similar problem is the vehicle routing problem (VRP) [17] in transportation science. However, existing algorithms for MCLP and VRP [18–20] do not consider network mean delay. Without modification, they cannot be applied to the DC-MCLP that has additional constraint, that is, the mean delay of network should be within the desired time. Although algorithms [21–24] for the topological design of backbone networks consider the network mean delay, they do not consider the traffic capacity constraint. Thus, they are not useful for the DC-MCLP.

The objective of this paper is to formulate the DC-MCLP and to develop an exact algorithm to solve the problem. We propose a dynamic programming-based exact algorithm for the solving the DC-MCLP. To investigate the feasibility of the algorithm for real networks, we carry out a performance evaluation of the algorithm. Our proposed algorithm solves the DC-MCLP in two phases: in the first phase, the algorithm uses dynamic programming to generate feasible solutions to satisfy the traffic capacity constraint. In the second phase, it finds exact loop topology based on the matching and the optimal link capacity allocation to the set of capacitated loops in order to satisfy the desired mean network delay constraint. Computational complexity shows that our algorithm can be effectively applied to problems where the number of nodes is less than thirty and traffic volume is light.

Our performance evaluation results obtained from simulation of analytical model shows that the execution time of our algorithm takes a maximum 200 seconds for the worst case with 30 nodes and heavy traffic volume. If the number of nodes is less than thirty, or the total traffic volume is not heavy, the execution time becomes less. We, therefore, suggest using our exact algorithm for small local networks (less than thirty nodes), and heuristic methods [10] can be used for large local networks. The main contributions of this paper are: (i) formulation of the DC-MCLP, (ii) proposing and evaluating its exact solution, and (iii) determining the threshold in choosing between heuristic methods and the exact algorithm depending on the size of the local network.

The suggested algorithm reduces the total cost in a short computation time. In addition, it can be applied to the design of synchronous optical network and the finding of broadcast loops from the source node in order to transfer the message to end-user nodes in the local network. An earlier version of this paper was presented in [25].

The rest of the paper is organised as follows. Section 2 presents the mathematical formulation of the DC-MCLP. Section 3 describes our proposed dynamic programming based modelling and exact algorithm for the DC-MCLP. Section 4 evaluates the computational complexity and execution time of our proposed algorithm. Finally, concluding remarks are given in Section 5.

2 Formulation of the DC-MCLP

In this section, we first describe the mathematical formulation of the MCLP, which is related to the DC-MCLP. Then we present the mathematical formulation of the DC-MCLP.

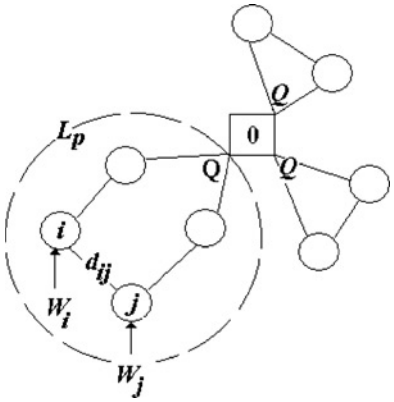

Fig. 1 *MCLP*

2.1 MCLP and DC-MCLP

To aid in better understanding the DC-MCLP formulation, we first familiarise the readers with the formulation of the MCLP. The MCLP (see Fig. 1) is concerned with finding a set of minimal cost loops that are used form the broadcast paths for real time traffic, such as multimedia. The end-user nodes are linked together by a loop connected to a port in the backbone node, where the links connecting the end-user nodes have finite capacity, i.e. they can handle limited amount of traffic (Q). This translates to restricting the number of end-user nodes served by a single loop. The solution of the MCLP results in a collection of loops that serve all end-user nodes with a minimal connection cost.

We now consider the modelling of MCLP. Assume that there is graph, $G = (V, E)$, where $V = \{0, 1, 2, \ldots, n\}$, traffic requirement at node is W_i ($i \in V - \{0\}$) and link cost between node i and node j is d_{ij} ($j \in V - \{0\}$ and $(i, j) \in E$). Q represents maximum traffic served by a single loop. Index 0 represents the source node and can be viewed as an object such as the backbone router with several ports. End-user nodes originate traffic and can be regarded as hosts or switching hubs. The problem formulation of MCLP is described by (1). The objective of the MCLP is to find a collection of least-cost loops rooted at node 0. L_p is the pth loop ($p = 1, 2, \ldots, lcnt$: where $lcnt$ is the number of loops in the solution) whose two of nodes are connected the source node. A particular case occurs when each W_i is equal to one. In that case, the constraint means that no more than Q nodes can belong to any loop of the solution. With this constraint, we can confine the fault to the loop only in which it occurred.

$$\begin{aligned} &\text{Minimise} \sum_{i,j} d_{ij}x_{ij} \\ &\text{s.t.} \\ &\sum_{i \in L_p} W_i x_{ij} \le Q \\ &\sum_{i,j} x_{ij} = m \\ &x_{ij} = 0 \quad \text{or} \quad 1 \end{aligned} \tag{1}$$

Since MCLP is NP-hard [19], optimal solutions for a large problem cannot be obtained in a reasonable amount of computing time. Moreover, no exact algorithm for MCLP has been reported in the literature. Therefore, heuristic methods [18, 20] have been developed to obtain

