A Variable 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, {rxq,yxx,gxk}@cs.nott.ac.uk
[2] School of Computer and Communication
Hunan University, Changsha, CHINA

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 variant of variable neighborhood search, variable descent search, for the delay-constrained least-cost multicast routing problems. The neighborhood structures designed in the variable descent search 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 delay-constrained, least-cost multicast routing problem in terms of both the total tree cost and the execution time. To our knowledge, this is the first study of variable neighborhood search on the delay-constrained, least-cost multicast routing problem. It outperforms other algorithms and heuristics 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 [1–3]. 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) multicast communication. These QoS requirements include the cost, delay, delay variation, lost and hop count, etc. The most important QoS requirements for constructing multicast trees are the delay and cost. The end-to-end delay is the total delay along the paths from the source to each of the destinations. The cost of the multicast tree is the sum of costs on the edges in the multicast tree.

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), which is also known to be NP-complete [6]. An early survey on multicast communication problems and related solutions was given in [7]. A recent overview on the multicast routing and associated optimization algorithms has been presented in [8].

A number of algorithms that construct low-cost multicast trees have been studied in the literature. Usually these algorithms can be classified as source-based and destination-based multicast routing algorithms. The source-based algorithms assume that each node has all the necessary information to construct the multicast tree (e.g. KPP [9], CAO [10], CKMB [11], CDKS [12] and BSMA [13]). The destination-based algorithms do not require that each node maintains the status information of the entire network , and multiple nodes are participating in constructing the multicast tree (e.g. DKPP [14], DDMC [15], DSPH [16] and QDMR [6]).

The Kompella-Pasquale-Polyzos (KPP) heuristic is the first heuristic for the DCST problem. KPP extends the Kou-Markowsky-Berman (KMB) heuristic [17], an unconstrained Steiner tree algorithm, to compute constrained paths assuming that the link delay and delay bound are integers. Both KMB and KPP use Prim's algorithm [18] to obtain a minimum spanning tree of a closure graph. The Constrained Adaptive Ordering (CAO) heuristic incrementally adds new tree branches by merging in delay-constrained low-cost unicast paths connecting to new destination nodes. It uses a constrained breadth-first search, whose computational time can be exponential in some cases. CKMB modifies KMB without assuming integer delay values. The Constrained Dijkstra (CDKS) heuristic constructs delay-constrained shortest path tree and was suggested to be a suitable algorithm for large networks by using Dijkstra's heuristic [19]. A well known deterministic multicast algorithm for the DCST problem is BSMA (Bounded Shortest Multicast Algorithm) [13]. It iteratively refines the tree to lower costs until the tree cost cannot be further reduced. BSMA was regarded as the best algorithm in terms of its good performance on tree costs. Although developed in 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 [20] 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, both using a similar greedy strategy. The Distributed Shortest Path Heuristic (DSPH) is a distributed algorithm of the delay-bounded multicast routing algorithm. It sequentially generates routing trees by extending the routing tree to one destination after another. Although DSPH performs well on tree costs, its computational time is extremely high, especially for large networks. 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 lowest weight to the current tree.

