A Variable Neighborhood Descent Search Algorithm for Delay-Constrained Least-Cost Multicast Routing

Rong Qu[1], Ying Xu[1,2], and Graham Kendall[1]

[1] The Automated Scheduling, Optimisation and Planning (ASAP) Group, School of Computer Science, The University of Nottingham, Nottingham, UK
[2] School of Computer and Communication, Hunan University, Hunan, China
{rxq,yxx,gxk}@cs.nott.ac.uk

Abstract. The rapid evolution of real-time multimedia applications requires Quality of Service (QoS) based multicast routing in underlying computer networks. The constrained Steiner Tree, as the underpinning mathematical structure, is a well-known NP-complete problem. In this paper we investigate a variable neighborhood descent (VND) search, a variant of variable neighborhood search, for the delay-constrained least-cost (DCLC) multicast routing problem. The neighborhood structures designed in the VND approaches are based on the idea of path replacement in trees. They are simple, yet effective operators, enabling a flexible search over the solution space of this complex problem with multiple constraints. A large number of simulations demonstrate that our algorithm is highly efficient in solving the DCLC multicast routing problem in terms of the tree cost and execution time. To our knowledge, this is the first study of VND algorithm on the DCLC multicast routing problem. It outperforms other existing algorithms over a range of problem instances.

1 Introduction

The general problem of multicast routing has received significant research attention in the area of computer networks and algorithmic network theory [1,2,3]. It is defined as sending messages from a source to a set of destinations that belong to the same multicast group. Many real-time multimedia applications (e.g. video conferencing, distance education) require the underlying network to satisfy certain quality of service (QoS). These QoS requirements include the cost, delay, delay variation and hop count, etc, among which the delay and cost are the most important for constructing multicast trees. The end-to-end delay is the total delay along the paths from the source to each destination. The cost of the multicast tree is the sum of costs on its edges.

To search for the minimum cost tree in the multicast routing problem is the problem of finding a Steiner Tree [4], which is known to be NP-complete [5]. The Delay-Constrained Least-Cost (DCLC) multicast routing problem is the problem of finding a Delay-Constrained Steiner tree (DCST), also known to be

T. Stützle (Ed.): LION 2009 III, LNCS 5851, pp. 15–29, 2009.

NP-complete [6]. Surveys in the literature of multicast communication problems exist on both the early solutions [7] and recent optimization algorithms [8].

Algorithms for multicast routing problems can usually be classified as source-based and destination-based algorithms. Source-based algorithms assume that each node has all the necessary information to construct the multicast tree (e.g. [9,10,11]). Destination-based algorithms do not require that each node maintains the status information of the entire network, and multiple nodes participate in constructing the multicast tree (e.g. [6,12]).

The first DCST heuristic, Kompella-Pasquale-Polyzos (KPP) heuristic, uses Prim's algorithm [14] to obtain a minimum spanning tree. Another heuristic, Constrained Dijkstra (CDKS) heuristic, constructs delay-constrained shortest path tree for large networks by using Dijkstra's heuristic [15]. Bounded Shortest Multicast Algorithm (BSMA) [11], a well known deterministic multicast algorithm for the DCST problem, iteratively refines the tree to lower costs. Although developed in the mid 1990s, it is still being frequently compared with many multicast routing algorithms in the current literature. However, it requires excessive execution time for large networks as it uses the k Shortest Path algorithm [16] to find lower cost paths.

The second group of algorithms considers distributed multicast routing problems. The idea of Destination-Driven MultiCasting (DDMC) comes from Prim's minimum spanning tree algorithm and Dijkstra's shortest path algorithm. The QoS Dependent Multicast Routing (QDMR) algorithm extends the DDMC algorithm by using a weight function to dynamically adjust how far a node is from the delay bound and adds the node with the lowest weight to the current tree.

In recent years, metaheuristic algorithms such as simulated annealing [17,18], genetic algorithm [19,20], tabu search [21,22,23,24], GRASP [25] and path relinking [26] have been investigated for various multicast routing problems. In the tabu search algorithm in [24], initial solutions are generated based on Dijkstra's algorithm. A modified Prim's algorithm iteratively refines the initial solution by switching edges chosen from a backup path set. In the path relinking algorithm in [26], pairs of solutions in a reference are iteratively improved. A repair procedure is used to repair any infeasible solution. Simulation results show that this path relinking algorithm outperforms other algorithms with regards to the tree cost. However, when the network size increases and many infeasible solutions need to be repaired, it is time consuming and this is suitable for real-time small networks.

In this paper we investigate variable neighborhood descent (VND) search, a variant of variable neighborhood search (VNS), for DCLC multicast routing problems. Although VNS algorithms have been applied to Steiner tree problems (e.g. VNS as a post-optimization procedure to the prize collecting Steiner tree problem [27], and the bounded diameter minimum spanning tree problem [28]), as far as we are aware, no research has been carried out using VND on DCST problems. Experimental results show that our VND algorithms obtained the best quality solutions when compared against the algorithms discussed above.