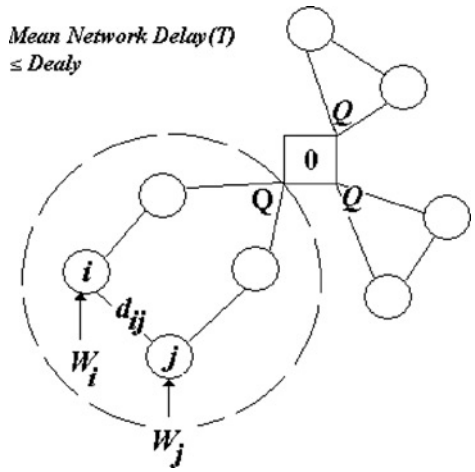

Fig. 2 *DC-MCLP*

approximate solutions for the problem. Heuristics for MCLP are to convert a given TSP tour into feasible loops, with the constraint that the maximum number of nodes to be included in a single loop is less than the predefined threshold. However, heuristic algorithms consider neither mean delay constraint nor allocate the link capacity. Thus, they supply only a collection of loops with a minimal distance.

While MCLP deals with only traffic capacity, DC-MCLP deals with both traffic capacity and network mean delay constraints simultaneously. The objective of the DC-MCLP is to find a collection of least-cost loops rooted at the source node. Since the number of nodes that a port can serve is limited in a real environment, several loops have to be found. In addition, network mean delay constraint, QoS parameter of end-user nodes is added to the problem. In this case, we have to consider the capacity allocation for links included in the loops to meet the required mean delay constraint. In DC-MCLP (see Fig. 2), the computed mean network delay (T) should be less than or equal to the predefined threshold (*Delay*). In the rest of this section, we formulate the DC-MCLP.

2.2 Notation

We will call our algorithm (to be presented in Section 3) the DC-MCLP algorithm, and use the following notations to formulate the DC-MCLP problem:

n	the number of nodes except the source node
0	the index of source node
m	the number of links included in the any topology obtained in the progress of the DC-MCLP algorithm
N'	set of node indexes except $0 = \{1, 2, \ldots, n\}$
N_j	set of node indexes except $j = N' - \{j\} = \{1, 2, \ldots, j-1, j+1, \ldots, n\}$
d_{ij}	distance (cost) from node i to node j
D	distance (cost) matrix
d	unit cost of link capacity
d_k	weight of link k in topology obtained by the second phase of DC-MCLP algorithm ($k = 1, \ldots, m$)
S	subset of N_j composed of k nodes (cardinality of $S = k$)
Q	maximum traffic served by single loop
W_i	traffic amount generated at each node i ($i = 1, 2, \ldots, n$)
$f_k(j, S)$	the minimum distance (cost) for a loop that covers set S composed of k nodes from the source node to node j to satisfy the traffic capacity constraint Q (k represents the number of nodes included in the set, S)
P_m	set, $\{j\} \cup S$ which is composed of the same elements ignoring the order
$f'_k(P_m)$	the minimum cost among $f_k(j, S)$ corresponding to the set, P_m
R_m	node sets corresponding to $f'_k(P_m)$ considering the order
$loop_x$	set of loops connecting of two end-nodes in R_m to the source node ($x = 1, 2, \ldots, l$)
$f(loop_x)$	total link cost corresponding to $loop_x$
R	set of optimal loops
T	mean delay time of network (s)
T_k	mean delay time of link k (s)
v	total traffic between source-destination pairs
$1/\mu$	average packet length (bits/packet)
C_k	the capacity of link k obtained by the second phase of DC-MCLP algorithm ($k = 1, \ldots, m$) (bits/s)
λ_k	traffic flow on link k (packets/s) ($k = 1, \ldots, m$)
Delay	desired mean delay time of network (s)
F	optimal cost

2.3 Assumptions

We make the following assumptions to formulate the DC-MCLP in the next section. The traffic generated at a node cannot exceed the maximum traffic served by one loop (Max $\{W_i\} \le Q$, $1 \le i \le n$) and the total traffic exceeds the maximum traffic served by a loop ($\sum_{i=1}^{n} W_i > Q$). In addition, an arrival of packet is based on a Poisson distribution and service time of packets is exponentially distributed.

2.4 Problem formulation

We formulate the DC-MCLP in order to satisfy the mean delay constraint (total network mean delay time is less than the desired mean delay) in addition to satisfying the traffic capacity limitation imposed by the existing MCLP. Generally, the mean delay time (T) of the network is given by (2).

$$T = \frac{1}{v}\sum_{k=1}^{m} \lambda_k T_k \tag{2}$$

Since each link k can be regarded as an independent M/M/1 queue, mean delay time (T_k) at link k can be obtained from queuing theory as follows.

$$T_k = \frac{1}{\mu C_k - \lambda_k} \tag{3}$$

Substituting (3) into (2) gives the network mean delay

$$T = \frac{1}{v}\sum_{k=1}^{m}\frac{\lambda_k}{\mu C_k - \lambda_k} \tag{4}$$

The total link cost for the specific $loop_x$ ($x = 1, 2, \ldots, l$) is assumed to be a linear function as given by (5).

