Appl Intell (2012) 36:229–241
DOI 10.1007/s10489-010-0256-x

A hybrid scatter search meta-heuristic for delay-constrained multicast routing problems

Ying Xu · Rong Qu

Published online: 21 October 2010

Abstract This paper investigates the first hybrid scatter search and path relinking meta-heuristic for the Delay-Constrained Least-Cost (DCLC) multicast routing problem. The underpinning mathematic model of the DCLC multicast routing problem is the constrained Steiner tree problem in graphs, a well known NP-complete problem. After combining a path relinking method as the solution combination method in scatter search, we further explore two improvement strategies: tabu search and variable neighborhood search, to intensify the search in the hybrid scatter search algorithm. A large number of simulations on some benchmark instances from the OR-library and a group of random graphs of different characteristics demonstrate that the improvement strategy greatly affects the performance of the proposed scatter search algorithm. The hybrid scatter search algorithm intensified by a variable neighborhood descent search is highly efficient in solving the DCLC multicast routing problem in comparison with other algorithms and heuristics in the literature.

Keywords Scatter search · Path relinking · Variable neighborhood search · Multicast routing

Y. Xu (✉) · R. Qu
The Automated Scheduling, Optimisation and Planning (ASAP) Group, School of Computer Science, The University of Nottingham, Nottingham NG8 1BB, UK
e-mail: cnhnxuyingying@hotmail.com

Y. Xu
School of Computer and Communication, Hunan University, Changsha, Hunan 410082, China

1 Introduction

The increasing development of numerous real-time multimedia applications (e.g. E-learning, E-commerce, video-conferencing) stimulates the demand of Quality of Service (QoS) based multicast routing in computer networks over the past decade. Multicast routing which transfers information from a source to a group of destinations simultaneously thus becomes an important communication technique. More specifically, a solution of a multicast routing problem is to construct a multicast tree which spans the source and all the destinations. Most real-time multimedia applications require the underlying computer networks to support the multicast routing which needs to concern certain QoS requirements. In reality, two most common and important QoS requirements are the cost of the multicast tree and the end-to-end delay from the source to each destination. The cost of a multicast tree is defined as the total cost of all the edges included in the tree. An end-to-end delay is the total delay of the edges along the path from the source to each destination in the multicast group (a set of all the destination nodes). Other QoS requirements include the bandwidth, delay variation, lost ratio and hop count, and so on. In this paper, we consider the Delay-Constrained Least-Cost (DCLC) multicast routing problem which concerns two of the most important QoS requirements: to minimize the total cost of the multicast tree while satisfying the end-to-end delay bound.

Multicast routing problems can be reduced to the Minimum Steiner Tree Problem in Graphs (MStTG) [1]. The MStTG problem is a well known NP-complete problem [2] that aims to search for a Steiner tree in the graph which spans a set of given nodes with the minimum total cost. The DCLC multicast routing problem can be defined as the Delay-Constrained Steiner Tree (DCST) problem, which is also known to be NP-complete [3]. The DCLC multicast routing problem with complex real world constraints thus demands effective and efficient intelligent algorithms. Due to the complexity and challenge of various QoS based multicast routing in real world applications, multicast routing problems have attracted a lot of research attention in the area of computer networks and algorithmic network theory [4–6] since the 1990s. An early survey was given in [7] to describe protocol functions, mechanisms for data transmission within a group (including multicast routing problems and end-to-end multipoint transmission controls) and related solutions. A recent survey in [8] has reviewed applications of combinatorial optimization problems and associated algorithms for multicast routing problems.

In this paper, we investigate a Scatter Search and Path Relinking (SSPR) meta-heuristic for the DCLC multicast routing problem. As far as we know, this is the first hybrid scatter search algorithm for multicast routing problems. We test our proposed SSPR algorithms on a set of small and medium sized instances (Steinb) for the benchmark Steiner tree problem in the OR-library [9] as well as a group of random graphs of different characteristics. Simulation results show that our proposed SSPR algorithm is highly efficient for solving the DCLC multicast routing problem in comparison with other existing algorithms and heuristics in the literature.

The rest of the paper is organized as follows. In Sect. 2, we present the formal definition of the DCLC multicast routing problem and summarize the related work. Section 3 presents the proposed SSPR algorithms. To evaluate the performance of our SSPR algorithms, a large amount of experimental results on a range of problem instances have been analyzed in Sect. 4. Finally, we conclude this paper and present the possible future work in Sect. 5.

2 Problem definition and related work

2.1 The network model and problem definition

A computer network is modeled as a connected, directed graph $G = (V, E)$ with $|V| = n$ nodes and $|E| = l$ edges. Each edge $e = (u, v) \in E$, where u and v are two adjacent vertices of e, is associated with two real values, namely the cost $c(e)$ and the delay $d(e)$. The edge cost $c(e)$ is a measure of the utilization of the network resources along the edge. The edge delay $d(e)$ is the delay caused by transferring messages through the edge in the network. We assume that the network is asymmetric, i.e. for edge $e = (u, v)$ and edge $e' = (v, u)$, it is possible that $c(e) \neq c(e')$ and/or $d(e) \neq d(e')$. For a multicast routing problem, there is a source node $s \in V$ and a set of destination nodes, called the multicast group, denoted by $D \subseteq V \backslash \{s\}$, each destination node $r_i \in D$ receives information from the source s simultaneously.

We define a path from node u to v as a series of edges along the path, denoted by $P(u, v) = \{(u, i), (i, j), \ldots, (k, v)\}$. A solution of a multicast routing problem is a multicast tree $T(s, D) \subseteq E$ which is rooted at source s and spans all destination nodes in D. The path in T from s to $r_i \in D$, denoted by $P(s, r_i) \subseteq T$, is a set of edges along the path. The end-to-end delay from s to each destination r_i is the sum of the delays of all edges along $P(s, r_i)$, denoted by $Delay(r_i)$, i.e.

$$Delay(r_i) = \sum_{e \in P(s, r_i)} d(e) \tag{1}$$

The delay of the tree, denoted by $Delay(T)$, is the maximum delay among all $Delay(r_i)$ from source s to each destination r_i, i.e.

$$Delay(T) = \max\{Delay(r_i) \mid \forall r_i \in D\} \tag{2}$$

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

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

The delay bound is the upper bound of the end-to-end delay for each destination, i.e. $Delay(r_i)$ along the path from s to r_i. Applications in computer communication networks may assign different upper bound δ_i to each destination $r_i \in D$. In this paper, we assume that the delay bounds for all destinations are the same, denoted by $\Delta = \delta_i, r_i \in D$.

Given the above definitions, we formally define the DCLC multicast routing problem, i.e. the Delay-Constrained Steiner Tree problem, as follows:

The Delay-Constrained Steiner Tree (DCST) problem Given a network G, a source node s, a set of destination nodes D, an edge cost function $c(.)$, an edge delay function $d(.)$, and a delay bound Δ, the objective of the Delay-Constrained Steiner Tree (DCST) problem is to construct a multicast tree $T(s, D)$ such that the delay bound is satisfied, and the tree cost $Cost(T)$ is minimized. So we can define the objective function of DCST as follows:

$$\min\{Cost(T) \mid P(s, r_i) \subseteq T(s, D),\ Delay(r_i) \leq \Delta, \\ \forall r_i \in D\} \tag{4}$$

For the ease of understanding, Fig. 1 presents a simple example of a random directed network graph with $|V| = 9$ nodes and $|E| = 14$ edges, the numbers beside each edge are the cost and delay of the directed edge, i.e. $c(e)/d(e)$, the source node $s = 5$, the multicast group $D = \{0, 2, 7\}$. An example multicast tree connected by bold arrow lines is shown in the figure. Here the delay bound Δ is assumed to be a very large number.

