Metaheuristic Approaches for QoS Multicast Routing Problems

by

Ying Xu, BSc, MSc.

Thesis submitted to the University of Nottingham for the degree of Doctor of Philosophy

January 2011

Contents

List of Figures

List of Tables

Abstract

The rapid evolution of real-time multimedia applications requires the Quality of Service (QoS) based multicast routing in underlying communication networks. This thesis is concerned with the investigation of metaheuristic approaches for QoS multicast routing problems. The underpinning mathematical structure of the QoS multicast routing problem is the constrained Steiner Tree problem which is a well-known NP-complete problem. Metaheuristics are a family of generic high level search strategies that can be adapted for tackling many optimisation problems, including the QoS multicast routing problems concerned in the thesis.

This thesis contains two main parts to systematically investigate metaheuristic approaches on different QoS multicast routing problems. In the first part, four new metaheuristic approaches have been developed and investigated for the Delay-Constrained Least-Cost (DCLC) multicast routing problem. The performance of these proposed metaheuristics is evaluated by means of extensive experiments on a number of benchmark instances from the OR-library and some random networks generated by a multicast routing simulator. Experimental results demonstrate that the proposed metaheuristic approaches are promising solutions for the DCLC multicast routing problem and superior to other existing algorithms in the literature. The first theoretical analysis of the fitness landscape on the DCLC multicast routing problem has been conducted on a set of benchmark instances. Two fitness landscape analysis techniques, the fitness distance correlation analysis and the auto-correlation analysis, have been used to investigate the underlying landscape characteristics of the problem. The second part of the thesis investigates multi-objective multicast routing problems. In recent research, the multi-objective multicast routing problem has emerged with multiple inter-dependent and conflicting objectives while satisfying certain constraints. Two novel hybrid metaheuristic approaches, a hybrid evolutionary multi-objective simulated annealing algorithm with variable neighbourhoods and a simulated annealing based multi-objective genetic local search algorithm, have been proposed and investigated for solving the multi-objective multicast routing problem concerning multiple objectives simultaneously. A large amount of systematic experiments have been performed to demonstrate the effectiveness and efficiency of both hybrid metaheuristics for the problem.

Acknowledgements

I would like to thank my financial sponsors, the Hunan University (China) and the University of Nottingham (UK), for providing the opportunity to study in UK.

I would like to express my deepest gratitude and admiration to my first supervisor Dr. Rong Qu for her guidance, ideas, advice and continuous support during my PhD program. I also thank my second supervisor Prof. Graham Kendall for his valuable comments and great help.

I would like to thank all members of the ASAP Group to make my PhD study within ASAP an enjoyable and memorable experience. Many thanks to Dr. Dario Landa-Silva for his valuable discussions and advice with regard to my research. Special thanks go to Juan Castro-Gutiérrez, Hui Li, Francisco Romero Campero, Janathan Blakes, Fang He and Huanlai Xing for their help not only to my research but also to my social life during my PhD career.

I would like to dedicate this work to my parents, my brother, my husband, my son, all my family and friends for their great supports and encouragements throughout my PhD.

CHAPTER 1. Introduction

The first chapter contains three introductory sections. The first section introduces the background and motivations of the research topics. The second section summarizes the issues together with the research aims of the work included in the thesis and finally, the third section gives a general overview of the thesis.

1.1 Background and Motivations

The enormous investments in communication and switching technologies have resulted in a new generation of high speed networks. Generally speaking, communication networks can be classified into two categories: local area networks (LANs) and wide area networks (WANs). A LAN covers a small geographical region, for instance a room, a building or a cluster of buildings. A WAN represents a computer network that covers a broad area, such as a metropolis, a nation, or several nations. For example, the Internet is a typical WAN interconnected by millions of computer networks to provide an internet service for global users worldwide. With the rapid growth of internet infrastructure, size, services and number of users, many issues and challenges, such as the number and locations of network nodes (computers or routers), routing mechanisms, resource reservation, flow scheduling, and admission control, etc, have arisen in computer communication networks. More details can be found in Minoux (1989); Amiri and Pirkul (1997); and Queiroz and Jr (2003). Among them, network routing is a fundamental engineering task in any WAN, especially for today's internet. For a given network, the purpose of the network routing is to send information (data streams or packets, etc.) through a path or a set of paths from a source to a destination or an end user. Its main objective is to reduce the total cost while maximising the utilisation of network resources.

Depending on the number of destinations, network routing can be categorized into three basic types: unicast (one to one), broadcast (one to all) and multicast (one to many). Unicast is the traditional point-to-point communication where the information is sent through a path from one source to one destination. Broadcast is when information is sent from one source to all nodes in the network. Multicast aims at sending information from one or more sources to a set of destinations in the network. Figure 1.1 presents an example network of multicast routing. Multicast can be considered as a generalization of both unicast and broadcast, since unicast is the special case of multicast when the number of destination nodes is one. If the destination nodes include all nodes in the network, then the routing becomes broadcast. Network routing is a complex problem in large networks due to the significantly large number of possible paths via the potential intermediate nodes a packet might traverse before reaching its final destination. Thus, the routing algorithm has a significant impact on the development and performance of computer communication networks.

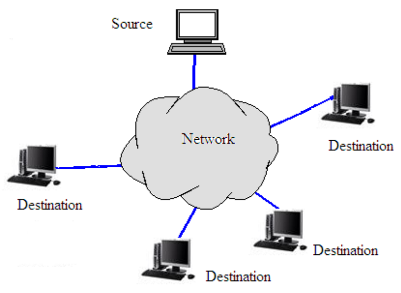


Figure 1. 1 An example multicast routing network

1.1.1 Quality of Service (QoS) Routing

Nowadays, the Internet Protocol (IP) based networks are rapidly becoming the main carriers of various multimedia applications, including video/audio conferencing, distance education, E-commerce, interactive distributed games and internet telephony, etc. These real time multimedia applications have a wide diversity of Quality of Service (QoS) (Ma and Steenkiste, 1997; Xiao and Ni, 1999) requirements for traffic, including the cost, bandwidth, end-to-end delay, delay jitter, packet loss ratio and hop count, etc. The notion of QoS in networks means the data delivery service should satisfy certain performance requirements or metrics. These metrics define the QoS guarantees that a network should provide. However, originally, the Internet was designed to support the best effort service for time independent information streams such as file transfer and electronic mail, and thus no QoS guarantees can be provided. In order to support QoS requirements, new challenges for network routing have emerged to satisfy certain constraints defined by the QoS requirements.

QoS routing (Paul and Raghavan, 2002a) in computer networks is defined as the process of transferring information from a source to a destination (or a group of destinations) through network elements, including hosts and routers, under certain constraints or performance metrics. It means that QoS constrained network routing must be able to utilise the network resources efficiently while providing the requested QoS requirements. QoS routing is a difficult task due to the following reasons. The multimedia applications have very diverse QoS requirements which make the routing problem intractable. In reality, the network state changes dynamically due to the nodes joining in or leaving the network, link congestions, and transmission failures. The increasing network size makes it very hard to gather updated state information in a dynamic environment. The QoS routing problem has been proven to be a NP-complete problem (Wang and Hou, 2000) due to the high computation complexity of searching a path based on certain combinations of QoS metrics. More details about research challenges in QoS routing can be found in (Masip-Bruin *et al.*, 2006). For these reasons, fast and efficient algorithms are required to support QoS routing for multimedia transmission.

In general, QoS routing problems can be classified into two major classes, namely unicast routing and multicast routing. QoS unicast routing is relatively easier, and is defined as finding a route from a single source to a single destination node that satisfies a set of QoS constraints. On the other hand, QoS multicast routing (Striegel and Manimaran, 2002) can be defined as follows: for a given source node, a set of destination nodes, a set of QoS constraints, and an optimisation objective, the goal is to send information from the source to all destinations simultaneously, while optimising the objective and satisfying certain QoS constraints. A QoS unicast routing problem can be treated as a special type of QoS multicast routing problem with only one destination, while the latter is more complex than the former as finding paths between a source and several destinations is more difficult than finding a path between a source and a single destination.