$$F = \sum_{x=1}^{l} f(loop_x) = \sum_{k=1}^{m} dC_k d_k \tag{5}$$

Having defined the network mean delay (4) and the objective function (5), the DC-MCLP can be formulated by (6)–(10)

$$\text{Minimise} \qquad F = \sum_{k=1}^{m} dC_k d_k \tag{6}$$

s.t.

$$\frac{\lambda_k}{\mu} \le C_k \tag{7}$$

$$T = \frac{1}{v}\sum_{k=1}^{m}\frac{\lambda_k}{\mu C_k - \lambda_k} \le Delay \tag{8}$$

$$\sum_{i \in L_p} W_i x_{ij} \le Q \tag{9}$$

$$\sum_{i,j} x_{ij} = m \tag{10}$$

$$x_{ij} = 0 \quad \text{or} \quad 1$$

The objective function of DC-MCLP is to find a collection of loops with minimal link cost (6). Constraints of DC-MCLP are: (i) average traffic flow on the link should be smaller than capacity of the link (7), (ii) mean delay time of network (T) has to be dropped within allowable mean delay time (*Delay*) (8) and, (iii) maximum traffic flow on one loop must be below Q (9). If the number of nodes to be included in a loop should be less than a specific value (Q), we change W_i to 1, $\forall i$ in (9). Equation (10) represents that m links have to be included in the solution. x_{ij} represents a link between node i and j (i, j: $i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, n$). If link (i, j) is included in any loop (L_p) of the solution, then x_{ij} is set to 1.

3 Solution of the DC-MCLP

Having formulated the DC-MCLP in the previous section, we now develop the solution for the problem. Our solution consists of two phases: feasible path generation phase and matching and link capacity allocation phase as described below.

3.1 Feasible path generation phase

To generate the feasible paths for DC-MCLP using dynamic programming, we define stage variable and state variables.

- Stage variable, k ($k = 1, 2, \ldots$), is the number of nodes required to form a path rooted at the source node to any arbitrary node j.
- State variables, j and S, are the node index to be connected and the set of node indexes included in the path in order to connect node j, respectively. Then, using the principle of optimality, the recurrence relation can be obtained as shown in (11)

$$\begin{aligned}
&\text{If } \sum_{q \in S} W_q + W_j \le Q,\\
&\quad f_k(j, S) = \min_{q \in S, q \ne j}[f_{k-1}(q, S - \{q\}) + d_{qj}]\\
&\qquad (k = 1, 2, \ldots; \quad S \subseteq N_j)\\
&\text{else}\\
&\quad f_k(j, S) = \infty
\end{aligned} \tag{11}$$

In (11), $f_k(j, S)$ represents the least cost to connect node j when the number of nodes included in S is k. $f_k(j, S)$ is set to infinity when the sum of traffics at end-user nodes ($\sum_{q \in S} W_q + W_j$) exceeds Q.

Since boundary condition (BC) represents the cost to connect from the source node to node j directly without intermediate nodes, BC is defined by (12).

$$f_0(j, -) = d_{0j} \tag{12}$$

To obtain a feasible solution, we compute $f_1(j, S)$ for all (j, S) satisfying $\sum_{q \in S} W_q + W_j \le Q$ by using f_0. Then, $f_2(j, S)$ is computed using f_1. By repeating this procedure, we reach the phase where $\sum_{q \in S} W_q + W_j > Q$, for all (j, S).

Since $f_k(j, S)$ are infinity for all (j, S) at such a phase, we set $L = k$. This means that the loop cannot be extended any further. So, paths obtained at the previous stage k (0, 1, 2, ..., $L - 1$) are feasible solutions.

3.2 Matching and link capacity allocation phase

At stage k of the path generation phase, $f_k(j, S)$ represents the cost of the path that is composed of the same elements as $j \cup S$, but the order of elements included in the set is different. Among $f_k(j, S)$, the minimum cost, $f'_k(P_m)$ is computed. That is, the cost for P_m at stage k, $f'_k(P_m)$ is as follows.

$$\begin{aligned}
&f'_0(P_m) = f_0(j, -) + d_{j0}, \quad k = 0\\
&f'_k(P_m) = \min[f_k(j, S) + d_{j0}], \ \forall (j, S)\\
&\quad \text{such that } P_m - \{j \cup S\} = \emptyset, \ k = 1, 2, \ldots, L - 1
\end{aligned} \tag{13}$$

The value of $f'_k(P_m)$ from (13) represents the cost of the loop including the connection cost to the source node, which is composed of the same node indexes of different order. Node set of the optimal policy, R_m corresponds to $f'_k(P_m)$. R_m is the set of node indexes included in the loop rooted from the source node and represents the node sequence of loop. We find the set of loops containing the source node by adding the source node index (0) to the both end-side indexes of R_m as (14).

$$\begin{aligned}
&loop_x = \{0 - R_m - 0\}, \quad \forall R_m\\
&\quad \text{such that } \cup R_m = N' \text{ and } \quad R_i \cap R_j = \emptyset \quad (i \ne j)
\end{aligned} \tag{14}$$