The rest of the paper is organized as follows. In Section 2, we present the network model and the problem formulation. Section 3 presents the proposed VND algorithms. We evaluate our algorithms by computer simulations on a range of problem instances in Section 4. Finally, Section 5 concludes this paper and presents possible directions for future work.

2 The Delay-Constrained Least-Cost Multicast Routing Problem

We consider a computer network represented by a directed graph $G = (V, E)$ with $|V| = n$ nodes and $|E| = l$ edges, where V is a set of nodes and E is a set of links. Each link $e = (i, j) \in E$ is associated with two parameters, namely the link cost $C(e)$: $E \mapsto \Re^+$ and the link delay $D(e)$: $E \mapsto \Re^+$. Due to the asymmetric nature of computer networks, it is possible that $C(e) \neq C(e')$ and $D(e) \neq D(e')$, for link $e = (i, j)$ and link $e' = (j, i)$. The nodes in V include a source node s, destination nodes which receive data stream from the source, denoted by $R \subseteq V - \{s\}$, called multicast groups, and relay nodes which are intermediate hops on the paths from the source to destinations.

We define a path from node u to node v as an ordered set of links, denoted by $P(u, v) = \{(u, i), (i, j), \ldots, (k, v)\}$. A multicast tree $T(s, R)$ is a set of paths rooted from the source s and spanning all members of R. We denote by $P_T(r_i) \subseteq T$ the set of links in T that constitute the path from s to $r_i \in R$. The total delay from s to r_i, denoted by $Delay[r_i]$, is simply the sum of the delay of all links along $P_T(r_i)$, i.e.

$$Delay[r_i] = \sum_{e \in P_T(r_i)} D(e), \ \forall r_i \in R \tag{1}$$

The delay of the tree, denoted by $Delay[T]$, is the maximum delay among all the paths from the source to each destination, i.e.

$$Delay[T] = max\{Delay[r_i] \mid \forall r_i \in R\} \tag{2}$$

The total cost of the tree, denoted by $Cost(T)$, is defined as the sum of the costs of all links in the tree, i.e.

$$Cost(T) = \sum_{e \in T} C(e) \tag{3}$$

Applications may assign different upper bounds δ_i for each destination $r_i \in R$. In this paper, we assume that the upper bound for all destinations is the same, and is denoted by $\Delta = \delta_i$, $r_i \in R$.

Given these definitions, we formally define the Delay-Constrained Steiner Tree (DCST) problem as follows [6]:

> **The Delay-Constrained Steiner Tree (DCST) Problem:** Given a network G, a source node s, destination nodes set R, a link delay function $D(\cdot)$, a link cost function $C(\cdot)$, and a delay bound Δ, the objective

of the DCST Problem is to construct a multicast tree $T(s, R)$ such that the delay bound is satisfied, and the tree cost $Cost(T)$ is minimized. We can define the objective function as:

$$min\{Cost(T) \mid P_T(r_i) \subseteq T(s,R), Delay[r_i] \leq \Delta, \forall r_i \in R\} \quad (4)$$

3 The Variable Neighborhood Search Algorithms

Variable neighborhood search (VNS), jointly invented by Mladenović and Hansen [29] in 1996, is a metaheuristic for solving combinatorial and global optimization problems. Unlike many standard metaheuristics where only a single neighborhood is employed, VNS systematically changes different neighborhoods within a local search. The idea is that a local optimum defined by one neighborhood structure is not necessarily the local optimum of another neighborhood structure, thus the search can systematically traverse different search spaces which are defined by different neighborhood structures. This makes the search much more flexible within the solution space of the problem, and potentially leads to better solutions which are difficult to obtain by using single neighborhood based local search algorithms [29,30,31]. The basic principles of VNS are easy to apply, parameters being kept to a minimum. Our proposed algorithm is based on basic variable neighborhood descent search (VND), a variant of VNS algorithm [29].

3.1 Initialisation

In our VND Multicast Routing (VNDMR) algorithm, let us denote N_k, $k = 1, \ldots, k_{max}$ as the set of solutions of the k^{th} neighborhood operator upon an incumbent solution x. We first create an initial solution T_0 and then iteratively improve T_0 by employing three neighborhoods, defined in Section 3.2, until the tree cost cannot be reduced, while the delay constraint is satisfied. To investigate the effects of different initial solutions, we design two variants of the algorithm, namely VNDMR0 and VNDMR1, with the same neighborhood structures, but starting from different initial solutions:

- Initialisation by DKSLD (VNDMR0): Dijkstra's shortest path algorithm is used to construct the least delay multicast tree;
- Initialisation by DBDDSP (VNDMR1): A modified Delay-Bounded DDSP (DBDDSP) algorithm is used as the initialisation method based on the Destination-Driven Shortest Path (DDSP) algorithm, a destination-driven shortest path multicast tree algorithm with no delay constraint developed in [32].