2.2 Related work

With the rapid development of computer networks, the DCLC multicast routing problem has received extensive research efforts in computer network community, among which lots of exact and heuristic algorithms have been investigated since the first DCLC multicast routing algorithm KPP [10] was presented in 1993. Most of the early multicast routing algorithms can be classified as source-based or destination-based algorithms. The source-based algorithms (e.g. [10–14]) assume each node has all the necessary information to construct the multicast tree. While the destination-based algorithms do not require every node in the network to maintain the information of the entire network, and multiple nodes can participate in constructing the multicast tree, examples can be found in [3, 15–17].

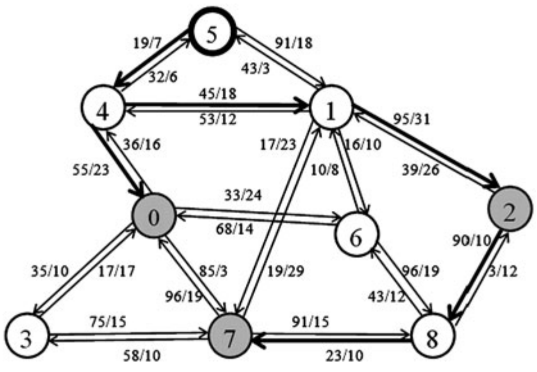


Fig. 1 An example of a random network graph and a multicast tree

Recently, many meta-heuristic algorithms such as simulated annealing [18, 19], genetic algorithm [20–23], tabu search [24–27], path relinking [28], Greedy Random Adaptive Search Procedure (GRASP) [29, 30], and Variable Neighborhood Search (VNS) [31] have been investigated for various multicast routing problems. We summarize the heuristic algorithms for multicast routing problems with different QoS requirements in the literature in Table 1, categorized by the type of heuristics and ordered in the year of publication. From the table we can see that a large amount of heuristics and algorithms exist in the literature for solving a wide range of multicast routing problems. In this paper, we only review the most relevant recent meta-heuristic approaches in this rich literature.

Ghaboosi et al. [28] have presented the first path relinking approach for the DCLC multicast routing problem. In their algorithm, a prüfer relinking is designed to implement the path relinking process, where a prüfer number encoding is used to represents a multicast tree, i.e. a solution. A multicast tree with n nodes can be encoded by a prüfer number with $n-2$ bits. After generating a reference set of random

Table 1 Summary of related multicast routing algorithms in the literature (* represents a DCLC multicast routing algorithm has been investigated in the corresponding work)

	Algorithms	Description
Heuristic Algorithms	Kompella et al. (1993) [10, 15]	KPP, the first source based multicast heuristic *
	Widyono (1994) [11]	CAO, source based heuristic *
	Zhu et al. (1995) [14]	BSMA, source based heuristic *
	Sun and Langendoerfer (1997) [13]	CDKS, source based heuristic *
	Jia (1998) [17]	DSPH, destination based heuristic *
	Guo and Matta (1999) [3]	QDMR, destination based heuristic *
Genetic	Wang et al. (2001) [20]	Bandwidth-delay-constrained least-cost multicast algorithm
Algorithms	Haghighat et al. (2004) [21]	Delay-delay variation-constrained multicast algorithm
	Zahrani et al. (2008) [22]	Capacity-delay-constrained genetic local search for group multicast
Tabu Search	Youssef et al. (2002) [24]	TS, based on Dijkstra's shortest path algorithm [31] *
Algorithms	Skorin-Kapov and Kos (2003) [25]	TS-CST, based on Prim's spanning tree algorithm [32] *
	Wang et al. (2004) [26]	TSDLMRA, based on the Kth shortest path algorithm [33] *
	Ghaboosi and Haghighat (2006) [27]	TS-based, uses the Kth shortest path and Prim's algorithm *
Path Relinking	Ghaboosi and Haghighat (2007) [28]	The prüfer number based path relinking algorithm *
GRASP	Skorin-Kapov and Kos (2006) [29]	GRASP-CST, a GRASP with tabu search algorithm *
Algorithms	Xu and Qu (2009) [30]	GRASP-VND, a hybrid GRASP and VNS algorithm *
VNS Algorithm	Qu et al. (2009) [31]	VNDMR, a variable neighborhood descent search algorithm *

initial solutions, the path relinking algorithm operates on pairs of randomly chosen elite solutions repeatedly. A repair procedure repairs infeasible solutions when they appear. At the end of each iteration, the algorithm updates the reference set by replacing the worst solution in the reference set by the better ones generated in the relinking process. The best solution in the reference set is output as the final solution after a given number of iterations. One disadvantage of the path relinking algorithm is its high time complexity since it has to spend a lot of time to repair infeasible solutions which occur during the path relinking phase.

In [29], a GRASP heuristic is developed for the delay-constrained multicast routing problem. During each iteration, a greedy randomized initial solution is constructed by using the Dijkstra's shortest path algorithm in the construction phase. A modified tabu search heuristic [25] is then applied to improve the initial solution in the local search phase. After a fixed number of iterations, the best solution found during the GRASP procedure is accepted as the final solution.

The first variable neighborhood descent search algorithm VNDMR [31] has been proposed in the authors' previous work for the DCLC multicast routing problem. In VNDMR, two neighborhood operators (node-based and path-based) have been designed to reduce the tree cost and while time satisfying the delay constraint. Experiment results demonstrate that the neighborhood structure plays a crucial role in the performance of the VNDMR algorithm and that better initial solutions lead to better final solutions and reduce the computational time. The VNDMR algorithm has shown to be highly efficient with regard to both the computational time and the tree cost. In [30], a new hybrid GRASP approach, named GRASP-VND, is also developed for the DCLC multicast routing problem, where the VNDMR algorithm [31] is applied in the local search phase. A large number of experiments carried out on some benchmark instances and a group of random graphs demonstrate that the proposed GRASP-VND outperforms another GRASP-CST in [29] and a Multi-VND (an extended multi-start algorithm of VNDMR by running it for a fixed number of iterations) along with other existing algorithms and heuristics in terms of the average tree cost.