In recent years, metaheuristic algorithms such as simulated annealing [21, 22], genetic algorithm [23–26], tabu search [27–32], GRASP [33] and path relinking [34, 35] have been investigated for various multicast routing problems. Hamdan et al. [23] address the delay and delay-variation constrained multicast routing problem using genetic algorithms. In [24], a genetic local search is investigated with a logarithmic simulated annealing in a pre-processing step. An analysis of the landscape of multicast routing problems and the associated objective function are given. Youssef et al. [27] present a tabu search algorithm to construct the minimum cost delay bounded multicast tree. Their algorithm starts with an initial feasible solution, and builds a sink tree for each destination using Dijkstra's shortest path algorithm. The algorithm refines the tree to a lower cost and stops after a certain number of iterations. Wang et al. [28] propose an efficient Delay-Constrained Low-Cost Multicast Routing algorithm based on tabu search. The main disadvantage of the algorithms proposed in [27, 28] is that they randomly select the paths to be added or deleted, which may lead to a disjointed multicast tree. Yang and Wen [29] apply tabu search to the problem of pre-planning delay-constrained backup paths for multicast trees to minimize the total cost of all the backup paths. In [33], a GRASP heuristic is developed for the DCST problem. Based on a randomized feasible solution constructed by Dijkstra's shortest algorithm, a tabu search algorithm is applied to further improve the solution quality. Ghaboosi et al. [30] propose a tabu search based algorithm and a path relinking algorithm [34]. The tabu search algorithm creates initial solutions based on Dijkstra's algorithm, and iteratively refines the initial solution by using a modified Prim's algorithm to switch edges chosen from the backup path set. Their path relinking algorithm first generates a reference set of random solutions and then iteratively applies a path relinking method to improve pairs of solutions (the initial solution and guiding solution). In each iteration, the chosen solutions are converted to Prüfer numbers. During each path relinking phase, a repair procedure is used to repair any infeasible solution. If the relinking process produces a better solution than the worst solution in the reference set, the worst solution is replaced by the better one. After a given number of iterations, the best solution in the reference set is output as the final solution. In [34], the simulation results show that their path relinking algorithm is the best performing with regards to the tree cost compared with other algorithms. However, their path relinking algorithm is time consuming when the network size increases and many infeasible need to be repaired. So their path relinking algorithm is suitable for real-time small networks.

In this paper we investigate variable descent search (VDS), 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 [36], and the bounded diameter minimum spanning tree problem [37]), as far as we are aware, no research has been carried out using VDS on DCST problems. Experimental results show that our VDS 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 VDS 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 edges, respectively. Each edge $e = (i, j) \in E$ is associated with two parameters, namely the link cost $C(e)$: $E \mapsto \Re^+$ and the link delay $D(e)$: $E \mapsto \Re^+$, where $\Re^+$ is the set of nonnegative real numbers. The link is bidirectional, i.e. the existence of a link $e = (i, j)$ from node i to node j implies the existence of another link $e' = (j, i)$ for any $i, j \in V$. Due to the asymmetric nature of computer networks, it is possible that $C(e) \neq C(e')$ and $D(e) \neq D(e')$. 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 ordering 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 Delay-Constrained Steiner Tree (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:

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

3 The Variable Descent Search Algorithms

Variable neighborhood search (VNS), jointly invented by Mladenović and Hansen [38] 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 the employment of 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 [39, 40]. The basic principles of VNS are easy to apply, parameters being kept to a minimum.

Let us denote by N_k, $k = 1, \ldots, k_{max}$, a finite set of pre-designed neighborhood structures, and by $N_k(x)$ the set of solutions by employing the k^{th} neighborhood operator upon an incumbent solution x. Our proposed algorithm is based on the basic variable descent search (VDS), a variant of VNS algorithm presented in Fig. 1.

- *Initialisation*: Define the set of neighborhood structures N_k, $k = 1, \ldots, k_{max}$; Find an initial solution x;
- *Repeat*
  1. Set $k = 1$;
  2. Repeat the following steps until $k = k_{max}$
     (a) *Local search*: Find a local optimum x' by applying a simple steepest descent to x, $x' \in N_k(x)$;
     (b) *Move or not*: If x' is better than x then $x \leftarrow x'$, $k = 1$; otherwise, $k = k+1$;
- Until no improvement is obtained

Fig. 1. Pseudo-code of the basic Variable Descent Search [38]

3.1 Initialisation

In our VNS Multicast Routing (VNSMR) algorithm, firstly we 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 within the VNS algorithm, we design two variants of the algorithm (namely VNSMR0 and VNSMR1) with the same neighborhood structures, but starting from different initial solutions produced by using the following two heuristics:

- Initialisation by DKSLD (VNSMR0): Dijkstra's shortest path algorithm is used to construct the least delay multicast tree;
- Initialisation by DBDDSP (VNSMR1): we extend the Destination-Driven Shortest Path (DDSP) algorithm in [41], which is a destination-driven shortest path multicast tree algorithm with no delay constraint. This modified DBDDSP (Delay-Bounded DDSP) algorithm is used as the initialisation method in VNSMR1.