Then, we compute $f(loop_x)$, the corresponding cost to $loop_x$. $f(loop_x)$ represents the cost for each $loop_x$ ($x = 1, 2, \ldots, l$) after allocating link capacities optimally and computing the total cost($\sum dd_k C_k$). In order to obtain the optimal link capacity for each link k, we use fixed routing to first find the mean arrival rate (λ_k: $k = 1, 2, \ldots, m$) of links on the capacitated loop ($loop_x$: $x = 1, 2, \ldots, l$). The reason to

use fixed routing is that, in order to transfer the traffic to the backbone node, each node in the loops has to send it to the neighbor node on the loop. Using the desired network mean delay (*Delay*), we can allocate link capacities optimally [20]

$$C_k = \frac{\lambda_k}{\mu}\left[1 + \frac{1}{\nu \times Delay}\sum_{j=1}^{m}\frac{\sqrt{d\lambda_j d_j}}{\sqrt{d\lambda_k d_k}}\right] \tag{15}$$

By adding the dd_kC_k for each $loop_x$, we can find the $f(loop_x)$

$$f(loop_x) = \sum_{k=1}^{m} dd_k C_k \text{ where } k \text{ is the index of each link included in the } loop_x \tag{16}$$

The global optimal value, F, is the least value among the cost, $f(loop_x)$ corresponding to $loop_x$.

$$F = \min_{x=1,2\ldots,l}[f(loop_x)] \tag{17}$$

3.3 Property of DC-MCLP algorithm

We present the following lemmas in order to show the properties of the proposed algorithm.

Lemma 1: Feasible path generation phase ends in the finite stages.

Proof: When the path generation phase ends depends on the relationship between $\sum_{i=1}^{n} W_i$ and Q. In the worst case ($\sum_{i=1}^{n} W_i \simeq Q$), we might generate maximum feasible paths. In such a case, k is close to n. The maximum of k ($=L$) can be nearly the same as $n-1$, but not more than or equal to n. If L is equal to n, we will find a single loop like travelling salesman tour. This is against our assumptions ($\sum_{i=1}^{n} W_i > Q$). Thus, we can finish the path generation phase in at most $n-1$ stages. □

Lemma 2: The number of additions and comparisons in the feasible path generation phase are $n(n-1)2^L$ and $n(n-2)2^L$ respectively.

Proof: In each stage ($k = 1, 2, \ldots, L$), we have to add ${}_{n-1}C_k$ for n nodes, and compare ${}_{n-1}C_k$ for the previous stage and n nodes. Therefore, the number of additions $= n\sum_{k=1}^{L} k \cdot {}_{n-1}C_k = n(n-1)\sum_{k=1}^{L}\{(n-2)!/(k-1)!(n-1-k)!\} = n(n-1)\sum_{k=1}^{L}\cdot {}_{n-2}C_{k-1} = n(n-1)2^L$ and the number of comparisons $= n\sum_{k=1}^{L}(k-1)\cdot {}_{n-1}C_k =$ the number of additions $- n\sum_{k=1}^{L}\cdot {}_{n-1}C_k = n(n-1)\sum_{k=1}^{L} k\cdot {}_{n-2}C_{k-1} - n\sum_{k=1}^{L} {}_{n-1}C_k \simeq n(n-1)2^L - n(2^L-1) = n(n-2)2^L$. □

Lemma 3: The upper bound for the number of additions and comparisons in the matching and link capacity allocation phase are $n(n-1)2^L$ and $n(n-2)2^L$, respectively.

Proof: In the matching and link capacity allocation phase, we do not consider the sequence of nodes unlike in the feasible path generation phase. It is natural for this to make the amount of computations less in most cases. However, since there might exist the case that the amount of computation is same according to the traffic capacity constraint ($\sum_{i=1}^{n} W_i \le Q$) regardless of the consideration for node sequence, we can state that the number of additions and comparisons in the matching and link capacity allocation phase is same as those in the feasible path generation phase at most. Thus, the upper bound for the number of additions and comparisons in the matching and link capacity allocation phase are $n(n-1)2^L$ and $n(n-2)2^L$ respectively. □

Lemma 4: DC-MCLP algorithm produces the optimal solution.

Proof: In the feasible path generation phase, we enumerate the feasible paths by using the optimality principle of dynamic programming. So, there can be no other feasible paths except our solutions. In the matching and link capacity allocation phase, we first find partitions of which unions compose of the node set, $N' = \{1, 2, \ldots, n\}$. Next, we generate loops composed of the above partitions. These loops are found by adding the index of the source node to indexes of both end nodes included in the partition. They are candidates for optimal solutions. Then, we find all the costs of loops by allocating the optimal link capacities satisfying the mean delay constraint (*Delay*). These costs are sub-optimal solutions because they represent the optimal cost for the specific corresponding loops. Finally, since we select the least cost solution among sub-optimal solutions, it is natural for that solution to be the global optimal solution. □

3.4 Optimal DC-MCLP algorithm

From the above model, the optimal DC-MCLP algorithm is described as the following.