In this paper, we investigate the first scatter search and path relinking approach, hereafter named SSPR, for the DCLC multicast routing problem. Although the scatter search meta-heuristic has been successfully applied to solve a variety of combinatorial optimization problems in the literature [35–40], to the best of our knowledge, no research has been carried out to apply it for solving the QoS multicast routing problem. As suggested in [36], a path relinking procedure is applied as the combination method in our SSPR meta-heuristic. To intensify the search towards better solutions, we explore two improvement strategies: Tabu Search (TS) and Variable Neighborhood Descent (VND) search in the proposed SSPR algorithm, namely SSPR-TS and SSPR-VND, respectively. We test these two variants of the algorithm on a set of small and medium sized (50–100 nodes) instances (Steinb) for the Steiner tree problem from the OR-library. Results indicate that SSPR-VND can obtain better solutions in comparison with SSPR-TS for the Steinb instances with different delay bounds. The proposed SSPR algorithms are also compared with two existing algorithms (GRASP-CST in [29] and GRASP-VND in our previous work [30]). SSPR-VND has similar performance as GRASP-VND and GRASP-CST for the Steinb instances with two larger delay bounds, while outperforms the two GRASP algorithms on the same instances with a tighter delay bound. Furthermore, results of our SSPR-TS and SSPR-VND algorithms on a set of random graphs (10–100 nodes) show that our proposed SSPR-VND algorithm has the best performance in terms of the total tree cost in comparison with some existing algorithms and heuristics in the literature.

3 The proposed scatter search and path relinking (SSPR) algorithm

Scatter search is a population-based meta-heuristic that has recently shown to be efficient for solving a wide range of combinatorial and nonlinear optimization problems. It operates on a set of solutions, called the reference set, which consists of good solutions obtained from the previous search. The aim of scatter search is to derive new solutions from combined solutions. Path relinking has been suggested as a solution combination approach to integrate intensification and diversification strategies in a search procedure. It operates on pairs of solutions (the initiating solution and the guiding solution) to explore the trajectory that connects each pair of solutions. Path relinking generates a new solution by recording the best solution during the path in the neighborhood space that starts from the initiating solution and move towards the guiding solution. More detailed features of scatter search and path relinking strategies can be found in [35, 36].

The scatter search meta-heuristic is a very flexible algorithm, where each component can be designed in alternative ways with regard to the problems being concerned. Figure 2 provides the pseudo-code of our proposed SSPR algorithm. Based on the basic framework of the scatter search meta-heuristic, the SSPR algorithm consists of the following five components:

a) A Diversification Generator is used to create a large set of diverse solutions *Pop*. Based on the initial population *Pop*, an initial reference set (*RefSet*) with *b* distinct solutions is built as the starting point of the procedure (where

Fig. 2 The pseudo-code of the SSPR algorithm

```
SSPR(G =(V, E), s, D, •, Psize, b)
{ // s: source node; D: destination set; • ≥ 0 is the delay bound;
  // Psize: the size of the initial population (Pop);
  // b: the size of the reference set (RefSet);
  if the Dijkstra's least delay of path $P(s, r_i) > •, \forall r_i \in D$;
  then return FAILED; // no feasible solution exists
  else
  {
      // a) Diversification Generator
      Pop = ø.
      repeat
          Create a solution x by Diversification Generator;
          If $x \notin Pop$ then add $x \in Pop$;
          else Discard x ;
      until (|Pop| = Psize);
      repeat
          Build RefSet = { $x^1, \ldots, x^b$} with b diverse solutions in Pop.
          Order the solutions in RefSet according to their objective function value in an increasing order,
          i.e. $x^1$ is the best solution with the smallest cost and $x^b$ is the worst.
          NewSolutions = True;
          while (NewSolutions)
            // b) Subset Generation Method
            Generate SubSets; // SubSets consists of all pairs of solutions in RefSet.
                              // Each pair includes at least one new solution.
            NewSolutions = False;
            while ( SubSets ≠ ø) do
                Select the next subset in SubSets; //The size of subset is 2.
                // c) Solution Combination Method
                The path relinking method is applied to generate a new solution x by combining the
                 solutions of the selected subset; //see section 3.1.
                // d) Improvement Method
                Apply the local search heuristic to improve the generated new solution x;
                // e) Reference Set Update Method
                If ( x is not in RefSet and $f(x) < f(x^b)$ ) then // f is the objective function.
                   $x^b = x$;
                   Reorder RefSet;
                   NewSolutions = True;
                end if
                Delete the subset from SubSets.
            end while
          end while
      until (the stopping criterion) //a fixed number of iterations or a given amount of execution time.
  }
}
```

b is usually a small value, e.g., no more than 20). Typically, the size of *Pop* (*Psize*) is 10 times the size of *Refset*, i.e. $Psize = 10 \times b$. The solutions in *RefSet* are ordered according to their objective function value, where the best solution is the first one in the set. In our SSPR algorithm, the Diversification Generator is a pure random generator to create initial solutions, each solution represents a multicast tree generated by starting from the source node and randomly selecting the next node which connects the tree until all the destination nodes have been mounted on the tree.

b) A Subset Generation Method operates on the reference set to generate a set of subset (*SubSets*) of *RefSet* as the basis to create combined solutions. One common *SubSets* generation method is to construct all pairs of solutions in *RefSet*, i.e. all subsets of size 2. The cardinality of *SubSets* is thus given by $(b^2 - b)/2$ corresponding to the initial *RefSet* of *b* solutions.

c) A Solution Combination Method is designed to produce new combined solutions based on the given subset of solutions from *SubSets*. The combination method is similar to the crossover operator in genetic algorithms except that it should be able to combine more than two solutions. We apply a path relinking approach (see Sect. 3.1) as the solution combination method in the proposed SSPR algorithm.

d) An Improvement Method is applied to enhance a solution by exploring neighborhoods of the current solution in order to generate new better solutions. In our SSPR algorithm, we develop and test two local search heuristics as the improvement methods (see Sect. 3.2), namely tabu search and variable neighborhood search, to further intensify the search.

e) A Reference Set Updates Method aims to maintain the reference set consisting of the *b* best solutions obtained in the previous search procedure. Different criteria may

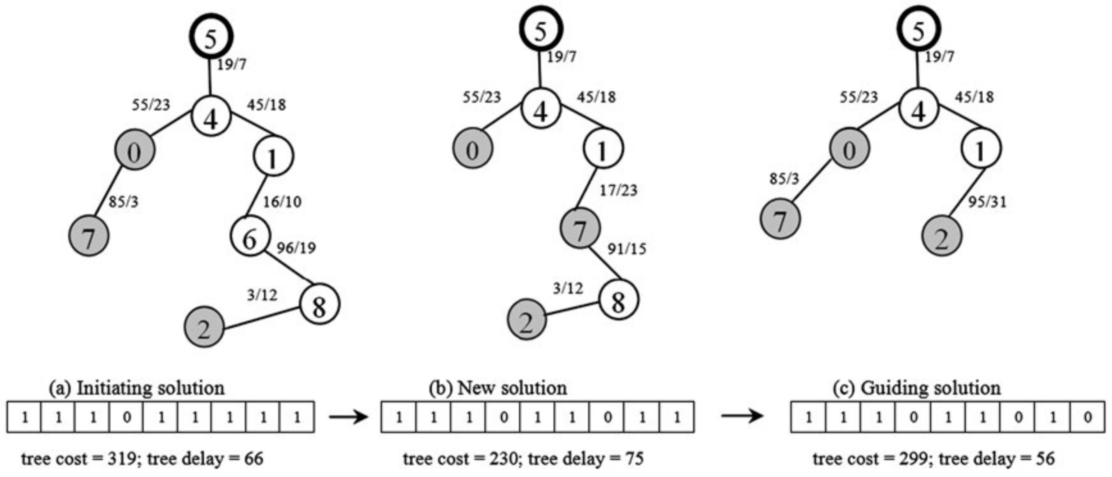


Fig. 3 An example of the path relinking process for the random network shown in Fig. 1

be defined to add solutions to *RefSet* and delete solutions from *RefSet*.