3.2 Neighborhood Structures within the VNS Algorithms

The neighborhood structures within our VNSMR algorithms are designed based on an operation called path replacement, where 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}. In our algorithm, the delay-bounded path replacement operation guarantees that the tree T_{i+1} is always a delay-bounded and loop free tree. To present the candidate paths chosen in the path replacement, we define *superpath* (based on [13], also called *key-path* in the literature [42, 43]) as the longest simple path in the tree T_i, in which all internal nodes, except the two end nodes of the path, are relay nodes. Each relay node connects exactly two tree edges. The pseudo-code of VNSMR is presented in Fig.2.

The three neighborhood structures 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, which divides the tree T_i into two subtrees T_i^1 and T_i^2. Then Dijkstra's shortest path algorithm is 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, a superpath is deleted, which 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 from r_i which reconnects r_i to T_i'.

- VNSMR($G = (V, E)$, 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 neighborhoods structures; N_k: the set of neighborhoods by employing neighbor 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(((*Cost*(T_i) < *Cost*(T_{best})) & (*Delay*(T_i) $\leq$ Δ)) $\|$ ((*Cost*(T_i) = *Cost*(T_{best})) & (*Delay*(T_i) < *Delay*(T_{best})))
      - then $T_{best} = T_i$; $k = 1$;
      - else k++;
    - end of while loop
  - return T_{best}

Fig. 2. The Pseudo-code of the VNSMR Algorithm

3.3 An Illustrative Example

For the ease of understanding, we present an example network in Fig.3.(a), where $s = \{0\}$ (source node) $R = \{2, 3, 9\}$ (destination nodes), Δ= 82 (delay bound) and the values a/b on each link $e = (i, j)$ represent the *Cost*/*Delay*. The resulting tree by applying BSMA and the VNSMR algorithm are shown in Fig.3.(b) and Fig.3.(d), respectively. The initial solution constructed by DBDDSP for VNSMR is shown in Fig.3.(c).

In Fig.3.(c), path $P(0, 3) = \{(0, 7), (7, 3)\}$ is a superpath where internal node (7 here) is a relay node (see the definition in Section 3.2). By using our VNSMR, $P(0, 3)$ is replaced by path $P(9, 3) = \{(9, 1), (1, 3)\}$ (using either ***Neighbor2*** or ***Neighbor3***), which results into the tree in Fig.3.(d). The tree cost is reduced from 336 in Fig.3.(c) to 281 Fig.3.(d), while the tree delay (*Delay*(T_{VNSMR}) = 66) still satisfies the delay constraint $\Delta = 82$. Comparing the trees generated by BSMA and VNSMR in Fig.3.(b) and Fig.3.(d), we can see that the tree cost found by VNSMR (281) is lower than that of BSMA (285). This is due to the flexibility of the VNSMR search which explores the search space defined by different neighborhood structures.

3.4 Time Complexity of the VNSMR Algorithm

Proof of the probability of transition from a spanning tree s_i to s_j (see [11]):

According to Cayley's theorem [44], 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