Algorithm 1. OPTIMAL DC-MCLP

PHASE 1: Feasible path generation

for $k = 0$ and **for all** j such that $j \notin N'$ **do**
  $f_0(j,-) \leftarrow d_{0j}$
end for
while $f_k(j, S) \neq \infty$, $\forall(j, S)$ **do**
  if $\sum_{q\in S} W_q + W_j \le Q$, **then**
    for all k such that $k = 1, 2, \ldots,$ and $S \subseteq N_j$ **do**
      $f_k(j, S) \leftarrow \text{Min}_{q\in S, q\neq j}[f_{k-1}(q, S - \{q\}) + d_{qj}]$
    end for
    else $f_k(j, S) \leftarrow \infty$
  end if
end while
$L \leftarrow k$

PHASE 2: Matching and link capacity allocation

for $k = 0$ and **for all** j such that $j \notin N'$ **do**
  $f'_0(P_j) \leftarrow f_0(j,-) + d_{j0}$
end for
for all k such that $k = 1, 2, \ldots, L-1$ **do**
  $m \leftarrow 1$
  for all j such that $j < \text{Min}\{q\}_{q\in S}$ **do**
    $P_m \leftarrow \{j\} \notin S$
    $m \leftarrow m + 1$
  end for
end for
for all k such that $k = 1, 2, \ldots, L-1$ **do**
  $m \leftarrow 1$
  for all (j, S) such that $P_m - \{j \notin S\} = \emptyset$ **do**
    $f'_k(P_m) \leftarrow \text{Min}[f_k(j, S) + d_{j0}]$
    $m \leftarrow m + 1$
  end for
end for
for all x **do**
  for all m **do**
    for all R_m such that $\cup R_m = N'$ and $R_i \cap R_j = \emptyset$ $(i \neq j)$ **do**

find $loop_x$
end for
end for
end for
for all k such that $k = 1, 2, \ldots, L-1$ **do**
for all i and j do
add W_i for each i included in $loop_x$ up to C_k.
end for
end for
for all k such that $k = 1, 2, \ldots, L-1$ **do**
compute λ_k by using the fixed routing.
end for
for all $loop_x$ such that $x = 1, 2, \ldots, l$ **do**
for all k such that $k = 1, 2, \ldots, L-1$ **do**
$sum \leftarrow 0$
for all j such that $j = 1, 2, \ldots, m$ **do**
$sum \leftarrow sum + \sqrt{(d\lambda_j d_j)}/\sqrt{(d\lambda_k d_k)}$
end for
$C_k \leftarrow \lambda_k/\mu\,[1 + 1/(v \cdot Delay) \times sum]$
end for
end for
for all $loop_x$ such that $x = 1, 2, \ldots, l$ **do**
$f(loop_x) \leftarrow 0$
for all k such that $k = 1, 2, \ldots, m$ **do**
$f(loop_x) \leftarrow f(loop_x) + dd_k C_k$
end for
end for
for all x such that $x = 1, 2, \ldots, l$ **do**
$F \leftarrow \text{Min}[f(loop_x)]$
end for
find the set of optimal loops (R) corresponding to F.
Variables:
Refer to notation in section 2.2

4 Performance evaluation

Having formulated the DC-MCLP and its solution in Sections 2 and 3, in this section, we evaluate the performance of the algorithm in terms of its computational complexity.

4.1 Numerical example

To facilitate better understanding our DC-MCLP algorithm, we provide below a simple example. We consider the problem that the maximum traffic (Q) per loop is eight, traffic requirement at each node are 1, 2, 3 and 4 (W_i: $i = 1, 2, 3, 4$). Mean network delay constraint ($Delay$) is 1 ms. Also, assume that mean packet length ($1/\mu$) is 1000 bits/packet, with d set to 1. Index of the source node is 0. Distance matrix (D) is non-symmetric as shown in Table 1.

The computing steps of the optimal DC-MCLP algorithm are shown in Table 2 for the feasible path generation phase and Table 3 for matching and link capacity allocation phase. The optimal policy in Table 2 shows the node index with the least cost in the previous stage to connect node j. Numbers in { } in Table 3 represent R_m.

In Table 3, $f(loop_x)$ represents cost for each $loop_x$ ($x = 1, 2, \ldots, 6$) after allocating link capacities and computing the total cost($\sum dd_k C_k$). d_k and C_k are the distance cost and the link capacity for link k respectively. For example, Table 4 shows the result after allocating link capacities to $loop_2$. Capacities and delay times for each link to meet the total network mean delay (1 ms) are also presented. Through the optimal link capacities, we find the optimal cost ($F = 64.31$) which is the minimum among $f(loop_x)$ ($x = 1, 2, \ldots, 6$) in Table 3. Thus, the corresponding optimal loops are (0-2-4-1-0) and (0-3-0).