3.1 The path relinking method

Path relinking is an evolutionary approach for solving optimization problems [35, 36, 41]. It combines elements of pairs of solutions by starting from one solution, called an initiating solution, and generating a trajectory in the neighborhood space which connects to the other solution, called the guiding solution. During the path relinking, the main goal is to incorporate attributes of the pairs of solutions and record a series of moves leading from the initiating solution to the guiding solution. In our proposed SSPR algorithm, a path relinking method is applied as the combination method in the hybrid scatter search algorithm. The aim of the path relinking method is to generate a path (a series of moves) between the paired solutions (the initiating solution and the guiding solution) and therefore better solutions may occur along the path.

In our path relinking method, a solution, i.e. a multicast tree, is represented by using a binary array with $|V| = n$ bits. Each bit (from 0 to $n - 1$) represents one node in the network, and takes a value of 1 if the corresponding node is in the multicast tree, 0 otherwise. During the path relinking procedure, the algorithm will calculate the difference between the initiating solution and the guiding solution by comparing the number of different bits between their multicast tree arrays. At each step, starting from the initiating solution, the algorithm will change one different bit in the initiating solution array to the corresponding bit in the guiding solution array. Then for each of these generated array, a modified Prim's spanning tree algorithm is applied to generate a new tree of the given nodes in the array while concerning the end-to-end delay bound from the source node to each destination node. This procedure repeats until the guiding solution is finally reached, i.e. the initiating solution array becomes the same as the guiding solution array. All the solutions generated during the process are recorded, from which the best solution generated is obtained as the result of the path relinking procedure.

An illustrative example of the path relinking method in the proposed SSPR meta-heuristic is shown in Fig. 3. As described above, we use the binary array to present a solution, i.e. a multicast tree. We can see that a better new solution (tree cost = 230) is generated compared with the initiating solution (tree cost = 319) and the guiding solution (tree cost = 299) during the path relinking process. Here delay bound $\Delta = 100$.

3.2 The improvement methods

The improvement method in scatter search is an important intensification strategy to further transfer the incumbent solution into one or more enhanced solutions. In order to test the effect of improvement method in the proposed SSPR algorithm, we design two variants of SSPR algorithms, namely SSPR-TB and SSPR-VND, by integrating two improvement methods, a tabu search heuristic and a variable neighborhood descent search heuristic, respectively, within the SSPR algorithm.

The tabu search improvement method in SSPR-TS applies the modified tabu search heuristic in [25]. The tabu search heuristic uses the same solution representation as that in the path relinking method described above. The initial solution of the tabu search heuristic is the best solution generated by the path relinking procedure. Neighboring solutions include all the solutions whose binary sets are exactly one bit different from the current solution. In other words, the neighboring solutions are all those solutions generated by adding or removing exactly one node excluding the source node or the destination nodes in the current multicast tree. The Prim's spanning tree algorithm is then applied to generate a new delay-constrained spanning tree of the given nodes. The best new neighboring solution becomes the current solution in the next iteration. The process stops after a desired number of iterations without improvements; here we set the number to 2 which is the same as that in [25]. In the tabu search heuristic, the tabu list is repeatedly updated by the corresponding bit of the last performed move. The size of tabu list is set to one which is enough to prevent the algorithm from visiting the solutions of the moves just came from.

The variable neighborhood descent search algorithm VNDMR2 in SSPR-VND employs three neighborhood structures, one is a node-based neighborhood by flipping nodes in the network to generate new neighboring solutions and the other two are based on a path replacement strategy by iteratively replacing high cost paths in the tree by new better paths satisfying the delay bound to reduce the tree cost. One important parameter in the node-based neighborhood of VNDMR2 is the number of non-improved iterations. To avoid consuming too long computational time, we set the iteration number to 2 in VNDMR2 of our SSPR-VND. Detailed information can be found in our previous work [31].

4 Performance evaluation

4.1 Simulation environment

We use a multicast routing simulator (MRSIM) implemented in C++ based on Salama's generator [4], to generate random network topologies by using a graph generation algorithm described in [42]. Like many other network simulators, the distance $l(u,v)$ between pairs of nodes (u,v) is determined by the Euclidean metric. The probability of edges placed connecting nodes (u,v) is given by:

$$p(u,v) = \beta \exp(-l(u,v)/\alpha L)\alpha, \quad \beta \in (0,1] \tag{5}$$

where L is the maximum distance between two nodes, the parameters α and β can be set to obtain desired characteristics in the graph to represent features of computer networks. For example, the average node degree is increased with the value of β, and a small α gives long connections between nodes. In our simulations, we set $\alpha = 0.25$, $\beta = 0.40$, average node degree $= 4$, capacity of each edge (resource capacities in computer networks) is set to a large enough value. The edge delay $d(e)$ of sending information data via the link in the simulator is defined as the propagation delay of the link (queuing and transmission delays are negligible), the edge cost $c(e)$ is assigned as the consumption of the bandwidth reserved on the edge in the network. All simulations were run on a Windows XP computer with P4-3.4 GHz, 1 GB RAM. More detailed information of all the problem instances tested and some example solutions obtained by the algorithms are publicly available at http://www.cs.nott.ac.uk/~rxq/MRPresource.html.

Table 2 Characteristics of the Steiner tree instances (Steinb) from the OR-library. $|V|$, $|E|$ and $|D|$ stand for the number of nodes, edges and destinations. 'OPT' denotes the optimal solution for each instance given by the OR-library

No.	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert D\rvert$	OPT	No.	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert D\rvert$	OPT
B01	50	63	9	82	B10	75	150	13	86
B02	50	63	13	83	B11	75	150	19	88
B03	50	63	25	138	B12	75	150	38	174
B04	50	100	9	59	B13	100	125	17	165
B05	50	100	13	61	B14	100	125	25	235
B06	50	100	25	122	B15	100	125	50	318
B07	75	94	13	111	B16	100	200	17	127
B08	75	94	19	104	B17	100	200	25	131
B09	75	94	38	220	B18	100	200	50	218

4.2 Experiments on Steinb problems in the OR-library

4.2.1 The test instances

We test our proposed algorithms on the set of small and medium sized Steiner tree problems (Steinb) from SteinLib, which is a publicly available library of test instances for the MStTG problem. Table 2 presents the characteristics of the 18 instances. Since the Steinb instances only concern the costs of edges, there is no delay for each edge. We thus extend these 18 Steinb instances to generate Delay-Constrained multicast routing problems by setting the delays of every edge randomly in our experiments.