3.2 Neighborhood Structures within the VND Algorithms

The first group of neighborhood structures within our VNDMR algorithms are designed based on an operation called path replacement, i.e. a path in a tree T_i

is replaced by another new path not in the tree T_i, resulting in a new tree T_{i+1}. Our delay-bounded path replacement operation guarantees that the tree T_{i+1} is always a delay-bounded and loop free. To present the candidate paths chosen in the path replacement, we define *superpath* (based on [11], also called *key-path* in the literature [33,34]) as the longest simple path in the tree T_i, where all internal nodes, except the two end nodes of the path, are relay nodes and each relay node connects exactly two edges. The pseudo-code of VNDMR is presented in Fig.1.

- VNDMR($G = (V, E)$, S, R, Δ, k_{max}, N_k, $k = 1, \ldots, k_{max}$)
- /*S: the source node; R: the destination nodes set; $\Delta \geq 0$: the delay bound; $k_{max} = 3$: the number of neighborhood structures; N_k: the set of neighborhoods by employing neighborhood N_k */
  - Create initial solution T_0; // by using DKSLD or DBDDSP, see Section 3.1
  - if $T_0 = NULL$ then return *FAILED*; // a feasible tree does not exist
  - else
    - $T_{best} = T_0$; $k = 1$;
    - while $k \leq k_{max}$
      - select the best neighbor T_i, $T_i \in N_k(T_{best})$;
      - if(T_i has lower cost or low delay) then $T_{best} = T_i$; $k = 1$;
      - else k++;
    - end of while loop
  - return T_{best}

Fig. 1. The Pseudo-code of the VNDMR Algorithm

The three neighborhood structures of VNDMR0 and VNDMR1 are described as below:

1. ***Neighbor1:*** the most expensive edges on each superpath in tree T_i are the candidates of the path replacement. At each step, one chosen edge is deleted, leading to two separate subtrees T_i^1 and T_i^2. The Dijkstra's shortest path algorithm is then used to find a new delay-bounded shortest path that connects the two subtrees and reduces the tree cost;
2. ***Neighbor2:*** this operator operates on all superpaths in the tree T_i (either connecting or not connecting to a destination node). At each step, one superpath is replaced by a cheaper delay-bounded path using the same path replacement strategy in ***Neighbor1***;
3. ***Neighbor3:*** all the superpaths connected to destination nodes in T_i are the candidate paths to be replaced. At each step, the deletion of a superpath divides the tree T_i into a subtree T_i' and a destination node r_i. Then the same path replacement strategy is used to search for a new delay-bounded shortest path reconnecting r_i to T_i'.

To test how different neighborhood structures will affect the performance of the VND algorithm with the same initial solution, another VND algorithm, named VNDMR2, is developed with an extended new node-based neighborhood structure. The three neighborhood structures of VNDMR2 are as follows:

1. ***Neighbor1':*** one neighborhood tree is defined by deleting a non-destination node from the current multicast tree and creating a minimum spanning tree which spans the remaining nodes by using Prim's spanning tree algorithm. Once a better tree is found, the current tree is updated. These steps are repeated until no better tree can be found for 3 times;
2. ***Neighbor2':*** the same as ***Neighbor2*** in VNDMR0 and VNDMR1;
3. ***Neighbor3':*** the same as ***Neighbor3*** in VNDMR0 and VNDMR1.

3.3 Time Complexity of the VNDMR Algorithm

Proof of the probability of transition from a spanning tree s_i to s_j (see [35]):

According to Cayley's theorem [36], for a n node network, there are n^{n-2} possible spanning trees. Thus, the number of Steiner trees is bounded by n^{n-2}. Let us consider a Markov chain of n^{n-2} states, where each state corresponds to a spanning tree. We sort these states in a decreasing order with respect to the cost of the Steiner tree. Replace each state in the sorted list with n copies of itself results into a total number of n^{n-1} states. In the Markov chain, transition edges from a state s_i go only to a right state s_j of s_i. Assume that each possible transition is equally likely. Thus the probability of a transition from s_i to s_j is:

$$p_{ij} = \frac{1}{i-1} \ (1 \leq j < i, P_{11} = 1) \tag{5}$$

We prove the time complexity of VNDMR based on the method used in [35]. Let m_i be the number of transitions needed to go from state s_i to s_1, the expected value $E[m_i] = \log(i)$. Therefore, if the VNDMR algorithm starts from the most expensive state, i.e. n^{n-1}, the expected number of transitions is $O(\log(n^{n-1})) = O(n\log(n))$. So the expected maximum number of iterations of the neighborhood structures in VNDMR is $O(n\log(n))$. The VNDMR algorithm includes three neighborhood structures (N_1, N_2, N_3), then the time complexity of VNDMR is:

$$O(nlog(n)(O(N1) + O(N2) + O(N3))) \tag{6}$$

For example, the three neighborhoods of VNDMR0 and VNDMR1 use the same path replacement strategy. A path-replacement operation is dominated by Dijkstra's shortest path algorithm which takes $O(l\log(n))$, where $l = |E|$ is the total links in the network. In the worst case, each neighborhood requires replacing at most $O(l)$ superpaths. Thus the time complexity of VNDMR0 and VNDMR1 is:

$$O(nlog(n)(3 * l * llog(n))) = O(l^2 nlog^2(n)) \tag{7}$$

4 Performance Evaluation

To evaluate the efficiency of our VNDMR algorithm, we use a multicast routing simulator (MRSIM) implemented in C++ based on Salama's generator [1]. MRSIM generates random network topologies using a graph generation algorithm described in [37]. The positions of the nodes are fixed in a rectangle of

size $4000 \times 4000km^2$. The simulator defines the link delay function $D(e)$ as the propagation delay of the link (queuing and transmission delays are negligible) and the link cost function $C(e)$ as the current total bandwidth reserved on the link in the network. The Euclidean metric is used to determine the distance $l(u, v)$ between pairs of nodes (u, v). Edges connect nodes (u, v), with a probability

$$P(u, v) = \beta exp(-l(u, v)/\alpha L) \quad \alpha, \beta \in (0, 1] \tag{8}$$

where parameters α and β can be set to obtain desired characteristics in the graph. A large β gives nodes a high average degree, and a small α gives long connections. L is the maximum distance between two nodes. In our simulations, we set $\alpha = 0.25$, $\beta = 0.40$, the average degree = 4 and the capacity of each link = 155Mb/s (in this paper we set the capacity to a large enough value so that such constraint is not considered in the problem). All simulations were run on a Windows XP computer with Pentium VI 3.4GHZ, 1G RAM.

To encourage scientific comparisons, we have put the problem details of all instances tested at http://www.cs.nott.ac.uk/∼yxx/resource.html, with some example solutions obtained by the proposed algorithms.

4.1 VNDMR with Different Initialisations

In the first set of experiments, we randomly generate 20 different network topologies for each size of 20, 50, 100, 200 and 300 nodes in the networks. For each network topology, the source node and the destination nodes are randomly selected. The delay bound in our experiments for each network topology is set as 2 times the tree delay of the DKSLD algorithm, i.e. $\Delta = 2 \times Delay(T_{DKSLD})$. For each network topology, the simulation was run 50 times, where the average tree costs and execution times were reported. We investigate the performance of two variants of VNDMR with different initializations, e.g. VNDMR0 with DKSLD and VNDMR1 with DBDDSP. Both variants employ the same neighborhood structures as defined in Section 3.2.

Fig.2 presents the tree cost and execution time of VNDMR0 and VNDMR1 for problems of different network sizes with a group size (number of destinations) of 10. We can see that the tree cost of the initial solutions obtained from DBDDSP and DKSLD can both be improved by the VNDMR algorithms. The paired t-test value of the average tree cost between VNDMR0 and VNDMR1 is 3.85, meaning VNDMR1 is significantly better than VNDMR0. We conclude that VNDMR1 performs better than VNDMR0 in terms of both tree cost and computational time.

Fig.3.(a) presents the tree costs of the two VNDMR algorithms with different initial solutions for networks of 50 nodes with different group sizes. In the table, the above observations still hold. The initial solutions from DBDDSP for VNDMR1 are better than that of DKSLD for VNDMR0. Both VNDMR algorithms can further reduce the tree cost, and VNDMR1 performs slightly better than VNDMR0. Fig.3.(b) also shows that VNDMR1 requires less execution time than that of VNDMR0.

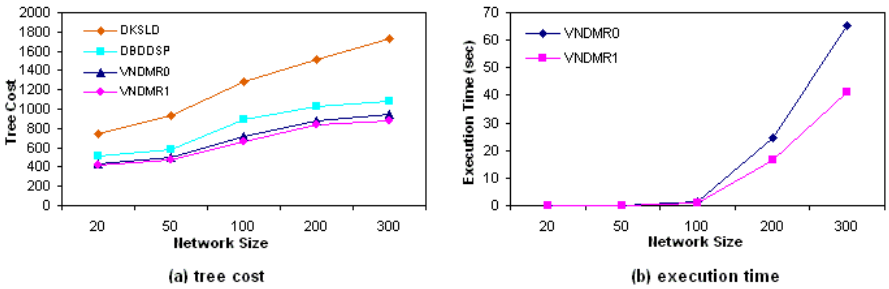

Fig. 2. Results of VNDMR with different initialisations, group size = 10

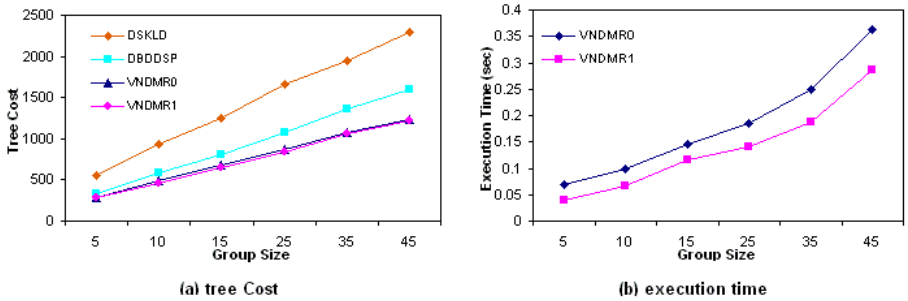

Fig. 3. Execution time by VNDMR with different initialisations, network size = 50

This group of experiments show that our VNDMR algorithms can always improve the initial solutions when constructing the DCLC multicast trees. The quality of initial solutions affects the performance of the VND algorithm. It is shown that better initial solutions from more intelligent heuristics lead to better final results, and also reduce the execution time of the VND algorithm.