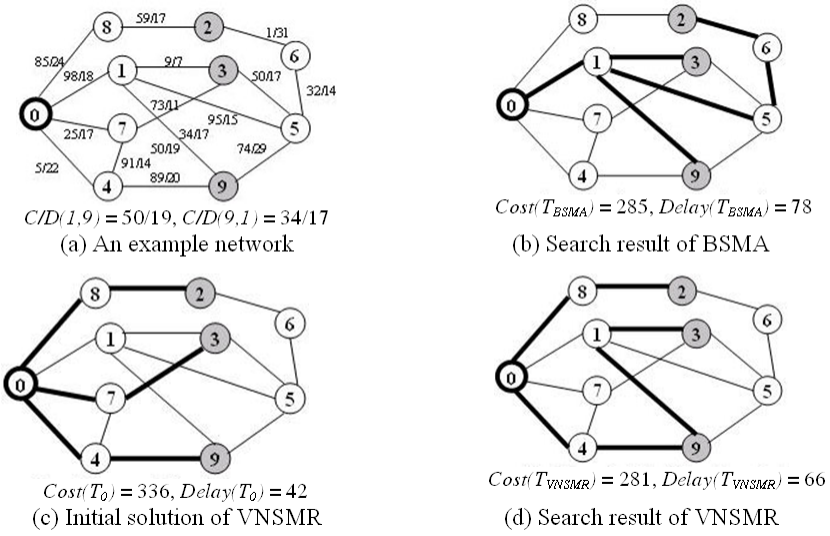


Fig. 3. An example network and solutions obtained from BSMA and VNSMR

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 of s_i. Assume that each possible transition is equally likely event. 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 VNSMR based on the method used in [11]. 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 VNSMR algorithm starts from the most expensive state, i.e. n^{n-1}, then the expected number of transitions is $O(\log(n^{n-1})) = O(n\log(n))$. Thus the expected maximum number of iterations of the neighborhood structures in VNSMR is $O(n\log(n))$. The three neighborhoods of VNSMR 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 VNSMR is:

$$O(nlog(n)(3 * l * llog(n))) = O(l^2 nlog^2(n)) \tag{6}$$

4 Performance Evaluation

To evaluate the efficiency of our VNSMR 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 [45]. 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. Like many other network simulators, the Euclidean metric is also used to determine the distance $l(u, v)$ between pairs of nodes (u, v). Edges connect nodes (u, v), with a probability

$$P(u, v) = \beta exp(-l(u, v)/\alpha L) \quad \alpha, \beta \in (0, 1] \tag{7}$$

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.

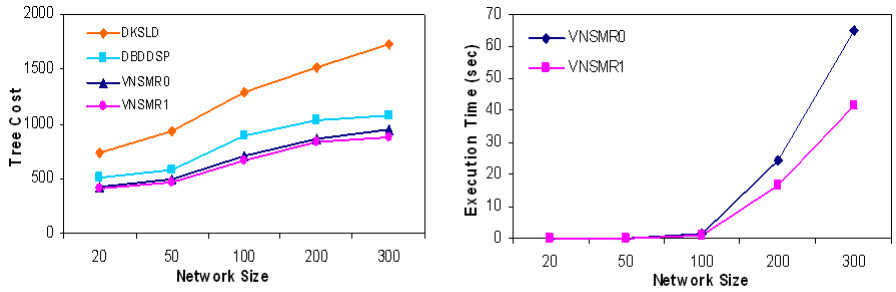
To encourage scientific comparisons, we have put problem details of all the instances tested at http://www.cs.nott.ac.uk/∼yxx/resource.html, with some example solutions obtained by the proposed algorithms.

4.1 VNSMR 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 to be 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 the VNSMR algorithm with two different initialization heuristics, e.g. VNSMR0 with DKSLD and VNSMR1 with DBDDSP (see Section 3.1 for more details). Both variants employ the same neighborhood structures as defined in Section 3.2.

Fig. 4 presents the tree cost and execution time of VNSMR0 and VNSMR1 for problems of different network sizes with a group size (number of destinations) of 10. The tree cost of the initial solutions obtained from DBDDSP and DKSLD are also presented in Fig.4.(a), where we can see that both can be improved by the VNSMR algorithm employing the three neighborhoods we have defined. VNSMR1 (with initial solution from DBDDSP) performs better than VNSMR0 (with initial solution from DKSLD) in terms of both the tree cost and the computational time.