4.2.2 The influence of population sizes within the SSPR algorithm

To properly set the parameters in the SSPR algorithms, a number of tests were carried out. In the first group of experiments, we evaluate the performance of the SSPR algorithms with different population sizes ($Psize = 20$, 30, 40, 50 and 60) and reference set sizes ($b = Psize/10$) on the Steinb instances. In this group of experiments, we set the delay bound of each instance to a large enough value, denoted by $\Delta = \infty$, so that the solutions obtained are actually the solutions to the minimum Steiner tree problem without constraint as the delays of the edges have no impact on the construction of the Steiner tree. In this case, the optimal solution for each instance is already known as shown in Table 2. The maximum iteration number is set as 4 in variants of the SSPR algorithms. The termination criterion of these SSPR algorithms is either the algorithm finds the optimal solution or

Table 3 Comparison of different population size (*Psize*) within the SSPR algorithm. (Best results are in bold)

No.	$Psize = 20$		$Psize = 30$		$Psize = 40$		$Psize = 50$		$Psize = 60$	
	Mean	σ	Mean	σ	Mean	σ	Mean	σ	Mean	σ
B01	**82**	0	**82**	0	**82**	0	**82**	0	**82**	0
B02	89.2	0.67	89	0	88.9	0.22	**88.3**	0.72	88.5	1.47
B03	139.5	2.93	139.2	3.14	138.3	1.34	**138**	0	**138**	0
B04	66.2	2.8	67.4	4.27	65.3	3.4	**63.6**	2.35	64.9	2.17
B05	62.9	0.88	62.1	1.1	61.6	0.69	61.6	0.6	**61.1**	0.31
B06	132.1	5.74	127.5	2.5	127	2.28	**125.6**	1.6	125.7	1.37
B07	114.2	3.31	112.9	2.34	112.1	0.85	112	0.51	**111.8**	0.55
B08	108	2.6	106.9	2.26	106.1	1.7	**105.7**	1.09	106	1.23
B09	221.3	1.34	220.5	0.51	220.4	0.49	220.4	0.49	**220.3**	0.47
B10	98.8	5.74	97.7	3.94	**95.2**	1.54	95.5	3.55	95.7	1.41
B11	107.2	4.5	105.7	3.54	102	5.99	**101**	4.64	101.7	5.22
B12	184.9	2.15	181.6	2.78	179.6	3.32	**178.9**	2.28	179.2	2.71
B13	182.1	5.37	179.1	5.22	178.9	4.56	175.3	4.32	**174.2**	4.08
B14	258.9	8.76	248.6	5.02	245.6	3.08	**246.4**	4.72	246.7	2.01
B15	330.9	4.82	327.5	3	325.5	1.36	**324.6**	1.62	324.7	1.46
B16	148.7	3.48	148.4	3.15	147.2	3.07	**140.9**	4.8	**140.9**	4.88
B17	139.9	1.93	136.5	2.5	137.4	2.06	136.5	2.01	**136.3**	1.74
B18	222.9	1.62	222.5	1.61	222.3	0.86	**221.4**	1.23	221.7	1.17

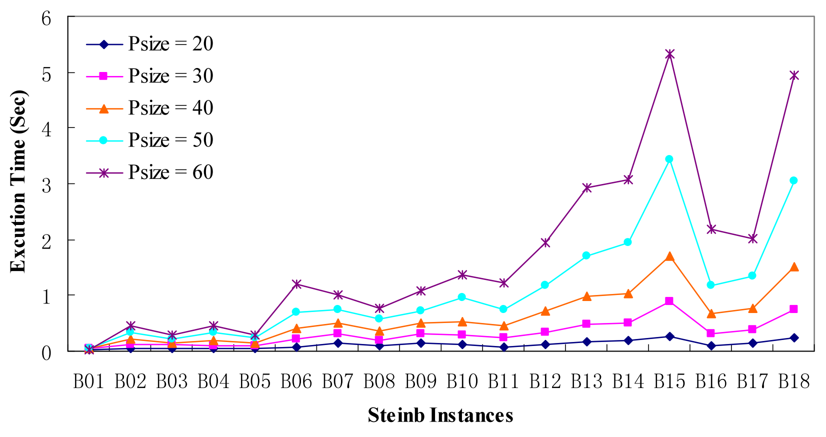


Fig. 4 The Computational time of the SSPR algorithm with different population size (*Psize*) on Steinb instances

within the fixed number of iterations. In order to clearly observe the influence of the population size, the improvement method is not applied in this set of experiments. On each instance, the simulation was run 30 times for each variant of the algorithm. Table 3 and Fig. 4 present the average tree cost, standard deviation (σ) and computational time of the SSPR algorithms on the 18 Steinb instances with different population sizes.

Results in Table 3 indicate that SSPR with $Psize = 40$, 50 and 60 outperform other variants of the algorithm. We further calculate the paired t-test value between the average tree costs on the 18 instances with $Psize = 40$ and 50. The result 2.55 is larger than the value 2.11 with $p = 0.05$, meaning their difference is statistically significant. While the t-test value between the average tree costs from SSPR with $Psize = 50$ and $Psize = 60$ is only 0.823, difference between these two variants can be seen as not significant.

From Fig. 4, we can see that with the increasing population size, the execution time of the SSPR algorithm increases. We further observe that the computing time of the SSPR algorithm with $Psize = 50$ is less than that of the SSPR algorithm with $Psize = 60$, while their performance is similar with respect to the average tree cost on each instance as shown in Table 3. We thus conclude that for our proposed SSPR algorithm, the most appropriate population size $Psize$ is 50. The population size is set as 50 in our later experiments.

Table 4 Experiment results for minimum Steiner tree problems ($\Delta = \infty$). (The values marked with '*' denote the optimal solutions and the best results are in bold)

No.	SSPR-VND			SSPR-TS			GRASP-VND			GRASP-CST		
	Avg.	Best	σ	Avg.	Best	σ	Avg.	Best	σ	Avg.	Best	σ
B01	**82***	82	0	**82***	82	0	**82***	82	0	**82***	82	0
B02	**83***	83	0	86.5	83	2.58	**83***	83	0	**83***	83	0
B03	**138***	138	0	**138***	138	0	**138***	138	0	**138***	138	0
B04	**59***	59	0	**59***	59	0	**59***	59	0	**59***	59	0
B05	**61***	61	0	**61***	61	0	**61***	61	0	**61***	61	0
B06	**122***	122	0	**122***	122	0	**122***	122	0	**122***	122	0
B07	**111***	111	0	**111***	111	0	**111***	111	0	**111***	111	0
B08	**104***	104	0	**104***	104	0	**104***	104	0	**104***	104	0
B09	**220***	220	0	**220***	220	0	**220***	220	0	**220***	220	0
B10	**86***	86	0	**86***	86	0	**86***	86	0	**86***	86	0
B11	**88***	88	0	**88***	88	0	**88***	88	0	**88***	88	0
B12	**174***	174	0	**174***	174	0	**174***	174	0	**174***	174	0
B13	168.1	165	1.92	168.5	165	2.37	167.3	165	2.39	**165.4**	165	1.09
B14	235.3	235	0.47	238.9	235	3.8	235.1	235	0.22	**235***	235	0
B15	**318***	318	0	319.2	318	1.01	319.5	318	0.89	319.8	318	0
B16	**127***	127	0	134.1	130	2	**127***	127	0	**127***	127	0
B17	**131***	131	0	**131***	131	0	131.2	131	0.67	**131***	131	0
B18	**218***	218	0	218.2	218	0.37	218.2	218	0.41	**218***	218	0

4.2.3 *Comparisons on Steinb instances with different delay bounds*

In the second group of experiments, we compare the performance of SSPR-VND and SSPR-TS with other two algorithms, GRASP-CST in [29] and GRASP-VND in our previous work [30], on the Steinb instances with different delay bounds. For a fair comparison, we set the same running time (60 seconds) for the four algorithms in each run and all algorithms were run 30 times on each instance.