4.2 VNDMR with Different Neighborhood Structures

In the second group of experiments, we test VNDMR1 and VNDMR2 on the same randomly generated network topologies in the same manner as mentioned in Section 4.1. Both VNDMR1 and VNDMR2 start from the same initial solution (DBDDSP), whereas they apply the different neighborhood structures described in Section 3.1 and 3.2, respectively.

The average tree cost and execution time of VNDMR1 and VNDMR2 on 5 different network sizes with group sizes equal to 10 are shown in Table 1. VNDMR2 always gets better average tree cost than VNDMR1 on these different network sizes. The paired t-test value of the average tree cost between VNDMR1 and VNDMR2 is 4.84, indicating a significantly difference between them. It is also observed that VNDMR2 spends longer computing time than VNDMR1.

Table 1. Average tree cost and execution time vs. network size, group size = 10

Network Size	Initial Solution DBDDSP	VNDMR1		VNDMR2	
		Cost	Time(s)	Cost	Time(s)
20	513.85	416.25	0.008	**407.9**	0.053
50	583.1	466.5	0.067	**456.85**	0.509
100	892.75	667.85	0.924	**650.2**	4.969
200	1029.15	840.55	16.474	**829.55**	29.891
300	1084.55	875.05	41.379	**851.25**	86.365

Table 2. Average tree cost and execution time vs. group size, network size = 50

Group Size	Initial Solution DBDDSP	VNDMR1		VNDMR2	
		Cost	Time(s)	Cost	Time(s)
5	330.05	280.75	0.038	**280.15**	0.075
10	583.1	466.5	0.067	**456.85**	0.214
15	809.1	682.95	0.117	**643.65**	0.373
25	1077.75	**840.25**	0.141	845.15	0.473
35	1359.95	**1055.75**	0.187	1063.45	0.583
45	1591.45	**1214.75**	0.287	1224.95	0.595

The average tree cost and execution time of VNDMR1 and VNDMR2 on the same group of 50-node networks with different group sizes are shown in Table 2. We can see that VNDMR2 gets better tree costs on the networks with small group sizes (5, 10, 15), while VNDMR1 performs better than VNDMR2 when the group size increases (25, 35, 45). It means the design of the neighborhood structures affects the performance of the VND algorithm. The *Neighbor1'* of VNDMR2 is based on an operation on the nodes in the multicast tree. With the increasing group size, i.e. the number of destination nodes, the amount of nodes which can be deleted from the current tree decreases. Since the possible neighborhood trees of the current tree that can be explored are reduced, *Neighbor1'* plays not much role when exploring the solution space. However, the edge-based VNDMR1 still performs well even on the networks with large group sizes. On the other hand, VNDMR1 spends less computing time than VNDMR2 on the tested problems.

4.3 Comparisons with Existing Algorithms

In the second set of experiments, we compare VNDMR1 with four existing multicast routing algorithms in terms of both the solution quality and the computational time on the same network topologies in Section 4.1. The four algorithms include BSMA, CDKS, QDMR, which are(DCLC multicast routing algorithms, and DKSLC, which uses Dijkstra's algorithm to construct the least cost multicast trees without the delay constraint. These algorithms have already been integrated in the MRSIM simulator and reviewed in Section 1.

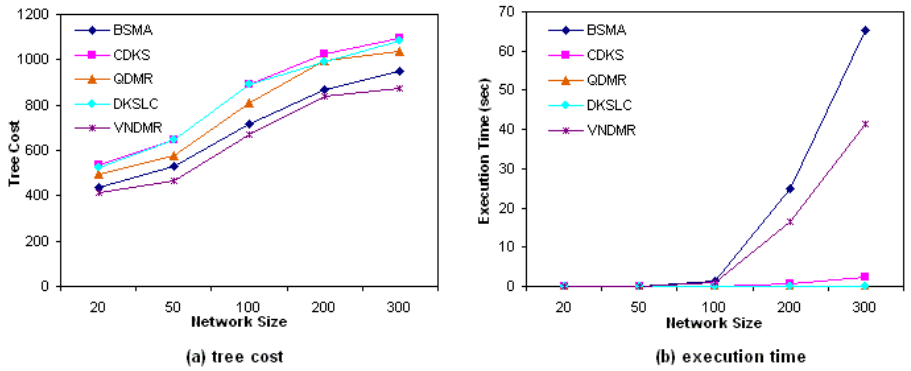

Fig. 4. Tree cost and execution time with group size = 10 from different approaches

Fig.4 presents the tree cost and execution time of these four algorithms and our VNDMR1 algorithm. It can be clearly seen in Fig.4.(a) that VNDMR1 outperforms the other four algorithms in terms of the tree cost. CDKS and DKSLD have the worst and similar tree cost; BSMA is better than QDMR but worse on the tree cost than VNDMR. In addition, Fig.4.(b) shows that VNDMR1 requires less execution time than BSMA. The other three algorithm CDKS, QDMR and DKSLC require lower computational time. However, the solution quality is of much lower quality than both BSMA and VNDMR1.