1.1.2 QoS Multicast Routing

Since the 1990s, the rapid evolution of numerous real time multimedia applications has been stimulating the demand for QoS based multicast routing in the underlying computer communication networks. QoS multicast routing is an important communication technique to support data transmission in computer networks; it aims at transferring information simultaneously from one or more sources (or senders) to a group of destinations (or receivers), namely the multicast group, while satisfying a set of QoS constraints. The thesis addresses the multicast routing problem with single source. A multicast group may be either dense-mode, in which group members are densely distributed in the network, or sparse-mode, where members are widely distributed in a sparse environment. A solution to the QoS multicast routing problem is to build a multicast tree, which spans from the source node to all destinations and satisfies certain QoS requirements. The main goals of QoS multicast routing are to efficiently allocate network resources, balance the network load, reduce congestion hot spots and provide adequate QoS guarantees for end users of multimedia applications (Wang and Crowcroft, 1996; Chen and Nahrstedt, 1998; Paul and Raghavan, 2002). Conventional unicast routing is inefficient, since it relies on point-to-point transmission and requires the source to build an independent path to every destination, and the same information might travel the same link many times, thus creating unnecessary traffic on the link and wasting network resources. By avoiding sending data packets from the source to each destination separately, multicast routing can utilise network resources more efficiently, as a data packet traverses each link only once, and some of the links are shared.

Real world multicast applications have a variety of QoS requirements. Some commonly required QoS metrics are briefly introduced as follows. First, in order to make good use of network resources, it is important to minimise the total cost of the multicast tree during the transmission. The total cost of the multicast tree is the sum of costs of links included in the tree, where each link cost is normally defined as the bandwidth utilisation or network resource cost on the link. Second, most real world applications of multicast routing problems are delay-sensitive for data delivery. For example, in interactive video/audio conferencing, the video/audio is expected to be consistent among all participants, without excessive delay. The delay-constrained multicast routing problem is thus defined as sending the video/audio from the sender to each participant within a predefined delay threshold. The end-to-end delay from the sender to each participant is the sum of all delays of links along the transimission path. In addition, as it is desirable for all participants to hear and see the speakers at nearly the same time, another time related constraint is the delay jitter (variation), which is defined as the difference between the minimum and maximum delay among all the end-to-end delays from the source to each destination. Many real-time applications, such as video conferencing and interactive games, are bandwidth-intensive. The multicast routing problem with bandwidth constraint is the requirement that the bandwidth of each link, i.e. the capacity of the link, in the constructed multicast tree should be above a bandwidth threshold. Finally, because congestion on the link may cause delay or failure of information transmission, it is important to minimise the maximal link utilisation, which reflects the congestion level in the current network, within the multicast tree. The link utilisation is defined as the ratio of the used link resource to the link capacity.

1.1.3 Multicasting on the Internet

Multicasting is the ability of a communication network to deliver information from a source to multiple destinations at different locations. Multicast routing can be easily implemented by broadcasting in a LAN since a small number of nodes are included in the LAN, while implementing multicasting in a WAN presents a challenge due to the large number of nodes included in the WAN. The work in this thesis therefore focuses on the multicast routing problems in WANs which may be connections of LANs. As a typical WAN, the Internet is the main carrier of the emerging real time multimedia applications; QoS multicast routing on the Internet thus becomes an important mechanism to support data transmissions to multiple end users of these multimedia applications. The following subsections briefly introduce the architecture of the Internet and then summarize some existing multicast routing protocols employed on the Internet.

1.1.3.1 The Architecture of the Internet

Today's Internet uses the standard Internet Protocol Suite (TCP/IP, i.e. Transmission Control Protocol /Internet Protocol) to provide services for billions of users. The Internet Service Providers (ISPs) connect their networks via Internet Exchange Points (IXP) to the Internet. For administration purposes, the Internet is divided into many administrative domains called Autonomous Systems (ASs). An AS is a connection of networks and gateway routers under the control of its own internal policy to collect and exchange information. Within an AS, routing is achieved by using the Interior Gateway Protocol (IGP), while the Exterior Gateway Protocol (EGP) passes the routing information across different ASs.

Originally, the architecture of the Internet and the Internet Protocol (IP) at the network layer were designed on the simple concept of best-effort delivery. It means the network provides no guarantee that each packet will reach its destination nor that it can arrive within a certain time. The reliability of the Internet depends on the Transmission Control Protocol (TCP) at the transport layer of the network to keep track of packets and then retransmit if required. However, the mechanisms used in TCP, such as the flow and congestion control schemes, are not sufficient to provide QoS support for either real-time continuous multimedia applications or multicast routing. In 1992, the Internet Engineering Task Force (IETF) firstly provided a multicast service on the Internet over the multicast backbone (Casner and Deering, 1992). Since then, several multicast routing protocols have been proposed to support multicasting on the Internet.

1.1.3.2 Multicast Routing Protocols on the Internet

A multicast routing protocol should be able to collect and maintain state information, select the most appropriate path among the various paths available from the sender(s) to receivers, and handle the changes in the networks or the multicast group. Based on where the multicast routing protocol is deployed, the Internet employs three types of multicast routing protocols. The first type of protocol is implemented on a host to support its joining or leaving a multicast group. The Internet Group Management Protocol (IGMP) (Deering *et al*., 1989) is an example of this type of protocol. The second type of protocol is the Multicast Interior Gateway Protocol (MIGP), which is deployed on multicast routers to enable multicast routing within an AS. Some examples of MIGPs include the Distance Vector Multicast Routing Protocol (DVMRP) (Waitzman *et al*., 1988); Multicast Open Shortest Path First (MOSPF) (Moy, 1994); Core Based Tree (CBT) (Ballardie *et al*., 1993); and Protocol Independent Multicast (PIM) (Deering *et al*., 1996). The third type of protocol is applied on border routers to allow multicast routing to transfer between different ASs. The Border Gateway Multicast Protocol (BGMP) (Thaler *et al*., 1998) belongs to this type of protocol.

IGMP is employed by a host to support multicast communication between the host and the neighbouring multicast router. It enables a multicast router to keep track of changes in the multicast group by sending messages between the host and the multicast router. For example, if a host joins a new group, it sends a join-group report to the multicast router. When a host leaves a group, it sends a leave-group report to the multicast router.

DVMRP is the first protocol implemented for IP multicast routing. DVMRP is widely used by most multicast routers of the Internet for supporting multicasting. DVMRP forwards packets through a multicast tree which is rooted at the source and spans all destinations by the reverse shortest paths aggregating these shortest paths into the tree. The Bellman-Ford algorithm (Bertsekas and Gallager, 1992) is used in DVMRP to build the unicast routing table and find the shortest paths for each destination. In addition, DVMRP supports dynamic multicast which allows destination nodes to join or leave dynamically during the multicast routing session. This is realized by periodically broadcasting the source packets to all nodes in the network. Both the unicast routing table and the broadcasting scheme used in DVMRP cause serious performance problems, especially when the number of destination nodes is large; thus DVMRP is inefficient and inflexible for large networks.

MOSPF uses the Dijkstra's shortest path algorithm (Bertsekas and Gallager, 1992) to create the shortest path trees from the source to all destination nodes. In the Dijkstra's algorithm, at each step, a new node with the minimum path cost from the source is added to the current tree in a greedy manner. After building a multicast tree, MOSPF prunes the non-destination leaves in the tree so that the network resource can be saved. The disadvantages of MOSPF are the scalability and computational resources. MOSPF needs extra computational time to prune multicast trees which is time consuming in large networks and is therefore impractical in large-scale networks. In addition, other QoS requirements are not considered in MOSPF since the Dijkstra's algorithm only creates the shortest paths in terms of one QoS requirement, usually the cost.

In order to reduce the state information stored at each router, CBT protocol is designed to build a single delivery tree rooted at a core router for each multicast group, regardless of the location of the source node. The delivery tree of a multicast group is shared by all nodes that send messages to the destinations of the multicast group. Messages from a source node to any group are sent toward the core router by a unicast path until these messages reach a router which is included in the corresponding delivery tree. A new issue arising in CBT is the selection of the core of the delivery tree, which is the key factor in constructing multicast trees. The selection of the core is another optimisation problem since it is difficult to choose a core which yields optimal routs for all multicast group members.