Table 1: Distance matrix

	0	1	2	3	4
0	0	3	1	5	4
1	1	0	5	3	4
2	5	4	0	2	1
3	3	1	3	0	3
4	5	2	4	1	0

Table 2: Feasible path generation phase

Step		$f_k(j, S)$	Optimal policy
1	$k = 0$	$f_0(1,-) = d_{01} = 3$	0
		$f_0(2,-) = d_{02} = 1$	0
		$f_0(3,-) = d_{03} = 5$	0
		$f_0(4,-) = d_{04} = 4$	0
2	$k = 1$	$f_1(1,\{2\}) = f_0(2,-) + d_{21} = 4 + 1 = 5$	2
		$f_1(1,\{3\}) = 6$, $f_1(1,\{4\}) = 6$	3, 4
		$f_1(2,\{1\}) = 8$, $f_1(2,\{3\}) = 8$, $f_1(2,\{4\}) = 8$	1, 3, 4
		$f_1(3,\{1\}) = 7$, $f_1(3,\{2\}) = 3$, $f_1(3,\{4\}) = 5$	1, 2, 4
		$f_1(4,\{1\}) = 6$, $f_1(4,\{2\}) = 2$, $f_1(4,\{3\}) = 8$	1, 2, 3
	$k = 2$	$f_2(1,\{2,3\}) = \text{Min}[f_1(2,\{3\}) + d_{21}, f_1(3,\{2\}) + d_{31}] = \text{Min}[12, 4] = 10$	3
		$f_2(1,\{2,4\}) = 4$, $f_2(1,\{3,4\}) = 6$,	4, 3
		$f_2(2,\{1,3\}) = 10$, $f_2(2,\{1,4\}) = 10$, $f_2(2,\{3,4\}) = \infty$	3, 4, x
		$f_2(3,\{1,2\}) = 9$, $f_2(3,\{1,4\}) = 7$, $f_2(3,\{2,4\}) = \infty$	1, 4, x
		$f_2(4,\{1,2\}) = 8$, $f_2(4,\{1,3\}) = 9$, $f_2(4,\{2,3\}) = \infty$	1, 1, x
	$k = 3$	$\forall(j, S)$, since $f_k(j, S) = \infty$, we let $L = 3$.	

4.2 Computational complexity and mean execution time

We first consider the amount of computation in the stage variable (k). It is maximum when the traffic requirement at each node is one ($W_i = 1, \forall i$) and the maximum traffic per single loop is Q. First, for any stage (k), $f_k(j, S)$ must be computed for $k \times {}_nC_{n-1}$ different (j, S) pairs. Since such computation requires k additions and $k-1$ comparisons, where $k = 1$ to L, the number of additions and comparisons in the feasible path generation phase are $n(n-1)2^L$ and $n(n-2)2^L$ respectively by Lemma 2 in Section 3.3. In addition, we also represent that the upper bound for number of additions and comparisons in the feasible path generation phase are $n(n-1)2^L$ and $n(n-2)2^L$ respectively by Lemma 3 in Section 3.3. Table 5 shows that the number of nodes and maximum Q corresponding to the sum of the number of additions and comparisons for number of nodes (n) being 30 and $Q = 29$.

Table 3: Matching and link capacity allocation phase

Step		$f_k'(P_m)$	R_m
1	$k=0$	$f_0'(\{1\}) = f_0(1,-) + d_{10} = 4$	{1}
		$f_0'(\{2\}) = 6$	{2}
		$f_0'(\{3\}) = 8$	{3}
		$f_0'(\{4\}) = 9$	{4}
2	$k=1$	$f_1'(\{1,2\}) = \text{Min}[\, f_1(1,\{2\}) + d_{10},$ $f_1(2,\{1\}) + d_{20}] = 6$	{2,1}
		$f_1'(\{1,3\}) = 7$, $f_1'(\{1,4\}) = 7$,	{3,1}, {4,1}
		$f_1'(\{2,3\}) = 6$, $f_1'(\{2,4\}) = 7$, $f_1'(\{3,4\}) = 8$	{3,2} {2,4}, {4,3}
	$k=2$	$f_2'(\{1,2,3\}) = \text{Min}[f_2(1,\{2,3\}) + d_{10}, f_2(2,\{1,3\}) + d_{20}, f_2(3,\{1,2\}) + d_{30}]$ $= \text{Min}[5, 15, 12] = 5$	{2,3,1}
		$f_2'(\{1,2,4\}) = 5$	{2,4,1}
		$f_2'(\{1,3,4\}) = 7$	{4,3,1}
3		$loop_1 = \{(0\text{-}2\text{-}3\text{-}1\text{-}0), (0\text{-}4\text{-}0)\}$, $f(loop_1) = 70.24$	
		$loop_2 = \{(0\text{-}2\text{-}4\text{-}1\text{-}0), (0\text{-}3\text{-}0)\}$, $f(loop_2) = 64.31$	
		$loop_3 = \{(0\text{-}4\text{-}3\text{-}1\text{-}0), (0\text{-}2\text{-}0)\}$, $f(loop_3) = 71.69$	
		$loop_4 = \{(0\text{-}2\text{-}1\text{-}0), (0\text{-}4\text{-}3\text{-}0)\}$, $f(loop_4) = 72.21$	
		$loop_5 = \{(0\text{-}3\text{-}1\text{-}0), (0\text{-}2\text{-}4\text{-}0)\}$, $f(loop_5) = 71.96$	
		$loop_6 = \{(0\text{-}4\text{-}1\text{-}0), (0\text{-}3\text{-}2\text{-}0)\}$, $f(loop_6) = 108.44$	
4		$F = \text{Min}[70.24, 64.31, 71.69, 72.21, 71.96, 108.44] = 64.31$	
		Optimal loops: (0-2-4-1-0), (0-3-0)	