Firstly, we set the delay bound $\Delta = \infty$. The average, best and standard deviation of the two variants of SSPR algorithms, along with GRASP-VND and GRASP-CST on the Steinb instances are illustrated in Table 4. From the table, we can see that SSPR-VND and GRASP-CST have similar performance, obtaining 16 and 17 best solutions out of 18 instances in terms of the average tree cost, respectively. Both algorithms are better than SSPR-TS which finds 12 best solutions and GRASP-VND which finds 13 best solutions. Similarly, both SSPR-VND and GRASP-CST always find the optimal solutions on 16 out of 18 instances, which are better than SSPR-TS and GRASP-VND which find optimal solutions on 12 and 13 out of 18 instances, respectively. In addition, the results obtained by SSPR-VND are more stable than those of SSPR-TS, since SSPR-VND has a smaller average standard deviation (0.133) over the 18 instances compared with that of SSPR-TS (0.674). The experiment results also show that SSPR-VND outperforms SSPR-TS, mainly due to the better improvement method VND within the same SSPR meta-heuristic.

In the DCLC multicast routing problem, the delay bound is a key factor which affects the difficulty of the problems, and thus leads to different search results. Generally, the smaller the delay bound, the more constrained the problems. In the second set of experiments, we set the delay bound $\Delta_1 = 1.1 \times \mathit{Delay}(T_{OPT})$, where T_{OPT} denotes the multicast tree of the optimal solution. With the slightly tighter bounded end-to-end delay compared with the delay bound $\Delta = \infty$ in the previous experiments, the SSPR-VND algorithm still outperforms the SSPR-TS algorithm in terms of both average tree costs and the standard deviation as shown in Table 5. SSPR-VND has similar overall performance as GRASP-VND and GRASP-CST, since they all find 15 out of 18 best results in terms of average tree cost.

We set the delay bound Δ_2 to a smaller value $0.9 \times \mathit{Delay}(T_{OPT})$. The optimal solutions are thus not known to any of the cases. Due to the tighter delay constraint, we can see that no feasible solutions were obtained for some instances as presented in Table 6. The table again shows that SSPR-VND outperforms SSPR-TS with respect to the average tree costs on 11 instances. We also observe that SSPR-VND is more stable than SSPR-TS comparing the average standard deviation on the instances. For this set of experiments, SSPR-VND performs better than GRASP-VND and GRASP-CST when comparing the number of best average tree costs found by each algorithm which are 14, 10 and 9 out of 18 instances, respectively. It demonstrates that the SSPR-VND is more flexible in solving problems of different delay bounds.

Table 5 Experiment results for the DCLC multicast routing algorithm with $\Delta_1 = 1.1 \times Delay(T_{OPT})$. (The values marked with '*' denote the optimal solutions and the best results are in bold)

No.	Δ	SSPR-VND			SSPR-TS			GRASP-VND			GRASP-CST		
		Avg.	Best	σ	Avg.	Best	σ	Avg.	Best	σ	Avg.	Best	σ
B01	145	**82***	82	0	**82***	82	0	**82***	82	0	**82***	82	0
B02	228	**83***	83	0	86.8	83	2.1	**83***	83	0	**83***	83	0
B03	248	**138***	138	0	**138***	138	0	**138***	138	0	**138***	138	0
B04	173	**59***	59	0	**59***	59	0	**59***	59	0	**59***	59	0
B05	125	**61***	61	0	**61***	61	0	**61***	61	0	**61***	61	0
B06	281	**122***	122	0	122.1	122	0.45	**122***	122	0	**122***	122	0
B07	212	**111***	111	0	**111***	111	0	**111***	111	0	**111***	111	0
B08	209	**104***	104	0	**104***	104	0	**104***	104	0	**104***	104	0
B09	280	**220***	220	0	**220***	220	0	**220***	220	0	**220***	220	0
B10	262	**86***	86	0	**86***	86	0	**86***	86	0	**86***	86	0
B11	235	**88***	88	0	**88***	88	0	**88***	88	0	**88***	88	0
B12	225	**174***	174	0	**174***	174	0	**174***	174	0	**174***	174	0
B13	190	168.1	165	1.67	169.2	165	1.82	167.6	165	2.39	**165***	165	0
B14	221	236.6	235	4.61	258.8	244	11.39	**235***	235	0	**235***	235	0
B15	308	**318.6**	318	0.92	319.5	318	1.39	319.6	318	0.82	319.6	318	0.82
B16	291	**127***	127	0	133.5	129	1.88	**127***	127	0	**127***	127	0
B17	219	131.7	131	1.24	**131***	131	0	131.9	131	0.88	131.5	131	0.51
B18	425	**218***	218	0	218.2	218	0.41	**218***	218	0	218.2	218	0.37

Table 6 Experiment results for the DCLC multicast routing algorithm with $\Delta_2 = 0.9 \times Delay(T_{OPT})$. (The best results are in bold)

No.	Δ	SSPR-VND			SSPR-TS			GRASP-VND			GRASP-CST		
		Mean	Best	σ	Mean	Best	σ	Mean	Best	σ	Mean	Best	σ
B01	118	**83**	83	0	**83**	83	0	**83**	83	0	**83**	83	0
B02	187	**84**	84	0	88.6	86	2.19	**84**	84	0	**84**	84	0
B03	203	**140.8**	139	0.75	142	142	0	/	/	/	/	/	/
B04	142	**62**	62	0	64.9	64	0.22	**62**	62	0	**62**	62	0
B05	102	**62**	62	0	62.5	62	0.95	**62**	62	0	**62**	62	0
B06	199	125	125	0	125	125	0	**124.6**	124	0.93	125	125	0
B07	173	**112**	112	0	**112**	112	0	/	/	/	/	/	/
B08	171	**107**	107	0	**107**	107	0	**107**	107	0	**107**	107	0
B09	229	**221**	221	0	**221**	221	0	**221**	221	0	**221**	221	0
B10	215	**87.9**	87	0.11	88	88	0	88	88	0	88	88	0
B11	180	**89**	89	0	**89**	89	0	**89**	89	0	**89**	89	0
B12	184	**177**	177	0	177.9	177	1.84	**177**	177	0.89	178.5	177	4.24
B13	139	**169**	169	0	170.4	169	0.94	169.3	168	0.83	172	168	2.44
B14	180	/	/	/	243.6	236	8.44	**237**	237	0	238.3	236	2.37
B15	194	332.1	328	5.22	335.3	323	5.45	323.7	322	1.55	**321.3**	319	1.31
B16	238	131.4	129	1.08	134.4	132	1.23	130.7	129	1.53	**129.3**	129	0.92
B17	180	**134**	134	0	/	/	/	134.5	134	0.61	134.3	134	0.44
B18	348	**219**	219	0	219.6	219	0.94	219.1	219	0.22	219.2	219	0.37

Table 7 Average tree costs of our SSPR algorithms and some existing heuristics and algorithms on random graphs