PIM is developed to provide the scalability of multicast routing protocol in large networks. It has two modes of operation: PIM-Dense Mode (PIM-DM) and PIM-Sparse Mode (PIM-SM). PIM-DM is similar to DVMRP, both building the reverse shortest paths for destinations, but PIM-DM is more flexible as it works independently of a particular unicast routing protocol while working with any one that is available. In the cases where multicast destination nodes are sparsely distributed in the network, PIM-SM can choose a centre router for each multicast group. During the multicast routing, data packets are sent from the source to the centre router first and then directed to the destination nodes. Due to the complexity of PIM, it is difficult to deploy on large-scale networks.

BGMP is employed on border routers to support multicast communication across different ASs. BGMP consists of two components: the MIGP component and the BGMP component. The MIGP component employs the MIGP protocol within an AS. The BGMP component enables multicasting between ASs by constructing a bidirectional centre-based tree to connect the border routers of different ASs which include the destination nodes. The root of the centre-based tree is the entire AS which claims the multicast transmission.

In general, a multicast routing protocol consists of a set of functional components to consider all possible situations in real world networks, such as the collection and update of state information, the construction of multicast trees, error control, flow scheduling, failure handling and connection management. The above mentioned multicast routing protocols all have their strengths and drawbacks. The main shortcoming of the above multicast routing protocols is their lack of providing any QoS support. For example, none of these protocols is designed to find a delay-bounded multicast tree that satisfies the delay constraint for the delay-sensitive application. Furthermore, these protocols do not consider the capacity constraint or other QoS requirements in the network. To support QoS on the Internet, other techniques have also been proposed, such as the Resource Reservation Protocol (RSVP) which aims to provide resource reservations for both unicast and multicast data flows (Braden *et al*., 1997), and the differentiated services architecture (Black *et al*., 1998) which provides differentiated quality of services by assigning different priority levels to data packets as requested by users or applications. For more detailed discussions of multicast routing protocols, the reader may refer to some surveys in Sahasrabuddhe *et al*. (2000); Wang and Hou (2000); Striegel and Manimaran (2002); and Paul and Raghavan (2002b).

Some recent multicast routing protocols for supporting QoS requirements have been proposed (e.g. Faloutsos *et al*., 1998; Chen *et al*., 2000; Chen and Shavitt, 2004). These QoS multicast routing protocols normally reply on simple unicast routing algorithms to iteratively grow a search tree by selecting the shortest paths which connect each new member to an on-tree node in terms of one particular QoS requirement. However, these shortest paths may not be the best choice for the other QoS requirements. As previously mentioned, the real-time multimedia applications on the Internet involve group users and demand different QoS requirements. Thus, multicast routing protocols need more intelligent and effective routing algorithms to support diverse QoS requirements in real world applications. To support QoS multicast routing on the Internet, the component for constructing multicast trees in a multicast routing protocol plays an important role. As a result, many multicast routing algorithms, which can be used as a component in the protocol to support multicast routing, have been proposed in the literature. A well designed multicast routing algorithm in a multicast routing protocol can greatly improve the performance of the multicast routing protocol. This thesis focuses on designing efficient and effective QoS multicast routing algorithms that optimise the network resource utilisation and provide adequate QoS guarantees for end users.

1.1.4 Motivations

The multicast routing problem without QoS constraints, i.e. the unconstrained multicast routing problem, is equivalent to the Minimum Steiner Tree Problem in Graphs (MStTG) (Hwang and Richards, 1992). The MStTG problem aims to search for a minimum cost Steiner tree spanning a subset of nodes in the graph. The MStTG problem is a well know NP-complete problem (Garey and Johnson, 1979) and has been widely studied for decades, while it still presents a great research challenge. Many real-time multimedia applications require the underlying computer network to provide QoS multicast communication which satisfies certain QoS requirements. The QoS multicast routing problem is subject to several QoS constraints, and can be seen as the constrained MStTG problem. The QoS multicast routing problem is thus also NP-complete problem, in which more QoS constraints need to be considered in addition to finding a minimum cost multicast tree. This can also be proven by the fact that the complexity of finding a path subject to two or more independent metrics in any possible combination is NP-complete (Wang and Crowcroft, 1996; Chen and Nahrestedt, 1998). The QoS multicast routing problem is mainly considered as a single-objective optimisation problem subject to some QoS constraints in the literature. In recent research, another more challenging QoS multicast routing problem which optimises multiple objectives in real world applications, namely the multi-objective multicast routing problem, has emerged (Fabregat *et al.*, 2005).

The importance and challenge of the QoS multicast routing problem have led to increased research interest from the scientific community. As a result, various optimisation algorithms, such as exact methods, heuristics, and metaheuristics, have been proposed for solving QoS multicast routing problems since the 1990s. As pointed out by Salama *et al.* (1997), most of the early heuristic algorithms are based on the deterministic procedure, and are either computationally expensive or inefficient in tackling the multicast routing problems. In recent research, the majority of state-of-the-art algorithms for multicast routing problems are based on metaheuristics including simulated annealing, genetic algorithm, tabu search, greedy randomized adaptive search algorithms (GRASP) and path relinking, etc.

The research in this thesis focuses on applications of metaheuristic approaches for solving QoS multicast routing problems in computer networks. Metaheuristics are general high-level frameworks that coordinate simple heuristics and rules to find good approximate solutions to computationally difficult optimisation problems. They are among the most effective solution strategies for solving optimisation problems in practice and have been applied to a very wide variety of problems in telecommunications, computer communications, and network design and routing (Martins *et al*., 2004; Ribeiro *et al*., 2007). Despite the increasing research attention of metaheuristics which deal with many real-world complex optimisation problems, the investigation of metaheuristics for QoS multicast routing problems has not been fully explored. Some recent metaheuristics, such as variable neighbourhood search, scatter search and jumping particle swarm optimisation have not been applied for variants of the problem. The motivation of the work in this thesis is to investigate the recent and hybrid metaheuristic methods for QoS multicast routing problems and to evaluate the proposed metaheuristic algorithms by simulation experiments. In addition, although the theoretical analysis of fitness landscape has been shown to be useful for observing the behaviour of search algorithms and thus can help in predicting their performance (Wright, 1932), few research efforts have been made to analyse the underlying landscape features of the multicast routing problem. The lack of the theoretical analysis of the landscapes for the multicast routing problem motivates the fitness landscape analysis of the problem within this thesis. Furthermore, this thesis has designed and evaluated some hybridized metaheuristic approaches for the more complicated multi-objective multicast routing problem.

1.2 Issues and Aims

1.2.1 Issues Related to QoS Multicast Routing

To provide QoS to multicast routing is a difficult task due to a number of issues. First, numerous real-time multimedia applications have diverse QoS requirements such as the cost, bandwidth, delay, delay jitter, and packet loss ratio. Multiple constraints often make the multicast routing problem intractable. Second, many practical issues (e.g. state information collection and update, handling network dynamics and multicast group changes, tree maintenance and scalability) have to be taken into consideration when integrating a multicast routing algorithm into a multicast routing protocol (Wang and Hou, 2000).

There are several ways of classifying QoS multicast routing problems, depending on the considered QoS requirements, there are different classes of QoS multicast routing problems (Wang and Hou, 2000). For example, one well studied multicast routing problem is the delay constrained least cost multicast routing problem, where the objective is to minimise the total cost of the multicast tree while satisfying the end-to-end delay constraint from the source to each destination. One interesting version of the multicast routing problem is to construct a minimum cost multicast tree where the degree of each node is constrained. The degree of a node is the number of links which connect the node and are included in the tree. This is particularly useful in high speed networks, where the speed requirement can tolerate a bounded number of copies of the received information at each node. Some multicast routing problems concern multiple QoS requirements simultaneously. An example of such problems as these is to build a minimum cost tree while satisfying several QoS requirements including the end-to-end delay, delay jitter and bandwidth.