In order to evaluate the proposed algorithm, we wrote a program in C and carried out the computational experiments on an IBM PC. We assumed that the traffic requirements at end-nodes are have Poisson distribution. If X is an exponentially distributed random variable, and λ is the average traffic rate of the network, they can be related by the following expression and the following expression is obtained.

$$X = \frac{1}{\lambda}\ln\left(\frac{1}{1-U}\right) \tag{18}$$

Table 5: Relationship between maximum Q and n

Number of nodes (n)	30	40	50	60	70	80	90	100
Maximum Q	29	28	27	26	26	26	25	25

In (18), U is an uniformly distributed random variable between 0 and 1. Using (18), n Poisson random number were generated and used as traffic requirements (W_i: $i = 1, 2, \ldots, n$). 10, 20, and 30 were used as the number of nodes (n). Maximum traffic (Q) handled in a single loop for three different cases (heavy traffic—$Q = 1/2\sum_{i=1}^{n} W_i$, medium traffic—$Q = 1/3\sum_{i=1}^{n} W_i$, and light traffic—$Q = 1/4\sum_{i=1}^{n} W_i$) were used.

Fig. 3 shows the mean real execution time of our proposed algorithm for three different cases. For each case, ten problems were randomly generated and executed on an IBM PC. Our proposed algorithm shows the best efficiency when the sum of the traffics is much less than Q. That is, when the number of nodes is 10, mean execution times for light, medium, and heavy traffic cases are 0.3, 0.3, and 0.2 seconds respectively. Alternatively, when the number of nodes is 30, mean execution times for light, medium, and heavy traffic cases are 60, 134 and 200 seconds respectively. In Table 5, we observe that the theoretical number of computations for ($n = 30$, $Q = 29$) and ($n = 100$, $Q = 25$) are almost the same, but the real execution time is different. This is because $k \times {}_nC_{n-1}$ storage spaces are required in each stage k to store $f_k(j, S)$, i.e. as the number of nodes becomes large, memory access time increases sharply because the main memory cannot maintain all the results from the previous computation. Thus, the execution time is sharply increased as the number of nodes becomes larger. In addition, the reason for the less execution time in the light traffics is that since the L value in the algorithm becomes small in the light traffic case, the amount of computations for our DC-MCLP algorithm also becomes small. To summarise, the proposed DC-MCLP algorithm is affected by the traffic volume and Q, and is effective in the case when the number of nodes is less than thirty and the traffic volume is light. Since the execution time increases exponentially as the number of nodes is more than thirty, it is desirable to use the heuristic method for the network with large nodes or heavy traffic.

5 Conclusions

In this paper, we have presented the problem formulation and an exact algorithm for DC-MCLP, which has not been reported in any previous work. The proposed exact

Table 4: Link capacity allocation for $loop_2$

$loop_2$	link		link cost (d_k)	Traffic flow (λ_k: pkts/s)	Capacity (C_k: Mbps)	Delay time (T_k: ms)	dd_kC_k
	k	i-j					
0-2-4-1-0	1	0-2	1.0	7.0	6.68	0.15	6.68
	2	2-4	1.0	2.0	3.57	0.28	3.57
	3	4-1	2.0	6.0	6.18	0.16	12.38
	4	1-0	1.0	7.0	6.68	0.15	6.68
0-3-0	5	0-3	5.0	3.0	4.37	0.23	21.87
	6	3-0	3.0	3.0	4.37	0.23	13.12

Mean network delay (*Delay*): 1 ms
$f(loop_2) = \sum dd_kC_k = 64.31$

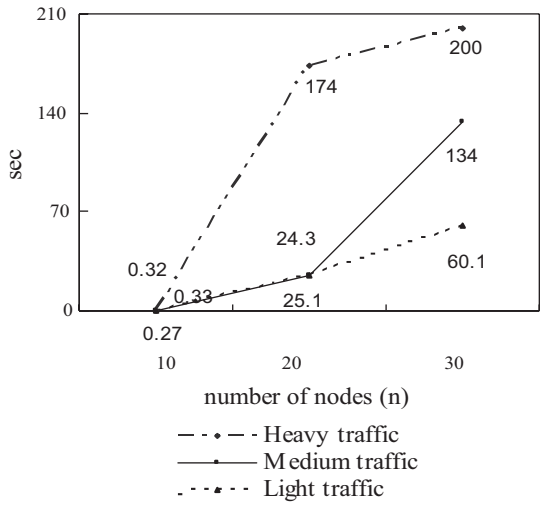

Fig. 3 *Mean execution time*

algorithm minimises the total cost to discover the optimal loop topology with additional mean network delay constraint. It consists of generating the feasible paths using dynamic programming, and finding the exact loop topology by matching and allocating the optimal link capacities to meet the network mean delay constraint. Several measures about the performance of our algorithm were derived, and through experiments, it was shown that the proposed algorithm is effective when the number of nodes is less than thirty and the total traffic volume is much smaller than the maximum traffic to be served by a port of the source node. Our proposed algorithm can be used by network designers for topological design, and applied to any network regardless of its configuration. In addition, it can be used to discover the broadcast loops for real-time multimedia traffic. Future work consists of developing a heuristic algorithm applicable to local networks with large number of nodes, in addition to an alternative exact algorithm for the small networks with further reduced computation time requirements.