Table 1 presents the tree costs of the two VNSMR algorithms with different initial solutions for networks of 50 nodes with different group sizes. In the table, the above observations still hold true. The initial solutions from DBDDSP for



(a) tree cost (b) execution time

Fig. 4. Tree cost and execution time vs. network size with group size = 10 by VNSMR with different initialisations

VNSMR1 are better than that of DKSLD for VNSMR0. Both VNSMR algorithms can further reduce the tree cost, and VNSMR1 always performs better than VNSMR0 (best results in bold in Table 1). Fig.5. also shows that VNSMR1 requires less execution time than that of VNSMR0.

Table 1. Tree cost vs. group size for problems of network size = 50 by VNSMR with different initialisations

Group size	DKSLLD	VNSMR0	DBDDSP	VNSMR1
5	560.5	287.3	330.05	**280.75**
10	938.6	497.9	583.1	**466.5**
15	1242.7	682.95	809.1	**652.95**
25	1666.5	867.7	1077.75	**840.25**
35	1944.45	1072	1359.95	**1055.75**
45	2294.35	1235.65	1591.45	**1214.75**

The above experiments show that our VNSMR algorithms can always improve the initial solutions when constructing the DCLC multicast trees. The quality of initial solutions affects the performance of the VDS 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 VDS algorithm.

4.2 Comparisons with Existing Algorithms

In the second set of experiments, we compare VNSMR1 with four other existing multicast routing algorithms in terms of both the solution quality and the computational time using the same simulation strategies as mentioned in Section 4.1. The four algorithms include BSMA, CDKS, QDMR, which are DCLC multicast

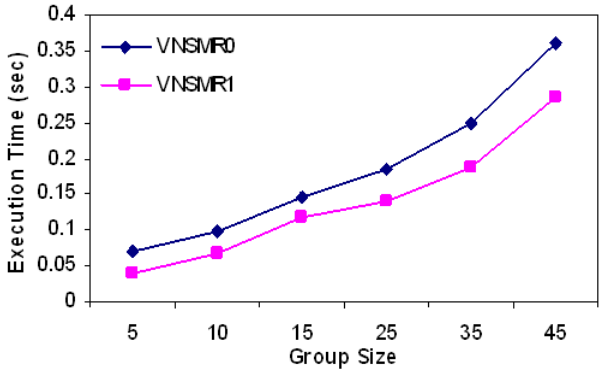


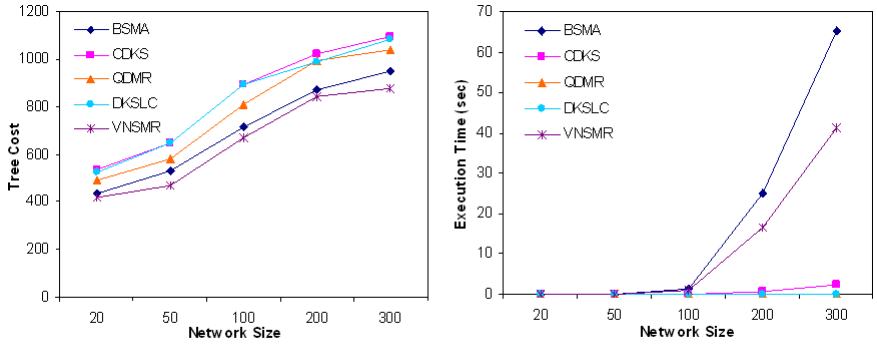
Fig. 5. Execution time vs. group size for problems of network size = 50 by VNSMR with different initialisations

routing algorithms, and DKSLC which uses Dijkstra's algorithm to construct the least cost multicast trees without the delay constraint. These algorithms have been reviewed in Section 1.