According to the state of the multicast group, i.e. the group of destination nodes, multicast routing problems can also be divided into two categories: the static (off-line) multicast routing problem, if the multicast group remains unchanged, and the dynamic (on-line) multicast routing problem with the members joining or leaving the multicast group at any time. In the static multicast routing problem, members in the multicast group remain unchanged, while in the dynamic multicast routing problem, nodes can leave or join in the multicast group during the multicast session. Clearly the multicast routing problem with the dynamic multicast group is more complicated due to the uncertainty of which nodes can be added or removed; another difficulty of the dynamic multicast routing problem is how to maintain and rearrange the multicast tree after the changes in the multicast group. This thesis focuses, however, on the static multicast routing problem. The reason is that research on the static multicast routing problem can be seen as a basis for more advanced research on the dynamic multicast routing problem. The algorithms proposed for solving the static multicast routing problem in the thesis can be extended and adopted for solving the dynamic multicast routing problem in the future.

A further classification of multicast routing problems is in terms of the number of multicast trees to be constructed in the problem. For simplicity, many algorithms in the literature consider one multicast tree at a time; however, in reality there may be more than one multicast group concurrently. A complicated multicast routing problem therefore arises, namely the group multicast routing problem. This type of multicast routing problem is actually a scheduling problem, where a group of multicast routing requests needs to be organized at the same time. In this case, the network resource must be shared according to the requirements of multiple multicast groups.

1.2.2 Two Case Studies

The focus of this thesis is the applications of metaheuristic methods to the QoS multicast routing problem. Two types of QoS multicast routing problems have been chosen as the case studies in this thesis: one is the Delay-Constrained Least-Cost (DCLC) multicast routing problem, a widely studied single-objective multicast routing in the literature; the other is a multi-objective multicast routing problem with multiple objectives to be optimised simultaneously.

The DCLC multicast routing problem concerns the two commonest and most important QoS requirements in real time applications: to minimise the total cost of the multicast tree while satisfying the end-to-end delay bound. The cost of a multicast tree is defined as the total cost of all the links included in the tree. The end-to-end delay is the total delay of the links along the path from the source to each destination in the multicast group. The DCLC multicast routing problem can be reduced to the Delay-Constrained Steiner Tree (DCST) problem, which is known to be NP-complete (Guo and Matta, 1999). The DCLC multicast routing problem thus demands effective and efficient intelligent algorithms.

At the early stage of research in the literature, the multicast routing problems are mainly defined and solved as a single-objective optimisation problem subject to some QoS constraints. When different conflicting objectives and realistic constraints are considered simultaneously, the multicast routing problems can be formulated as more complicated multi-objective optimisation problems. With the recent developments in multi-objective research, multi-objective optimisation algorithms have been investigated for multicast routing problems (Roy *et al*., 2002; Cui *et al*., 2003; Roy and Das, 2004; Crichigno and Baran, 2004a; Crichigno and Baran, 2004b; Koyama *et al*., 2004; Diego and Baran, 2005; Fabregat *et al*., 2005; Li *et al*., 2007; and Rai *et al*., 2010). In this thesis, a multi-objective multicast routing problem that simultaneously minimises five real world objectives, including the total tree cost, the maximal end-to-end delay, the delay jitter, the average delay and the link utilisation of the constructed multicast tree, has been selected as another case study.

1.2.3 Aims

With the rapid emergence of multimedia applications, QoS multicast routing is a key requirement of the underlying networks that support such applications. Due to the complexity and challenge of various QoS multicast routing in real world applications, multicast routing problems have attracted significant research attention in the area of computer networks and algorithmic network theory (Salama *et al*., 1997; Diot *et al*., 1997; Yeo *et al*., 2004; Oliveira and Pardalos, 2005; and Masip-Bruin *et al*., 2006) since the 1990s. As a result, many algorithms have been proposed for solving variants of QoS multicast routing problems in the past two decades.

Overall, the aim of this research is to investigate effective and efficient metaheuristic approaches for solving the two types of multicast routing problems. In addition, the fitness landscape of the DCLC multicast routing problem has been analysed for the first time to provide the theoretical foundations of the landscape characteristics on the multicast routing problem. Specifically, the thesis aims to:

1. Identify important issues of various QoS multicast routing problems in computer communication networks.
2. Investigate new metaheuristic approaches for solving the DCLC multicast routing problem.
3. Analyse the fitness landscape of the DCLC multicast routing problem.
4. Design novel hybridized metaheuristic algorithms for solving the multi-objective multicast routing problem.
5. Evaluate and compare the respective performance of the proposed metaheuristic algorithms on the two cases of multicast routing problems by intensive experiments on some benchmark problems and random networks.

1.3 Overview of the Thesis

The thesis is organized as follows: Chapter 2 overviews some mostly studied metaheuristic methods that have been adapted for solving a wide range of optimisation problems in the field of operational research. The principles associated with these metaheuristics are discussed. Brief overviews of metaheuristics for both combinatorial and multi-objective optimisation problems are also presented in this chapter. Chapter 3 introduces various QoS multicast routing problems in the applications of computer communication networks. As case studies in this thesis, two QoS multicast routing problems, the DCLC multicast routing problem and the multi-objective multicast routing problem, have been formally defined and the related work has been reviewed. Chapter 4 presents the proposed metaheuristic approaches for the DCLC multicast routing problem. Four new metaheuristic algorithms, including a variable neighbourhood descent search, a GRASP algorithm, a jumping particle swarm optimisation and a scatter search algorithm, have been implemented and investigated for solving the DCLC multicast routing problem. The large amount of experimental results has demonstrated the performance of the proposed algorithms on a set of benchmark instances and a group of random networks compared with other algorithms and heuristics in the literature. Chapter 5 analyses the fitness landscape of the DCLC multicast routing problem for the first time. Two fitness landscape analysis methods, the fitness distance analysis and auto-correlation analysis, have been applied to analyse the landscape of the problem. Experimental study has been carried out on a set of benchmark instances and the landscape features of the problem have been discussed in this chapter. Chapter 6 specifically studies two novel hybridized metaheuristic frameworks for the multi-objective multicast routing problem. An evolutionary multi-objective simulated annealing algorithm with variable neighbourhoods has been proposed and investigated on some benchmark networks as well as on some random networks. The advantages of the proposed hybridized algorithm have been discussed in comparison with two conventional evolutionary algorithms. Furthermore, a simulated annealing based genetic local search algorithm has also been designed and investigated for the multi-objective multicast routing problem. Experimental results show that simulated annealing based strategies integrated in the genetic local search algorithm contribute to better solutions for the multi-objective multicast routing problem compared with other algorithms in the literature. Finally, Chapter 7 concludes the thesis, summarizing the contributions on both the DCLC multicast routing problem and the multi-objective multicast routing problem, and presenting the directions for future research.

CHAPTER 2. The Optimisation Technique: Metaheuristics

2.1 Introduction

The emergence of metaheuristics for solving complex optimisation problems is an important achievement of the past 25 years in the area of operational research. This chapter firstly introduces some main concepts and definitions related to metaheuristics. Next, the principles of some most popular metaheuristic methods in the literature, including simulated annealing (SA), tabu search (TS), variable neighbourhood search (VNS), greedy randomized adaptive search procedure (GRASP), genetic algorithm (GA), scatter search (SS), particle swarm intelligence (PSO), path relinking (PR), and Ant Conoly Optimisation (ACO) are presented. Finally, the chapter briefly reviews the applications of metaheuristics in both the combinatorial optimisation and the multi-objective optimisation problems.

2.2 Basic Concepts and Definitions

Due to the effectiveness and efficiency of metaheuristics to deliver near-optimal solutions for complex optimisation problems such as the combinatorial optimisation problems and the multi-objective optimisation problems, the study and development of metaheuristics have become an important research area in operational research. The following subsections will go through a number of fundamental concepts and definitions used throughout the thesis.

2.2.1 Combinatorial Optimisation

Combinatorial optimisation problems (Papadimitriou and Steiglitz, 1982; Cook *et al*., 1998) refer to a set of problems with discrete variables to be optimised that include many different types of problems, such as the travelling salesman problem, the quadratic assignment problem, timetabling, scheduling, planning, resource allocation, decision making and routing.