6 References

1 Astic, I., and Festor, O.: 'A hierarchical topology discovery service for IPv6 networks'. IEEE/IFIP Network Operations and Management Symp., 2002, pp. 497–510

2 Breibart, Y., Garofalakis, M., Martin, C., Seshadri, S., and Silberschatz, A.: 'Topology discovery in heterogeneous IP networks'. INFOCOM, 2000, pp. 265–274

3 Lin, H., Lai, H., and Lai, S.: 'Automatic link layer topology discovery of IP networks'. ICC'99: 1999 IEEE Int. Conf. on Communications, 1999, pp. 1034–1038

4 Lin, H., Wang, Y., Wang, C., and Chen, C.: 'Web-based distributed topology discovery of IP networks'. 15th Int. Conf. on Information Networking, 2001, pp. 857–862

5 Lin, H., Lai, S., and Chen, P.: 'An algorithm for automatic topology discovery of IP networks'. IEEE Int. Conf. on Communication, 1998, pp. 1192–1196

6 Kershenbaum, A.: 'Telecommunication network design algorithm', McGraw-Hill, 1993

7 Lee, Y., and Lee, D.: 'Two phase algorithm for the topological design of local access tree networks'. Proc. IASTED Int. Conf., Acta Press, 1996, pp. 67–70

8 Gavish, B.: 'Topological design of telecommunication networks-local access design methods', *Ann. Oper. Res.*, 1991, **33**, pp. 17–71

9 Lee, Y., and Kim, T.: 'Two phase heuristic algorithm for delay constrained minimal cost loop problem', *Int. J. Netw. Manage.*, 1996, **6**, (3), pp. 142–149

10 Lee, Y.: 'Minimal cost heuristic algorithm for delay constrained loop network', *Int. J. Comput. Syst. Sci. Eng.*, 2004, **19**, (4), pp. 209–219

11 Lenster, J., and Kan, R.: 'Complexity of vehicle routing and scheduling problems', *Networks*, 1981, **11**, pp. 221–227

12 Cheng, G., and Ansari, N.: 'On multiple additively constrained path selection', *IEE Proc., Commun.*, 2002, **149**, (5), pp. 237–241

13 Cheng, G., and Ansari, N.: 'Finding all hop(s) shortest path', *IEEE Commun. Lett.*, 2004, **8**, (2), pp. 122–124

14 Chen, S., and Nahsted, K.: 'On overview of quality of service routing for next-generation high-speed networks: problems and solution', *IEEE Netw.*, 1998, **12**, (6), pp. 64–70

15 Cheng, G., and Ansari, N.: 'Achieving 100% success ratio in finding the delay constrained least cost path'. Proc. IEEE GLOBECOM '04, Dallas, TX, USA, vol. 3, pp. 1505–1509

16 Juttner, A., Szyiatowszki, Mecs, I., and Rajko, Z.: 'Lagrange relaxation based method for the QoS routing problem'. Proc. IEEE INFOCOM, Anchorage, AK, USA, 2001, vol. 2, pp. 859–869

17 Magnanti, T., Ahuja, R., and Orlin, J.: 'Network flows—theory, algorithms, and applications' (Prentice-Hall, Englewood Cliffs, NJ, USA, 1993)

18 Altinkemer, K., and Gavish, B.: 'Heuristics for delivery problems with constant error guarantees', *Trans. Sci.*, 1990, **24**, (4), pp. 294–297

19 Christofides, N., Mingozzi, A., and Toth, P.: 'Exact algorithms for the vehicle routing problem based on spanning tree and shortest path relaxations', *Math. Program.*, 1981, **20**, pp. 255–282

20 Haimovich, M., and Kan, R.: 'Bounds and heuristics for capacitated routing problems', *Math. Oper. Res.*, 1985, **10**, (4), pp. 527–542

21 Bejerano, Y., Breitbart, M., and Rastogi, R.: 'Physical topology discovery for large multi subnet networks'. INFOCOM, San Francisco, CA, USA, 2003, vol. 1, pp. 342–352

22 Huffaker, B., Plummer, D., Moore, D., and Claffy, K.: 'Topology discovery by active probing'. Applications and the Internet (SAINT) Workshops, Nara City, Nara, Japan, 2000, pp. 90–96

23 Reeves, D., and Salama, H.: 'A distributed algorithm for delay-constrained unicast routing', *IEEE/ACM Trans. Netw.*, 2000, **8**, (2), pp. 239–250

24 Zhengying, W., Bingxin, S., and Ling, Z.: 'A delay-constrained least-cost multicast routing heuristic for dynamic multicast groups', *Electron. Commer. Res.*, 2002, **2**, pp. 323–335

25 Lee, Y., and Atiquzzamman, M.: 'Optimal delay-constrained minimum cost loop algorithm for local computer network'. Proc. 10th IEEE Symp. on Computers and Communications, IEEE Computer Society, Cantagena, Murcia, Spain, 2005, pp. 501–506