Fig.5 presents the results of our VNDMR1 algorithm and other algorithms in terms of the tree cost and execution time for problems of different network sizes, where the group size is 10% of the overall network size. Again, it can be seen in Fig.5.(a) that VNDMR1 outperforms the other four algorithms upon the solution quality. Fig.5.(b) shows that VNDMR1 requires less execution time than BSMA. This is due to that the time complexity of VNDMR1 is $O(l^2nlog^2(n))$, while BSMA's time complexity is $O(kn^3log(n))$ (n: the number of nodes, l: the number of edges, k: the k^{th} shortest path between source and a destination).

In [26], Ghaboosi and Haghighat develop a path relinking algorithm and show that it outperforms a number of existing algorithms including KPP, BSMA, GA-based algorithms [19,20], tabu search based algorithms [23,24,22,21] and another path relinking algorithm [38]. In order to compare our VNDMR algorithms with these algorithms in the literature, we generate a group of random graphs with different network sizes (10, 20, 30, 40, 50, 60, 70, 80, 90, 100 nodes). For a fair comparison, three random topologies are generated for each network size, which are the same as the simulations designed in [26]. In these graphs, the link cost depends on the link length, all the link delays are set to 1, the group size is set to 30% of the network size, the delay bounds are set to different values depending on the network sizes ($\Delta = 7$ for network size 10-30, $\Delta = 8$ for network size 40-60, $\Delta = 10$ for network size 70-80 and $\Delta = 12$ for network size 90-100).

We test two variants of VND, VNDMR1 and VNDMR2, with the same initial solution DBDDSP but different neighborhood structures as described in Section 3.2. The simulation results are reported in Tables 3 and 4.

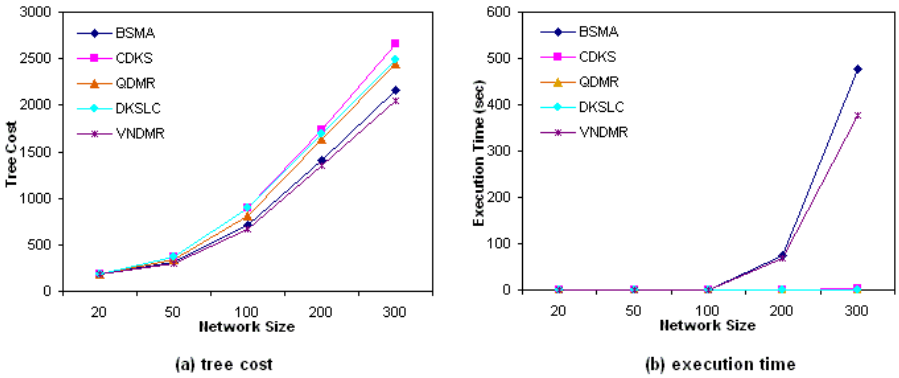

Fig. 5. Results of different approaches, group size = 10% of network size

Table 3. Average tree costs of existing algorithms on random graphs

Algorithms		Average Tree Costs
Heuristics	KPP1 [9]	905.581
	KPP2 [9]	911.684
	BSMA [11]	872.681
GA-based Algorithms	Wang et al. [19]	815.969
	Haghighat et al. [20]	808.406
TS-based Algorithms	Skorin-Kapov and Kos [22]	897.578
	Youssef et al. [21]	854.839
	Wang et al. [23]	1214.75
	Ghaboosi and Haghighat [24]	739.095
Path relinking	Ghaboosi and Haghighat [26]	691.434
VNS Algorithms	VNDMR1	**680.067**
	VNDMR2	**658.967**

As only the average tree cost over all problem instances of different sizes are reported in [26], we report the same in Table 3. It shows that the VNDMR2 performs the best in terms of the average tree cost from 10 runs for each graph. Details of the average tree cost and the execution time of VNDMR1 and VNDMR2 on each network size are given in Table 4, showing that VNDMR2 obtains the best solutions on 8 out of 10 network sizes, while VNDMR1 gets 1 best result. We also observe that VNDMR2 spends longer computing time than VNDMR1 to get the better results. The standard deviations of both VNDMR1 and VNDMR2 for each graph are 0, due to that the order of the nodes changed in the search is fixed for comparisons, i.e. there is no random factor in VNDMR1 and VNDMR2. For the 3 graphs of each size, results vary in VNDMR1 and VNDMR2. For example, for the largest graph, VNSMR2 obtained solution of 1097, 922 and 998 (average 1005.67), compared with those of 1130, 916 and 1076 (average 1040.67) from VNSMR1.