A Combinatorial Optimisation Problem $P=(S, f)$ can be defined as (Blum and Roli, 2003):

• *a set of variables $X=\{ x_1,\dots, x_n \}$;*

• variable domains $D_1,\ldots, D_n$;

• constraints among variables;

• an objective function f, where $f: D_1 \times \ldots \times D_n \rightarrow \Re$;

The set of all possible feasible solutions is

$$S = \{s = \{(x_1, v_1), \ldots, (x_n, v_n)\} \mid v_i \in D_i, \text{ s satisfies all the constraints}\}.$$

S is usually called a search (or solution) space which includes all the candidate solutions for the problem. Assuming a minimisation problem is considered, to solve the combinatorial optimisation problem, one has to find a solution $s^* \in S$ with minimum objective function value, i.e. $f(s^*) \leq f(s)$, $\forall s \in S$. s^* is called the globally optimal solution of problem $P=(S, f)$.

In complexity theory, problems are usually classified into two types: P and NP. The problem class P (standing for polynomial) is the set of problems that can be solved by a deterministic algorithm in polynomial time, while the problem is NP (standing for nondeterministic polynomial) means that the problem can be solved in polynomial time by a nondeterministic algorithm. Assuming that $P \neq NP$, NP-complete (Garey and Johnson, 1979) problems are considered to be the hardest problems in NP because finding a polynomial time algorithm for the problems in NP-complete is unlikely. For combinatorial optimisation problems that are NP-complete with no polynomial time algorithm exists, solving them is difficult because it requires exponential computation time. Because of this, exact methods (e.g. linear programming, dynamic programming, branch and bound and Lagrangian relaxation) can be computationally expensive and impractical especially for problems with a large solution space. In such circumstances, approximation approaches, such as heuristics and metaheuristics, are able to deliver near-optimal solutions in a significantly reduced amount of time for combinatorial optimisation problems, which makes them extremely important methods in the area of operational research.

2.2.2 Multi-objective Optimisation

Multi-objective optimisation has become an important and challenging research topic due to the requirement of simultaneous optimisation of several conflicting objectives in many real world optimisation problems. It consists of finding a set of alternative solutions where multiple objectives are optimised at the same time while satisfying all the constraints (Ulungu and Teghem, 1994; Deb, 2005). A general multi-objective optimisation problem with *m* objectives and *r* restrictions can be defined as follows:

$$\text{Optimise } F(x) = (f_1(x), f_2(x), \ldots, f_m(x)) \qquad (2.1)$$
$$\text{s.t. } e(x) = (e_1(x), e_2(x), \ldots, e_r(x))$$

where $x \in X$ is a vector of decision variables (a solution), *X* denotes the decision space of the set of feasible regions of the solution space; $F(x)$ is the image of *x* in the *m*-objective space given by the vector of *m*-objective functions $f_i(x)$, $i \in \{1, \ldots, m\}$. The set of restrictions $e(x)$ determines the set of feasible solutions. In general, there is no unique optimal solution but a set of solutions, none of which can be seen as superior to the others when all the objectives are taken into account. If *X* consists of a discrete set of solutions, then the problem defined in Eq. (2.1) is called a multi-objective combinatorial optimisation problem.

Without loss of generality, assuming in a minimisation context, for any two decision vectors $u, v \in X$, *u* is said to dominate *v*, denoted by $u \succ v$, iff $f_i(u) \leq f_i(v)$ for all $i \in \{1, \ldots, m\}$ and there exists at least one objective $j \in \{1, \ldots, m\}$ satisfying $f_j(u) < f_j(v)$. A solution x^* is said to be Pareto-optimal if no solution in *X* dominates x^*. The set of all Pareto-optimal solutions in *X* is called the Pareto-optimal set. Correspondingly, the objective vectors of all solutions in the Pareto-optimal set in *X* is called the Pareto-optimal front (*PF*).

2.2.3 Heuristics

As defined by Reeves in (Reeves, 1996),

"*A heuristic technique (or simple heuristic) is a method which seeks good (i.e. near optimal) solutions at a reasonable computation cost without being able to guarantee optimality, and possibly not feasibility. Unfortunately, it may not even be possible to state how close to optimality a particular heuristic solution is* ".

A heuristic can therefore be used to find good-quality solutions to optimisation problems quickly, without necessarily providing any guarantee of solution optimality. Heuristics are often specific to the problem, so a heuristic which works for one problem might not be used to solve another. One simple example is hill climbing, which is a basic local search heuristic. Local search heuristic explores the search space, i.e. the set of all solutions for a problem, by defining a neighbourhood of the current solution. For a search space S, a set of neighbours of the current solution $s \in S$ is given by a function $N(s)$, where N represents a neighbourhood structure, and $N(s) \subseteq S$ is called the neighbourhood of s. A hill climbing method starts from an initial solution and keeps moving to a neighbour solution if and only if the neighbour solution is better than the current solution with respect to the objective function, until a stopping criterion is met. The drawback of hill climbing is that it is easy to get trapped in a local optimum. A local optimum with respect to a neighbourhood structure N for a minimisation problem is a solution $\hat{s}$ such that $\forall s \in N(\hat{s}): f(\hat{s}) \leq f(s)$, where f is the objective function of the problem.

2.2.4 Metaheuristics

To prevent simple heuristics from getting trapped in local optima, a new kind of advanced heuristic approaches by combining basic heuristics in higher level frameworks has emerged in the area of operational research. These approaches are nowadays called metaheuristics. The term *metaheuristic* was firstly coined by Glover (1986), which combines two Greek words. The prefix *meta* means "beyond, in a higher level", the meaning of *heuristic* is "to find", derived from the verb *heuriskein*. There are different definitions of metaheuristic in the literature, for example:

"*A metaheuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solution.*"

(Osman and Kelly, 1996)

"*A metaheuristic is an iterative master process that guides and modifies the operations of subordinate heuristics to efficiently produce high-quality solutions. It may manipulate a complete (or incompelete) single solution or a collection of solutions at each iteration. The subordinate heuristics may be high (or low) level procedures, or a simple local search, or just a construction method* ." (Voß *et al*., 1999)

Metaheuristics represent a family of generic search strategies that can be adapted for solving various combinatorial optimisation problems (Martins *et al*., 2004). As a general high-level framework that specifies simple heuristics, a metaheuristic provides a method to tackle different types of complex problems with relatively few modifications. Metaheuristics overcome the shortcoming of heuristics that are normally used in optimisation as a means of solving specific problems.

2.3 Overview of Metaheuristics

Despite the recent advances in the performance of mathematical programming algorithms, however, exact methods can solve only small instances in acceptable computational time for many complex optimisation problems. Approximate approaches based on metaheuristics are effective and flexible for delivering near-optimal solutions to complex problems, thus metaheuristics become important tools in the area of operational research. The last two decades have witnessed numerous applications of metaheuristics for both combinatorial optimisation problems and multi-objective optimisation problems, a brief overview of which is presented in the following subsections.

2.3.1 Metaheuristics in Combinatorial Optimisation

Metaheuristics have been intensively investigated and widely applied to many real world applications, including scheduling, production planning, resource assignment, supply chain management, decision support systems and bio informatics (Osman and Kelly, 1996; Glover and Kochenberger, 2003; Gendreau and Potvin, 2005; Ribeiro *et al.*, 2007; and Blum *et al.*, 2008).

Blum and Roli present an overview of metaheuristics for the applications to combinatorial optimisation problems (Blum and Roli, 2003). According to their classifications, metaheuristics can be classified in different ways. One commonly used classification distinguishes between how many solutions that are used at a time: single solution-based metaheuristics such as hill-climbing, simulated annealing and tabu search, or population-based metaheuristics such as genetic algorithm, ant colony optimisation, scatter search and particle swarm intelligence. They outline the different components and principles that are used in some important metaheuristics and compare these metaheuristics based on two main concepts: intensification and diversification, which guide the search in the metaheuristic methods. Comparing advantages and disadvantages of different metaheuristics, they point out the importance of mixing and hybridization of metaheuristics for optimisation.