Fig.6. presents the tree cost and execution time of these four algorithms and our VNSMR1 algorithm. It can be clearly seen in Fig.6.(a) that VNSMR1 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 VNSMR. In addition, Fig.6.(b) shows that VNSMR1 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 VNSMR1.

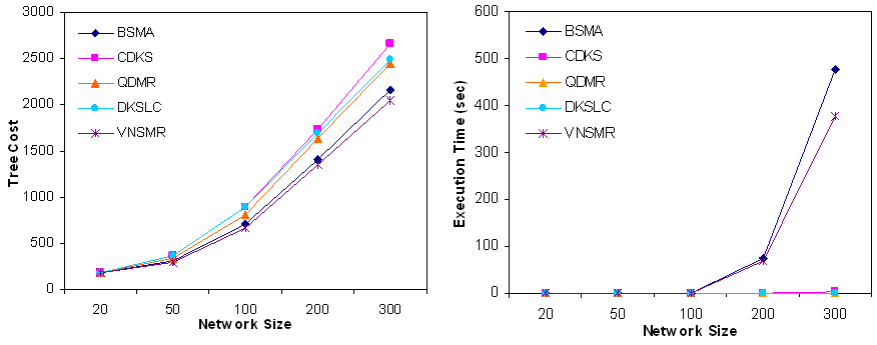
Fig.7 presents the results of our VNSMR1 algorithm and other algorithms in terms of the tree cost and execution time for problems of different network sizes, where the group size is 10% of the overall network size. Again, it can be seen in Fig.7.(a) that VNSMR1 outperforms the other four algorithms when comparing the solution quality. Fig. 7.(b) shows that VNSMR1 requires less execution time than BSMA. The time complexity of the VNSMR1 is $O(l^2 nlog^2(n))$, while BSMA's time complexity is $O(kn^3 log(n))$ (n: the number of nodes, l: the number of edges, k: the k^{th} shortest path between a source and a destination).

In [34], Ghaboosi and Haghighat develop a path relinking algorithm and show that it outperforms a number of existing algorithms, including KPP, BSMA, GA-based algorithms [25, 26], tabu search based algorithms [28, 30–32] and another path relinking algorithm [35]. In order to compare our VNSMR1 algorithm with these algorithms, we generate three random networks for each size of 10 to 100 nodes with group size = 30% of the network size. The average tree cost over all problem instances are used to compare the algorithm's performance. This is the same design as the simulations in [34].



(a) tree cost (b) execution time

Fig. 6. Tree cost and execution time vs. network size with group size = 10 from different approaches



(a) tree cost (b) execution time

Fig. 7. Tree cost and execution time vs. network size with group size = 10% of network size from different approaches

Table 2 shows that the VNSMR1 algorithm performs the best in terms of the average tree cost. As only the average tree cost over all problem instances of different sizes are reported in [34], we report the same in Table 2. The average tree costs for each size of the problems are also presented in Table 2.

Table 2. Average tree costs from our VNSMR1 and existing heuristics and algorithms on random graphs of 10-100 nodes

	Algorithms	Average Tree Costs
Heuristics	KPP1 [9]	905.581
	KPP2 [9]	911.684
	BSMA [13]	872.681
GA-based Algorithms	Wang et al. [25]	815.969
	Haghighat et al. [26]	808.406
TS-based Algorithms	Skorin-Kapov and Kos [31]	897.578
	Youssef et al. [32]	854.839
	Wang et al. [28]	1214.75
	Ghaboosi and Haghighat [30]	739.095
Path relinking	Ghaboosi and Haghighat [34]	691.434
VDS Algorithm	VNSMR1 for problem of different sizes	
	10 20 30 40 50 60 70 80 90 100	
	95 282 415 518 727 812 805 922 1183 1041	**680.067**

Furthermore, Fig.8 also shows that VNSMR1 can find better solutions in a very short time (less then 5 seconds), which is much less than that of the path relinking algorithm. This indicates that the path relinking algorithm in [34] is less practical for solving real-time network routing problems of very large and densely connected networks due to its high time complexity.