	Algorithms	Average Tree Cost
Heuristics	KPP1 [10]	905.581
	KPP2 [15]	911.684
	BSMA [14]	872.681
GA-based Algorithms	Wang et al. [20]	815.969
	Haghighat et al. [21]	808.406
TS-based Algorithms	Skorin-Kapov and Kos [25]	897.875
	Wang et al. [26]	869.291
	Youssef et al. [24]	854.839
	Ghaboosi and Haghighat [27]	739.095
Path relinking	Ghaboosi and Haghighat [28]	691.434
VNS Algorithms	VNDMR1 [31]	680.067
	VNDMR2 [31]	676.427
	Multi-VND [30]	653.257
GRASP Algorithms	GRASP-CST [29]	669.927
	GRASP-VND [30]	649.203
Scatter Search Algorithms	Our proposed SSPR-TS	679.690
	Our proposed SSPR-VND	**644.840**

4.3 Experiments on random graphs

In [28], a set of random network graphs is generated by Ghaboosi and Haghighat to test their proposed path relinking algorithm. Their simulations show that the path relinking algorithm outperforms a number of existing heuristics and algorithms in the literature, where the average tree cost of each algorithm over all these random graphs is reported. The same simulations have been conducted in our previous papers [30, 31] by generating a set of random networks as designed in [28]. Furthermore, in order to compare our SSPR algorithms with other existing heuristics and algorithms, a group of experiments have been carried out on the same set of random graphs in our previous work [30, 31]. These random graphs include 3 random topologies for each network size (10, 20, 30, 40, 50, 60, 70, 80, 90, and 100 nodes), so there are 30 random network graphs in total. For each network graph, edge costs depend on the length of the edges, with edge delays are set to 1. The multicast group size is considered to be 30% of the network size in each random graph. Delay bounds vary according to 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). The simulation results are shown in Table 7.

In Table 7, we can see that SSPR-VND has the best overall performance on the random graphs with respect to the average tree cost in comparison with other heuristics and algorithms in the literature. SSPR-VND outperforms SSPR-TS in terms of the average tree cost, showing that the better improvement method improves the performance of the SSPR meta-heuristic.

Table 8 presents more details of the average tree cost, standard deviation and computational time of these four algorithms on each network size. We observe that SSPR-VND performs the best by obtaining 7 best solutions out of the 10 different size networks, compared with GRASP-VND which achieves 4 best solutions, and GRASP-CST only finds 2 best results. Table 8 also shows that SSPR-VND is better on large problems, while GRASP-VND is better on smaller problems. The average standard deviation of SSPR-VND for the 10 network sizes is 4.8, which is better than that of SSPR-TS (11.39). It demonstrates again that SSPR-VND gives more stable and better quality solutions than SSPR-TS on all the tested random graphs. We also notice that SSPR-VND obtains better results by consuming longer computational time compared with other three algorithms.

5 Conclusions

In this paper, we have investigated a hybrid Scatter Search with Path Relinking (SSPR) for solving the Delay-Constrained Least-Cost (DCLC) multicast routing problem for

Table 8 Average tree cost and standard deviation of our SSPR algorithms compared with GRASP-VND and GRASP-CST on random graphs

Network size	SSPR-VND			SSPR-TS			GRASP-VND			GRASP-CST		
	Cost	σ	Time (s)	Cost	σ	Time (s)	Cost	σ	Time (s)	Cost	σ	Time (s)
10	**94.67**	0.00	0.039	97.93	2.25	0.017	**94.67**	0	0.008	**94.67**	0	0.009
20	272.53	2.41	0.314	274.4	1.97	0.141	272.07	2.25	0.085	**271.13**	1.48	0.048
30	393.5	3.57	1.453	414.1	14.96	0.577	**392.33**	0	0.353	394.67	0	0.156
40	513.33	0.00	3.522	533.43	3.22	1.291	**512.8**	1.55	0.857	526.47	1.79	0.388
50	**660.83**	0.53	9.575	707.43	8.27	2.727	662.33	1.94	2.109	697.07	3.43	0.815
60	**748.07**	7.03	13.637	759.07	17.92	5.271	757.33	13.48	3.894	761.13	17.13	1.625
70	**779.5**	5.97	29.608	811.2	5.57	9.08	780.83	2.96	9.029	797.53	1.64	2.648
80	**863.33**	5.93	66.356	935.33	22.65	15.791	868.87	7.73	19.421	902.67	5.49	5.941
90	**1132.77**	19.35	116.419	1215.17	21.07	31.595	1155.57	19.02	32.621	1201.93	18.02	10.27
100	**989.87**	3.19	177.454	1048.83	16.03	47.218	995.23	4.23	41.681	1052	23.4	10.983
Avg.	**644.84**	4.8	41.837	679.690	11.39	11.371	649.203	5.316	11.006	669.927	7.238	3.289

the first time. The problem is also known as the Delay-Constrained Steiner Tree problem and has been proved to be NP-complete. Although both scatter search and path relinking have shown to yield promising solutions for various optimization problems, little attention has been given to them for solving the constrained multicast routing problem.

In our proposed SSPR algorithm, the path relinking heuristic is applied as the combination method to incorporate attributes of high quality solutions. Two local search heuristics, tabu search and variable neighborhood search, have been designed and tested as the improvement methods in the SSPR meta-heuristic. A large number of simulations on small and medium sized problems from SteinLib in the OR-library and a group of random graphs demonstrate that SSPR with a variable neighborhood search outperforms SSPR with a tabu search in terms of average tree costs. This indicate that the improvement method will greatly affect the performance of the proposed SSPR meta-heuristic, and better local search can contribute to better results. Experiments demonstrate that our SSPR algorithm is able to find high quality solutions for the DCLC multicast routing problem in comparison with some existing algorithms and heuristics.

In our future work, the proposed SSPR meta-heuristic can be improved in different ways. For example, more combination methods may be applied and investigated. In addition, we plan to extend our SSPR meta-heuristic to solve more sophisticated multicast routing problems in reality, such as the multicast routing problem with multiple QoS constraints (e.g. the bandwidth, delay-variation, node degrees) or the dynamic multicast routing problem with multicast members leaving and joining the multicast group at various times during the connection.

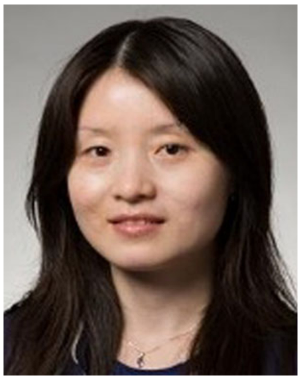
Acknowledgement This research was supported by Hunan University, China, and the School of Computer Science at The University of Nottingham, UK.