Gendreau and Potvin provide a survey of metaheursitics and related applications to combinatorial optimisation (Gendreau and Potvin, 2005). A number of well-known metaheuristics are introduced and divided into two categories: single-solution metaheuristics (including GRASP, simulated annealing, tabu search and variable neighbourhood search) and population metaheuristics (including ant colony optimisation, evolutionary algorithms and scatter search). A unifying framework with some algorithmic components is illustrated to provide a general view of metaheuristics. A review of applications of metaheuristics is also presented for vehicle routing and scheduling problems. Finally, some most recent developments in the field of metaheuristics are discussed.

Ribeiro *et al*. provide a valuable summary of metaheuristics for optimisation problem in computer communications, telecommunications, network design and routing (Ribeiro *et al*., 2007). The summary reviews the principles associated with some of the main metaheuristics (e.g. simulated annealing, tabu search, GRASP, variable neighbourhood search, genetic algorithms, particle swarm optimisation and path relinking), and gives the templates for basic implementations of these metaheuristics.

Blum *et al*. present an overview of the recent developments of hybrid metaheuristics for optimisation problems (Blum *et al*., 2008). Some examples of the combinations of metaheuristics with other classic optimisation techniques, such as linear programming, dynamic programming or branch and bound, for different complex combinatorial optimisation problems are discussed in the book. These hybrid metaheuristic approaches demonstrate that they can provide more efficient and higher quality solutions for combinatorial optimisation problems because the hybridization can combine the advantages of different optimisation techniques and thus complement the strengths of one another.

More information about metaheuristics is detailed by Glover and Kochenberger (2003) and Resende and de Sousa (2004). In the following subsections, some popular metaheuristic methods which have been widely used in many applications of combinatorial optimisation problems will be outlined in turn.

2.3.1.1 Single Solution based Metaheuristics

1) Simulated Annealing (SA)

The concept of simulated annealing in combinatorial optimisation was introduced by Kirkpatrick *et al.* (1983), based on an analogy between the physical annealing process of solids and the problem of solving combinatorial optimisation problems. Annealing is known as a thermal process, where a solid is melted by increasing its temperature and then followed by a progressive temperature decrease aimed at recovering a solid state of lower energy. For a combinatorial optimisation problem, a solution corresponds to a state of the physical system and the solution cost to the energy of a state.

The procedure of SA metaheuristic is guided by the cooling schedule which starts from a high temperature, i.e. the initial temperature, and repeatedly reduces the temperature until the temperature arrives at the final temperature or a certain stopping criterion is satisfied. In each iteration, a neighbourhood solution is obtained by randomly choosing a move of the current solution. If the neighbouring solution is better, it is accepted automatically. Otherwise, the non-improving neighbouring solution is accepted with a probability given by the Metropolis criterion, where the probability of acceptance depends on the current temperature and the difference between the fitness of the neighbouring solution and the current solution. SA thus has the ability in this process of escaping from local optima by visiting worse neighbours, which makes it a very effective metaheuristic when exploring the search space of complex combinatorial optimisation problems.

Owing to its simplicity of implementation and effectiveness in many problems, in practice, SA has been applied to solve a variety of combinatorial optimisation problems, including graph colouring, route planning, scheduling and timetabling (Thompson and Dowsland, 1998; Liu, 1999; and Bouleimen and Lecocq, 2003). More detailed implementations and developments of SA are discussed in Aarts and Ten Eikelder (2002); Hederson *et al.* (2003) and Suman and Kumar (2006). The disadvantage of SA is that its convergence speed and quality of solutions rely on the choices of several parameters such as the initial temperature, the temperature decrement and the stopping criterion in the cooling schedule, which need to be tuned for different problems.

2) Tabu Search (TS)

Tabu search was proposed by Glover (1986). The principle of TS is to make use of a short-term memory, called the tabu list, to drive the search by escaping from local optima and avoiding cycling (Glover, 1997). The tabu list stores the forbidden neighbours which are a set of previously visited solutions (or attributes of recently visited solutions), so as to prevent the search from trapping in local optima. Different from hill climbing or other simple local search procedures, TS is deterministic where, at each iteration, the best non-forbidden neighbourhood solution of the current solution is selected, even if it leads to a worse solution with respect to the objective function. TS is thus able to escape from the local optima. The search procedure stops after a fixed number of iterations or a maximum number of continuous iterations without improvements to the best known solution.

Besides the short-term memory, some long-term memories have also been used for the purpose of providing intensification or diversification in the algorithm. For example, a frequency memory can be used to count how often certain attributes are found in previously visited solutions. Neighbourhood solutions which contain attributes with high frequency counts can then be penalized to allow the search to visit other regions of the search space. The diversification of search is then realised.

TS is effective in providing solutions very close to the optimal solution. It has been widely applied to various complex combinatorial optimisation problems in many domains (Skorin-Kapov, 1990; Gendreau *et al*., 1994; Glover and Laguna, 1997; and Gaspero and Schaerf, 2001). Detailed descriptions and recent developments of TS can be found in Glover and Laguna (2002), in Gendreau (2002) and in Gaspero and Schaerf (2007). However, in order to achieve good performance, the implementations of TS algorithms often need to appropriately tune parameters such as the size of tabu list and the stopping criterion.

3) Variable Neighbourhood Search (VNS)

Variable neighbourhood search, jointly invented in the mid 1990s by Mladenović and Hansen (1997), is an efficient metaheuristic for solving combinatorial optimisation problems. The basic idea is to systematically change the employment of different neighbourhoods within a local search. This makes the search more flexible to explore the solution space of the problem, and potentially leads to better solutions which are difficult to obtain by using standard single neighbourhood based local search algorithms (Jari *et al*., 2007; Burke *et al*., 2008). A basic VNS works as follows. An initial solution is generated as the current solution, and then the algorithm exploits a set of predefined neighbourhoods around the current solution in the search space. A solution is selected at random in the first neighbourhood of the current solution. If its fitness value is better than the current solution, the algorithm moves to that solution and continue the search with the first neighbourhood. Otherwise, the search proceeds to the next neighbourhood. The search stops when a stopping condition is satisfied. A widely used variant of VNS is the deterministic variable neighbourhood descent (VND) search where the best neighbouring solution, instead of a random one, of the current solution is selected in each neighbourhood structure, and no local search method is applied after each neighbourhood.

The basic principles of VNS are easy to apply and reply on few parameters: the stopping criterion and the number of neighbourhoods. Recently, some hybrids of VNS with other metaheuristics, such as TS or GRASP, have been reported (Martins *et al*., 2000; Hansen and Mladenović, 2003). Detailed description and applications of VNS can be found in Hansen and Mladenović (2003).

4) Greedy Randomized Adaptive Search Procedure (GRASP)

Greedy Randomized Adaptive Search Procedure (Feo and Resende, 1995) is a multi-start metaheuristic used in a wide variety of combinatorial optimisation problems. Each GRASP iteration consists of two phases: a construction phase which creates a feasible initial solution and a local search phase which explores the neighbourhood of the initial solution. The construction phase builds the feasible solution in a greedy randomized manner by iteratively creating a candidate list of elements, called the restricted candidate list (RCL). The quality of each element in RCL is evaluated by a certain greedy function. The size of RCL is limited either by the number of the elements or by the quality of the elements with respect to the best candidate element. At each step, an element is randomly selected from RCL and added in the unfinished solution until a feasible solution is obtained. Through the randomized manner, the best current element is not necessarily chosen when constructing the initial solution; the construction phase in GRASP thus makes the search diversified. After the construction phase, a local search phase is applied to improve the current solution with better neighbouring solutions until a local optimum is found, which intensifies the search. After applying a given number of iterations, the best overall solution is output as the final result. GRASP is easy to implement with few parameters needing to be set and tuned, such as the maximum number of iterations and the size of RCL in the construction phase.

GRASP is a very effective metaheuristic and has been applied to a wide range of combinatorial optimisation problems including the vehicle routing problem (Kontoravdis and Bard, 1995); the graph planarization problem (Resende and Ribeiro, 1997); the traffic assignment (Prais and Ribeiro, 2000); the job shop scheduling problem (Binato *et al*., 2001); and the minimum spanning tree problem (Souza *et al*., 2003). Recently, some advanced GRASP algorithms hybridised with other methods have been proposed.