Table 4. Average results of our VNDMR on the random graphs

Network Size	VNDMR1		VNDMR2	
	Cost	Time(s)	Cost	Time(s)
10	**94.67**	0.005	**94.67**	0.003
20	282.33	0.015	**275.33**	0.032
30	415.67	0.036	**399.67**	0.17
40	518	0.063	**514**	0.362
50	726.67	0.151	**674.67**	0.859
60	812.33	0.292	**777.67**	1.392
70	805.33	0.682	**805**	2.571
80	922.33	1.286	**905.33**	5.127
90	1182.67	3.151	**1137.67**	11.705
100	1040.67	4.292	**1005.67**	15.332

We re-implemented the path relinking algorithm in [26]. Fig.6 presents the execution time of the path relinking algorithm, VNDMR1 and VNDMR2 tested on the same computer. Our VNDMR algorithms can obtain better results in a very short time compared with that of the path relinking algorithm.

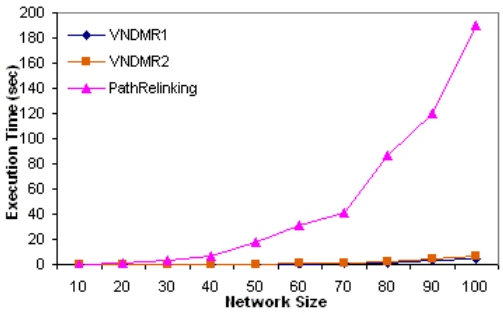

Fig. 6. Average execution time of VNDMR and the Path Relinking [26]

In summary, over a large number of simulations on instances of different characteristics, we have demonstrated that the proposed VND algorithms outperform other existing algorithms with regard to both the average tree cost and computational time. Our VNDMR2 obtains the best average tree cost on the random graphs so far.

5 Conclusions

In this paper, we have investigated variable neighborhood descent (VND) search algorithms for solving multicast network routing problems, where delay-constrained least-cost multicast trees are constructed. The problem is a Delay-Constrained Steiner tree problem and has been proved to be NP-complete. The

main characteristic of our VND algorithms is that of using three simple, yet effective, neighborhood structures. Each neighborhood is designed to reduce the tree cost in different ways and at the same time satisfy the delay constraint. This enables a much more flexible search over the search space. A large number of experimental results demonstrate that our VND algorithms are the best performing algorithms in comparison with other existing algorithms in terms of both the total tree cost and the execution time.

Many promising directions of future work are possible. Real world network scenarios are mostly dynamic with some nodes leaving and joining the multicast groups at various times. Additionally, our VND algorithm can be easily adapted for solving a variety of network routing problems with different constraints.

Acknowledgements. This research is supported by Hunan University, China, and the School of Computer Science at The University of Nottingham, UK.

References

1. Salama, H.F., Reeves, D.S., Viniotis, Y.: Evaluation of multicast routing algorithms for realtime communication on high-speed networks. IEEE Journal on Selected Areas in Communications 15, 332–345 (1997)
2. Yeo, C.K., Lee, B.S., Er, M.H.: A survey of application level multicast techniques. Computer Communications 27, 1547–1568 (2004)
3. Masip-Bruin, X., Yannuzzi, M., Domingo-Pascual, J., Fonte, A., Curado, M., Monteiro, E., Kuipers, F., Van Mieghem, P., Avallone, S., Ventre, G., Aranda-Gutierrez, P., Hollick, M., Steinmetz, R., Iannone, L., Salamatian, K.: Research challenges in QoS routing. Computer Communications 29, 563–581 (2006)
4. Hwang, F.K., Richards, D.S.: Steiner tree problems. IEEE/ACM Trans. Networking 22, 55–89 (1992)
5. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, New York (1979)
6. Guo, L., Matta, I.: QDMR: An efficient QoS dependent multicast routing algorithm. In: Proceedings of the 5th IEEE Real Time Technology and Applications Symposium, pp. 213–222 (1999)
7. Diot, C., Dabbous, W., Crowcroft, J.: Multicast communication: a survey of protocols, functions, and mechanisms. IEEE Journal on Selected Areas in Communications 15, 277–290 (1997)
8. Oliveira, C.A.S., Pardalos, P.M.: A survey of combinatorial optimization problems in multicast routing. Computers & Operations Research 32(8), 1953–1981 (2005)
9. Kompella, V.P., Pasquale, J.V., Polyzos, G.C.: Multicast routing for multimedia communication. IEEE/ACM Transactions on Networking 1, 286–292 (1993)
10. Sun, Q., Langendoerfer, H.: Efficient multicast routing for delay-sensitive applications. In: Proceedings of the 2nd Workshop on Protocols for Multimedia Systems, pp. 452–458 (1995)
11. Zhu, Q., Parsa, M., Garcia-Luna-Aceves, J.J.: A source-based algorithm for delay-constrained minimum-cost multicasting. In: Proceedings of the 14th Annual Joint Conference of the IEEE Computer and Communication (INFOCOM 1995), pp. 377–385. IEEE Computer Society Press, Boston (1995)