References

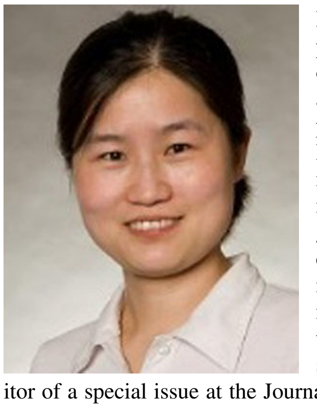
1. Cheng X, Du DZ (2001) Steiner trees in industry. Kluwer Academic, Dordrecht
2. Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. Freeman, New York
3. Guo L, Matta I (1999) QDMR: An efficient QoS dependent multicast routing algorithm. In: Proceedings of the 5th IEEE real time technology and applications symposium, pp 213–222
4. Salama HF, Reeves DS, Viniotis Y (1997) Evaluation of multicast routing algorithms for real-time communication on high-speed networks. IEEE J Sel Areas Commun 15:332–345
5. Yeo CK, Lee BS, Er MH (2004) A survey of application level multicast techniques. Comput Commun 27:1547–1568
6. 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 (2006) Research challenges in QoS routing. Comput Commun 29:563–581
7. Diot C, Dabbous W, Crowcroft J (1997) Multipoint communication: a survey of protocols, functions, and mechanisms. IEEE J Sel Areas Commun 15:277–290
8. Oliveira CAS, Pardalos PM (2005) A survey of combinatorial optimization problems in multicast routing. Comput Oper Res 32(8):1953–1981
9. Beasley JE (1990) OR-Library: distributing test problems by electronic mail. J Oper Res Soc 41(11):1069–1072. http://people.brunel.ac.uk/~mastjjb/jeb/orlib/steininfo.html
10. Kompella VP, Pasquale JC, Polyzos GC (1993) Multicast routing for multimedia communication. IEEE/ACM Trans Netw 1:286–292
11. Widyono R (1994) The design and evaluation of routing algorithms for realtime channels. Technical Report, ICSI TR-94-024, International Computer Science Institute, UC Berkeley
12. Sun Q, Langendoerfer H (1995) Efficient multicast routing for delay-sensitive applications. In: Proceedings of the 2nd workshop on protocols for multimedia systems, pp 452–458
13. Sun Q, Langendoerfer H (1997) An efficient delay-constrained multicast routing algorithm. Technical Report, Internal Report, Institute of Operating Systems and Computer Networks, TU Braunschweig, Germany
14. Zhu Q, Parsa M, Garcia-Luna-Aceves JJ (1995) 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). IEEE Comput Soc, Boston, pp 377–385

15. Kompella VP, Pasquale JC, Polyzos GC (1993) Two distributed algorithms for the constrained Steiner tree problem. In: Proceedings of the 2nd international conference on computer communications and networking, pp 343–349
16. Shaikh A, Shin K (1997) Destination-driven routing for low-cost multicast. IEEE J Sel Areas Commun 15:373–381
17. Jia X (1998) A distributed algorithm of delay-bounded multicast routing for multimedia applications in wide area networks. IEEE/ACM Trans Netw 6:828–837
18. Wang XL, Jiang Z (2004) QoS multicast routing based on simulated annealing algorithm. In: Proceedings international and applications, pp 511–516
19. Kun Z, Heng W, Feng-Yu L (2005) Distributed multicast routing for delay variation-bounded Steiner tree using simulated annealing. Comput Commun 28:1356–1370
20. Wang Z, Shi B, Zhao E (2001) Bandwidth-delay-constrained least-cost multicast routing based on heuristic genetic algorithm. Comput Commun 24:685–692
21. Haghighat AT, Faez K, Dehghan M, Mowlaei A, Ghahremani Y (2004) GA-based heuristic algorithms for bandwidth-delay-constrained least-cost multicast routing. Comput Commun 27:111–127
22. Zahrani MS, Loomes MJ, JA Malcolm, Dayem Ullah AZM, Steinhofel K, Albrecht AA (2008) Genetic local search for multicast routing with pre-processing by logarithmic simulated annealing. Comput Oper Res 35:2049–2070
23. Kim SJ, Choi MK (2007) Evolutionary Algorithms for route selection and rate allocation in multirate multicast networks. Appl Intell 26(3):197–215
24. Youssef H, Al-Mulhem A, Sait SM, MA Tahir (2002) QoS-driven multicast tree generation using tabu search. Comput Commun 25(11–12):1140–1149
25. Skorin-Kapov N, Kos M (2003) 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
26. Wang H, Fang J, Wang H, Sun YM (2004) TSDLMRA: an efficient multicast routing algorithm based on tabu search. J Netw Comput Appl 27:77–90
27. Ghaboosi N, Haghighat AT (2006) A tabu search based algorithm for multicast routing with QoS constraints. In: 9th international conference on information technology, pp 18–21
28. Ghaboosi N, Haghighat AT (2007) A path relinking approach for delay-constrained least-cost multicast routing problem. In: 19th international conference on tools with artificial intelligence, pp 383–390
29. Skorin-Kapov N, Kos M (2006) A GRASP heuristic for the delay-constrained multicast routing problem. Telecommun Syst 32(1):55–69
30. Xu Y, Qu R (2009) A GRASP approach for the delay-constrained multicast routing problem. In: Proceedings of the 4th multidisciplinary international scheduling conference (MISTA4). Dublin, Ireland
31. Qu R, Xu Y, Kendall G (2009) A variable neighborhood descent search algorithm for delay-constrained least-cost multicast routing. In: Stützle T (ed) Proceedings of learning and intelligent optimization (LION3). LNCS, vol 5851. Springer, Berlin, pp 15–29
32. Betsekas D, Gallager R (1992) Data networks, 2nd edn. Englewood Cliffs, Prentice-Hall
33. Cormen TH, Leiserson CE, Revest RL (1997) Introduction to algorithms. MIT Press, Cambridge
34. Eppstein D (1998) Finding the k shortest paths. SIAM J Comput 28(2):652–673
35. Glover F (1998) A template for scatter search and path relinking. In: Hao JK, Lutton E, Ronald E, Schoenauer M, Snyers D (eds) Artificial Evolution. LNCS, vol 1363. Springer, Berlin, pp 3–51
36. Glover F, Laguna M, Martí R (2000) Fundamentals of scatter search and path relinking. Control Cybern 29(3):653–684
37. Glover F, Løkketangen A, Woodruff DL (2000) Scatter search to generate diverse MIP solutions. In: Laguna M, González-Velarde JL (eds) OR computing tools for modeling, optimization and simulation: interfaces in computer science and operations research, pp 299–317
38. Drias H, Khabzaoui M (2001) Scatter search with random walk strategy for SAT and MAX-W-SAT problems. In: LNCS, vol 2070. Springer, Berlin, pp 35-44
39. Hamiez JP, Hao JK (2002) Scatter search for graph coloring. In: LNCS, vol 2310. Springer, Berlin, pp 168–179
40. Tang J, Zhang J, Pan Z (2010) A Scatter search for solving vehicle routing problem with loading cost. Expert Syst Appl 37(6):4073–4083
41. Bastos MP Ribeiro CC (1999) Reactive tabu search with path relinking for the Steiner problem in graphs. In: Proceedings of the third metaheuristics international conference, Angra dos Reis, Brazil
42. Waxman BM (1988) Routing of multipoint connections. IEEE J Sel Areas Commun 6:1617–1622



Ying Xu is currently a PhD student in the School of Computer Science at the University of Nottingham. She is also a Lecturer in the School of Computer Science in Hunan University of China. Her main research interests focus on heuristics and meta-heuristics for solving optimisation and scheduling problems



Rong Qu gained her PhD in Computer Science from the University of Nottingham in 2002, and now is a Lecturer in the School of Computer Science at the University. Her main research interests include meta-heuristics, knowledge-based methodologies and constraint programming on modeling and solving complex optimisation and scheduling problems. She has published more than 40 refereed papers in international journals and conferences since 2000, and acted as a guest editor of a special issue at the Journal of Scheduling, the program chair of four workshops, and a committee member for more than 10 conferences in the last two years.