For example, in Höller *et al*. (2008), the pilot method is incorporated in a VNS and a GRASP approach for the design of SDH/WDM networks. The pilot method is a tempered greedy method by performing repetition, which intends to record the best result before arriving at a promising solution. It may be seen as an intelligent technique to look ahead to possible choices after certain iterations have been applied to support the final decision of respective options; applications of pilot methods can be found in Bertsekas *et al*. (1997); Duin and Voß (1999); and Höller *et al*. (2008). In the two hybrid algorithms, one pilot method is applied to decide the ordering of the neighbourhood structures explored in VNS; another pilot method is invoked subsequently to the standard GRASP to find better solutions by redirecting transit traffic in the networks. Both hybridised algorithms combined with the pilot methods have shown to obtain better solutions, while the computational time has been considerably increased. See Resende and Ribeiro (2005); Festa and Resende (2008); and Festa and Resende (2009) for extensive surveys of recent advances and applications of GRASP metaheuristics.

2.3.1.2 Population based Metaheuristics

1) Genetic Algorithm (GA)

Genetic algorithm was first proposed by Fraser (1957). It is a population-based evolutionary algorithm motivated by the evolution and adaptation of species to the environment, based on the Darwinian principle of natural selection. A population of solutions evolves from one generation to the next through a series of operators, i.e. selection, crossover and mutation, to generate new solutions. A solution in the population needs to be encoded as a chromosome (e.g. a bit or integer string). An objective function is defined to evaluate the quality of each solution. The selection method chooses good quality solutions in the population as the parents. The idea behind selection methods is to prefer better solutions to worse ones, and many selection methods have been used to accomplish this idea, such as roulette-wheel selection, tournament selection and ranking selection, etc. The crossover operator takes the selected parents and combines them to generate one or two offspring solutions. The role of crossover is to inherit some desirable features from parents to the next generation. Before updating the old population, the mutation operation is applied to each offspring by some random perturbations with certain probability in order to introduce some new traits which are not presented in the current parent solutions. This evolutional process is repeated for a given number of generations, and the best solution found is returned as the final solution. The distinctive feature of genetic algorithms is that it can explore a population of solutions and create new offspring solutions through the combination of good attributes of two parent solutions. A good introduction about genetic algorithms can be found in Holland (1975).

GAs have been successfully applied to many combinatorial optimisation problems (Holland, 1992; Reeves, 2003; and Buriol *et al*., 2005). Based on the standard framework outlined above, hybridization can be an extremely effective way to improving the performance and effectiveness of GAs. One common form of hybridization is to apply a local search operator as an intensification strategy in GA to improve the offspring solution until a local optimum is reached. The combination of GA and local search is known as the memetic algorithm. A good introduction to memetic algorithms can be found in Moscato and Cotta (2003). Refer to Hart *et al.* (2003); Krasnogor *et al.* (2004); and Krasnogor and Smith (2005) for recent advances in the theory and applications of memetic algorithms.

2) Scatter Search (SS)

Scatter search (SS) (Glover, 1998; Glover *et al*., 2000; and Glover *et al*., 2003) is a population-based metaheuristic. The basic procedure of SS starts with producing a large set of diverse solutions by a diversification generation method. An initial reference set of solutions is then selected from the diverse solution set, where all solutions in the reference set are ordered according to the quality. Scatter search repeatedly operates on a subset of the reference set to generate new solutions by combining elements of solutions selected from the subset using a solution combination method. The aim of scatter search is to derive new solutions from combined solutions in order to yield better solutions. An improvement method may be applied upon the new solution to further improve its quality. If the newly-created solution is better than the worst solution currently in the reference set, the new solution replaces the worst solution and the reference set is reordered. The reference set is thus updated to constitute elite solutions obtained from the previous search. This procedure is repeated until no better solutions can be generated from the reference set anymore.

The basic design of SS is very flexible, since each component in SS can be implemented in different ways. Recent research has shown that SS is effective in yielding promising outcomes for solving a wide variety of combinatorial optimisation problems. Examples include quadratic assignment (Cung *et al*., 1997), graph coloring (Hamiez and Hao, 2002), arc routing (Greistorfer, 2003), and nurse rostering (Maenhout and Vanhoucke, 2006; and Burke *et al*., 2010). More detailed descriptions and different improvements of SS metaheuristic can be found in Laguna (2002); Greistorfer (2003); and Glover *et al*., (2006).

3) Path Relinking (PR)

Path relinking was originally proposed by Glover (1996). It is an evolutionary method which generates new solutions by exploring trajectories that connect elite solutions in the search space. A typical path relinking procedure starts from creating a reference set of diverse initial solutions. At each step, a pair of solutions (the initiating solution and guiding solution) is chosen from the reference set. Starting from the initiating solution, a path relinking process generates a path in the neighbourhood space leading toward the guiding solution, and better solutions may be obtained by exploring the path connecting the two solutions. A neighbourhood structure is applied to generate moves, i.e. neighbourhood solutions, along the path. The main goal of path relinking is to introduce attributes that are presented in the elite guiding solution to the intermediate solutions so that better solutions may be generated.

Path relinking has been suggested as an approach to integrate intensification and diversification strategies (Glover *et al*., 2000). In the literature, most recent implementations of path relinking have been considered and proposed in conjunction with many other metaheuristic algorithms, such as GRASP, SS, or TS (Laguna and Martí, 1999; Glover et al., 2006; Bastos and Ribeiro, 2001; and Resende and Ribeiro, 2005).

4) Particle Swarm Optimisation (PSO)

Particle swarm optimisation proposed by Kennedy and Eberhart (1995) is a bio-inspired population-based metaheuristic based on swarm intelligence. It is inspired by the behaviour of swarms of fishes or flocks of birds to find a good food place. A PSO algorithm maintains a population of particles (the swarm), where each particle represents a location (solution) in a search space. The particles start at random locations and search for the minimum (or maximum) which are evaluated according to a given objective function by moving in a multi-dimensional continuous space (search space). The movements of a particle depend on its velocity and the good locations have been found by the particle itself or other particles in the swarm. In such a way, particles can interact by sharing information about their positions both locally and globally. The coordination of movements of the particles in the swarm is the appealing aspect of PSO.

The original PSO was designed for continuous optimisation problems. A variant called Discrete Particle Swarm Optimisation (DPSO) has been designed by Kennedy and Eberhart (1997). In DPSO, particles no longer move in a multi-dimensional continuous space, since a discrete exploration of search space is more appropriate for solving combinatorial optimisation problems. For example, a recent DPSO algorithm named Jumping Particle Swarm Optimisation (JPSO) was introduced by Moreno-Pérez *et al.* (2007) to solve combinatorial optimisation problems. Later, it was used by Consoli *et al.* (2008) to tackle the minimum labelling Steiner tree problem. This approach has also been recently applied to deal with the vehicle routing problem with time windows (Castro-Gutiérrez *et al.*, 2009).

PSO is relatively easy to understand and to implement. A number of variants exist in the literature and have been highly successful in dealing with a range of combinatorial optimisation problems, such as job shop scheduling (Sha and Hsu, 2006), flow shop scheduling problems (Lian *et al.*, 2006; Tasgetiren, 2007), resource constraint project scheduling (Zhang *et al.*, 2006), the Travelling Salesman Problem (TSP) (Onwubolu and Clerc, 2004), the Steiner tree problem (Consoli *et al.*, 2008), and the vehicle routing problem with time windows (Castro-Gutiérrez *et al.*, 2009). More information about developments of PSO can be found in Kennedy and Eberhart (1995) and at http://www.swarmintelligence.org/.

5) Ant Colony Optimisation (ACO)

Ant Colony Optimisation is a metaheuristic proposed by Dorigo (1992) which simulates the behaviour of a natural ant colony seeking the shortest path from the nest to a food source in the real world. ACO uses a colony of artificial ants (agents) that work cooperatively and that communicate through a pheromone trail. An ant chooses the shortest path to the food source based on the level of pheromone on each available path. After an ant finishes its tour, it leaves a certain amount of pheromone on the path it travels according to the distance of that tour. The principle is the shorter the path, the more pheromone is left along the path. Thus, the more ants pass along a certain path, the higher the level of pheromone is deposited on the path, and so other ants are more likely to follow this path. ACO is an iterative process where the pheromone trail is transferred from one iteration to the next. To prevent the pheromone information from influencing the later iterations for too long, some amount of pheromone evaporates during subsequent updates of the pheromone values. The idea of ACO is to allow the ants to find good solutions for a given optimisation problem by making a sequence of decisions based on a probability proportional to the pheromone trail.