In summary, over a large number of simulations on instances of different network sizes and different group sizes, we have demonstrated that our VNSMR1 (VNS algorithm with DBDDSP as the initialization method) outperforms other existing heuristics and algorithms with regard to both the average tree cost and computational time.

5 Conclusions

In this paper, we have investigated variable descent search (VDS) 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 VDS algorithm is that of using path replacement strategy in designing 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 within the VDS algorithms

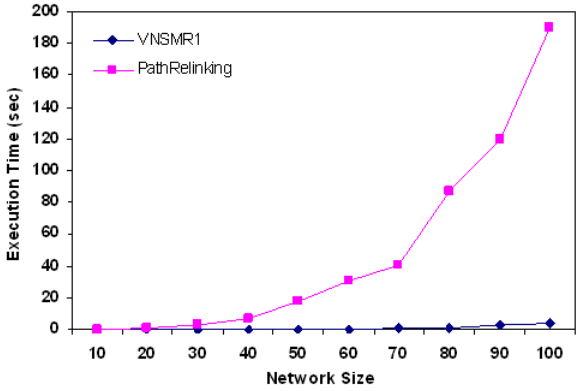


Fig. 8. Average execution time vs. network size of our VNSMR1 and the Path Relinking in [34]

when compared with a local search with single neighborhood structure. A large number of simulations have been carried out to evaluate the proposed algorithm in comparison with the existing algorithms in the literature. Experimental results demonstrate that our VDS algorithms is the best performing algorithm when compared against other heuristics and algorithms in terms of both the total tree cost and the execution time.

Many promising directions of future work are possible. The proposed algorithm is tested for the static case of multicast routing problems, where the nodes in the multicast groups are known and do not change. In reality, network scenarios are mostly dynamic with some nodes leaving and joining the multicast groups at various times. Additionally, real-time applications have other QoS requirements such as the delay variation and the packet loss rate, etc. This leads to the multi-objective optimization problem on constrained routing. The VDS algorithm can be easily adapted for solving a variety of network routing problems with different constraints.

6 Acknowledgement

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** (1997) 332–345

2. Yeo C.K., Lee B.S., Er M.H.: A survey of application level multicast techniques. Computer Communications. **27** (2004) 1547–1568
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** (2006) 563–581
4. Hwang F.K., Richards D.S.: Steiner tree problems. IEEE/ACM Trans. Networking. **22** (1992) 55–89
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 RealTime Technology and Applications Symposium, 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** (1997) 277–290
8. Oliveira C.A.S., Pardalos P.M.: A survey of combinatorial optimization problems in multicast routing. Computers & Operations Research. **32**(8) (2005) 1953–1981
9. Kompella V.P., Pasquale J.V., and Polyzos G.C.: Multicast routing for multimedia communication. IEEE/ACM Transactions on Networking. **1**, (1993) 286–292
10. Widyono R.: The design and evaluation of routing algorithms for realtime channels. Technical Report, ICSI TR-94-024, International Computer Science Institute, U.C. Berkeley (1994)
11. 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)
12. Sun Q., Langendoerfer H.: Efficient multicast routing for delay-sensitive applications. In: Proceedings of the 2nd Workshop on Protocols for Multimedia Systems, 452–458, (1995)
13. 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 95), 377–385. IEEE Computer Society Press, Boston, Massachusetts (1995)
14. Kompella V.P., Pasquale J.C., Polyzos G.C.: Two distributed algorithms for the constrained steiner tree problem. In: Proceedings of the 2nd International Conference on Computer Communications and Networking, 343–349 (1993)
15. Shaikh A., Shin K.: Destination-driven routing for low-cost multicast. IEEE Journal on Selected Areas in Communications. **15** (1997) 373–381
16. Jia X.: A distributed algorithm of delay-bounded multicast routing for multimedia applications in wide area networks. IEEE/ACM Transactions on Networking. **6** (1998) 828–837
17. Kou L., Markowsky G., Berman L.: A fast algorithm for Steiner trees. Acta Informatica, **15** (1981) 141–145
18. Cormen T.H., Leiserson C.E., Revest R.L.: Introduction to Algorithms, MIT Press. (1997)
19. Betsekas D., Gallager R.: Data Networks (2nd edition). Englewood Cliffs, HJ: Prentice-Hall, (1992)
20. Eppstein D.: Finding the k shortest paths. SIAM Journal of Computing. **28** (1998) 652–673