12. Shaikh, A., Shin, K.: Destination-driven routing for low-cost multicast. IEEE Journal on Selected Areas in Communications 15, 373–381 (1997)
13. Kou, L., Markowsky, G., Berman, L.: A fast algorithm for Steiner trees. Acta Informatica 15, 141–145 (1981)
14. Cormen, T.H., Leiserson, C.E., Revest, R.L.: Introduction to Algorithms. MIT Press, Cambridge (1997)
15. Betsekas, D., Gallager, R.: Data Networks, 2nd edn. Prentice-Hall, Englewood Cliffs (1992)
16. Eppstein, D.: Finding the k shortest paths. SIAM Journal of Computing 28, 652–673 (1998)
17. Wang, X.L., Jiang, Z.: QoS multicast routing based on simulated annealing algorithm. In: Proceedings of International Society for Optical Engineering on Network Architectures, Management, and Applications, pp. 511–516 (2004)
18. Zhang, K., Wang, H., Liu, F.Y.: Distributed multicast routing for delay variation-bounded Steiner tree using simulated annealing. Computer Communications 28, 1356–1370 (2005)
19. Wang, Z., Shi, B., Zhao, E.: Bandwidth-delay-constrained least-cost multicast routing based on heuristic genetic algorithm. Computer communications 24, 685–692 (2001)
20. Haghighat, A.T., Faez, K., Dehghan, M., Mowlaei, A., Ghahremani, Y.: GA-based heuristic algorithms for bandwidth-delay-constrained least-cost multicast routing. Computer Communications 27, 111–127 (2004)
21. Youssef, H., Sait, M., Adiche, H.: Evolutionary algorithms, simulated annealing and tabu search: a comparative study. Engineering Applications of Artificial Intelligence 14, 167–181 (2001)
22. Skorin-Kapov, N., Kos, M.: The application of steiner trees to delay constrained multicast routing: a tabu search approach. In: Proceedings of the seventh international Conference on Telecommunications, Zagreb, Croatia, pp. 443–448 (2003)
23. Wang, H., Fang, J., Wang, H., Sun, Y.M.: TSDLMRA: an efficient multicast routing algorithm based on tabu search. Journal of Network and Computer Applications 27, 77–90 (2004)
24. Ghaboosi, N., Haghighat, A.T.: A tabu search based algorithm for multicast routing with QoS constraints. In: 9th International Conference on Information Technology, pp. 18–21 (2006)
25. Skorin-Kapov, N., Kos, M.: A GRASP heuristic for the delay-constrained multicast routing problem. Telecommunication Systems 32, 55–69 (2006)
26. Ghaboosi, N., Haghighat, A.T.: A path relinking approach for Delay-Constrained Least-Cost Multicast routing problem. In: 19th International Conference on Tools with Artificial Intelligence, pp. 383–390 (2007)
27. Canuto, S.A., Resende, M.G.C., Ribeiro, C.C.: Local search with perturbations for the prize collecting Steiner tree problem in graphs. Networks 38, 50–58 (2001)
28. Gruber, M., Raidl, G.R.: Variable neighborhood search for the bounded diameter minimum spanning tree problem. In: Hansen, P., Mladenović, N., Pérez, J.A.M., Batista, B.M., Moreno-Vega, J.M. (eds.) Proceedings of the 18th Mini Euro Conference on Variable Neighborhood Search, Tenerife, Spain (2005)
29. Mladenovic, N., Hansen, P.: Variable neighborhood search. Computers & Operations Research 24, 1097–1100 (1997)
30. Jari, K., Teemu, N., Olli, B., Michel, G.: An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. Computers & Operations Research 34, 2743–2757 (2007)

31. Burke, E.K., Curtois, T.E., Post, G., Qu, R., Veltman, B.: A hybrid heuristic ordering and variable neighbourhood search for the nurse rostering problem. European Journal of Operational Research 2, 330–341 (2008)
32. Zhang, B., Mouftah, H.T.: A destination-driven shortest path tree algorithm. In: IEEE International Conference on Communications, pp. 2258–2262 (2002)
33. Martins, S.L., Resende, M.G.C., Ribeiro, C.C., Pardalos, P.M.: A parallel GRASP for the Steiner tree problem in graphs using a hybrid local search strategy. Journal of Global Optimization 17(1-4), 267–283 (2000)
34. Leitner, M., Raidl, G.R.: Lagrangian Decomposition, Metaheuristics, and Hybrid Approaches for the Design of the Last Mile in Fiber Optic Networks. In: Blesa, M.J., Blum, C., Cotta, C., Fernández, A.J., Gallardo, J.E., Roli, A., Sampels, M. (eds.) HM 2008. LNCS, vol. 5296, pp. 158–174. Springer, Heidelberg (2008)
35. Sun, Q., Langendoerfer, H.: An efficient delay-constrained multicast routing algorithm. Technical Report, Internal Report, Institute of Operating Systems and Computer Networks. TU Braunschweig, Germany (1997)
36. Cayley, A.: A theorem on trees. Journal of Math. 23, 376–378 (1989)
37. Waxman, B.M.: Routing of multipoint connections. IEEE Journal on Selected Areas in Communications 6, 1617–1622 (1988)
38. Bastos, M.P., Ribeiro, C.C.: Reactive tabu search with path relinking for the Steiner problem in graphs. In: Proceedings of the third Metaheuristics International Conference, Angra dos Reis, Brazil (1999)