21. 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, 511–516 (2004)
22. Zhang K., Wang H., Liu F.Y.: Distributed multicast routing for delay variation-bounded Steiner tree using simulated annealing. Computer Communications. **28** (2005) 1356–1370
23. Hamdan M., El-Hawary M.E.: Multicast routing with delay and delay variation constraints using genetic algorithm. Canadian Conference on Electrical and Computer Engineering, 2363–2366 (2004)
24. Zahrani M.S., Loomes M.J., Malcolm J.A., Dayem Ullah A.Z.M., Steinhofel K., Albrecht A.A.: Genetic local search for multicast routing with pre-processing by logarithmic simulated annealing. Computer and Operations Research. **35** (2008) 2049–2070
25. Wang Z., Shi B., Zhao E.: Bandwidth-delay-constrained least-cost multicast routing based on heuristic genetic algorithm. Computer communications. **24** (2001) 685–692
26. 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** (2004) 111–127
27. Youssef H., Al-Mulhem A., Sait S.M., Tahir M.A.: QoS-driven multicast tree generation using tabu search. Computer Communications. **25** (2002) 1140–1149
28. 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** (2004) 77–90
29. Yang C.B., Wen U.P. Applying tabu search to backup path planning for multicast networks. Computers & Operations Research. **32** (2005) 2875–2889
30. Ghaboosi N., Haghighat A.T.: A tabu search based algorithm for multicast routing with QoS constraints. In: 9th International Conference on Information Technology, 18–21 (2006)
31. 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, 443–448, (2003)
32. Youssef H., Sait M., Adiche H.: Evolutionary algorithms, simulated annealing and tabu search: a comparative study. Engineering Applications of Artificial Intelligence. **14** (2001) 167–181
33. Skorin-Kapov N., Kos M.: A GRASP heuristic for the delay-constrained multicast routing problem. Telecommunication Systems. **32** (2006) 55–69
34. 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, 383–390 (2007)
35. 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)
36. 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** (2001) 50–58
37. Gruber M., Raidl G.R.: Variable neighborhood search for the bounded diameter minimum spanning tree problem. In: P. Hansen, N. Mladenović, J.A.M. Pérez, B.M. Batista, and J.M. Moreno-Vega (eds.), Proceedings of the 18th Mini Euro Conference on Variable Neighborhood Search, Tenerife, Spain (2005)
38. Mladenovic N., Hansen P.: Variable neighborhood search. Computers & Operations Research. **24** (1997) 1097–1100

39. 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** (2007) 2743–2757
40. 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** (2008) 330–341
41. Zhang B., Mouftah, H.T.: A destination-driven shortest path tree algorithm. In: IEEE International Conference on Communications, 2258–2262 (2002)
42. 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) (2000), 267–283
43. Leitner M., Raidl G.R.: Lagrangian Decomposition, Metaheuristics, and Hybrid Approaches for the Design of the Last Mile in Fiber Optic Networks. In: Hybrid Metaheuristics 2008, LNCS 5296, 158–174, Malaga, Spain, October 2008. Springer-Verlag Berlin Heidelberg.
44. Cayley A.: A theorem on trees. Journal of Math. **23** (1989) 376–378
45. Waxman B.M.: Routing of multipoint connections. IEEE Journal on Selected Areas in Communications. **6** (1988) 1617–1622