In recent years, ACO has been successfully applied to different types of combinatorial optimisation problems, including the travelling salesman problem (Stützle and Dorigo, 1999), the vehicle routing problem (Gambardella *et al*., 1999), the sequential ordering problem (Gambardella and Dorigo, 2000), the resource constrained project scheduling problem (Merkle *et al*., 2002), and the communication network routing problem (Pinto and Baran, 2005; Huang *et al*., 2007). For more detailed descriptions of ACO, refer to Dorigo and Di Caro (1999); Maniezzo and Carbonaro (2002); and Dorigo and Stützle (2003).

2.3.2 Metaheuristics in Multi-objective Optimisation

Many real world applications, such as problems in engineering, finance, and logistics, are recognized to be multi-objective. There does not exist a single optimal solution but rather a set of solutions which are the best trade-offs among the objectives, known as the Pareto optimal solutions, for decision-makers to choose. Over the past two decades, metaheuristics have gained increasing popularity in multi-objective optimisation due to the flexibility for implementation and the ability to find multiple Pareto-optimal solutions in a single run. Thus many metaheuristic algorithms, including GA, SA, TS, PSO and ACO, have been investigated to solve different multi-objective optimisation problems (Ulungu and Teghem, 1994; Ehrgott and Gandibleux, 2000; Jones *et al*., 2002; Landa-Silva *et al*., 2004; Gandibleux *et al*., 2004; Konak *et al*., 2006; Li *et al*., 2007; Li and Landa-Silva, 2008; Liu *et al*., 2008; Wei *et al*., 2009; Balram *et al*., 2010; Liu *et al*., 2010; and Martins and Costa, 2010). The majority of research studies in this area have focused on generating a set of non-dominated Pareto optimal solutions for these complex multi-objective optimisation problems.

2.3.2.1 Multi-objective GA (MOGA)

Due to the population-based nature and the ability to simultaneously search different regions of a solution space, genetic algorithms (GAs) have become the most popular metaheuristic for solving multi-objective optimisation problems in the literature. The multi-objective GA extends the generic single-objective GA by finding a set of non-dominated solutions. Since the first multi-objective GA was proposed by Schaffer (1985), variations of multi-objective genetic algorithm have been developed, including Niched Pareto

Genetic Algorithm (NPGA) (Horn *et al.*, 1994); Non-dominated Sorting Genetic Algorithm (NSGA) (Srinivas and Deb, 1994); Multi-objective Genetic Algorithm (MOGA) (Murata and Ishibuchi, 1995); Strength Pareto Evolutionary Algorithm (SPEA) (Zitzler and Thiele, 1999); Pareto Evolutionary-based Selection Algorithm (PESA) (Corne and Knowles, 2000); Dynamic Multi-objective Evolutionary Algorithm (DMOEA) (Yen and Lu, 2003); and MultiObjective Evolutionary Algorithm based on Decomposition (MOEA/D) (Zhang and Li, 2007). These multi-objective GAs differ in their fitness assignment approaches, in diversification strategies, and in maintaining elitism.

Different approaches of assigning fitness value for solutions in the population have been investigated. A classical approach is the weighted sum approach, where a weight is assigned to each objective function so that the multi-objective problem is converted to a single objective problem with a scalar objective function. For the multi-objective problem defined in Section 2.2.2, the weighted sum of a linear combination of different objectives called the weighted linear utility function can be defined as follows:

$$\text{Minimise } g^{(ws)}(x, \lambda) = \lambda_1 f_1(x) + \lambda_2 f_2(x) + \cdots + \lambda_m f_m(x) \tag{2.2}$$

where $f_i(x)$, $i \in \{1, \ldots, m\}$ represents the m-objective functions, λ is a weight vector, $\sum_{i=1}^{m} \lambda_i = 1$. Each $\lambda_i \in [0,1]$ is associated with one objective function $f_i(x)$.

Function (2.2) is also called the weighted scalarising function, which is one of the mostly used scalarising functions in multi-objective algorithms in the literature (see more details of other scalarising functions in Miettinen, 1999).

In MOGA (Murata and Ishibuchi, 1995), a weighted sum of multiple objective functions is used to evaluate solution fitness by generating a random normalized weight vector for each solution during the search phase at each generation. Another commonly used approach is the Pareto-ranking which utilises the Pareto dominance to evaluate fitness for solutions. Goldberg (1989) firstly presents the idea to rank the individual by its degree of non-domination in the population. This has been widely used within GA for multi-objective optimisation. Individuals that are non-dominated with respect to all other individuals in the population are ranked 1. For the unranked individuals that are non-dominated with respect to the remaining individuals are assigned the rank 2, and so on, until all individuals have been ranked. Other Pareto-ranking approaches include non-domination sorting (Srinivas and Deb, 1994; Deb *et al.*, 2000) and strength Pareto ranking (Zitzler and Thiele, 1999).

In order to obtain a set of diverse and uniformly distributed Pareto optimal solutions in the objective space, i.e. a good approximation to the true Pareto front, several diverse strategies have been developed, such as fitness sharing (Horn *et al*., 1994; Murata and Ishibuchi, 1995; and Miller and Shaw, 1996), which encourage the search in unexplored regions of a Pareto front, and crowding distance (Deb *et al*., 2000) which aims at diverse solutions along the Pareto front by measuring population density around a solution.

The maintenance of elitist solutions for multi-objective GA is also important. Multi-objective GA uses two principal methods to implement elitism: maintaining elitist solutions in the population or storing elitist solutions in an external population (Jensen, 2003). For example, Zitzler and Thiele (1999) maintain an external population which stores non-dominated solutions that found during the search. Sarker *et al*. (2002) propose a multi-objective GA with a dynamic population size and a pure elitist strategy where only non-dominated solutions participate to generate the next generation.

Several valuable surveys on multi-objective GA can be found in the literature. A recent overview and tutorial of GA for multi-objective optimisation, focusing on the issues and components of implementing multi-objective GA, has been presented by Konak *et al*. (2006). For further discussions on state-of-the-art multi-objective GA algorithms, please refer to surveys by Deb (2001); Coello *et al*. (2002); Konak *et al*. (2006); Li *et al*. (2007); Wei *et al*. (2009); and Liu *et al*. (2010).

2.3.2.2 Multi-objective SA (MOSA)

Multi-objective simulated annealing algorithms have been investigated by researchers for solving a range of multi-objective optimisation problems (Serafini, 1992; Czyzak and Jaszkiewicz, 1998; Ulungu *et al*., 1999; Haidine and Lehnert, 2008; Li and Landa-Silva, 2008; and Martins and Costa, 2010). The main concern of adapting simulated annealing to multi-objective optimisation problems is the acceptance probability of neighbouring solutions. A good survey of multi-objective simulated annealing algorithms for solving both single and multi-objective optimisation problems and their performance comparisons is presented by Suman and Kumar (2006).

The first simulated annealing algorithm for multi-objective problems was proposed by Sefarini (1992). It is a single-solution based simulated annealing. At each step, the algorithm optimises the weighted scalarizing function and uses the acceptance probability defined by the weighted scalarizing function to determine local search moves. In order to find more non-dominated solutions, the random weight vector of scalarizing function is adjusted slightly to diversify the search.

Czyzak and Jaszkiewicz (1998) present a population-based MOSA with adaptive search directions. The algorithm simultaneously optimises multiple weighted scalarizing functions. To diversify the search to the unexplored regions of the search space, it adaptively tunes the weight vector of each solution during the search according to the closeness to its neighbourhood solutions.

Ulungu *et al.* (1999) propose an effective multi-objective simulated annealing algorithm for bi-objective knapsack problems. In the MOSA algorithm, a number of predefined weight vectors determine a set of weighted linear utility functions. Each of the functions is optimised sequentially by an independent SA process. The outcome of the algorithm is a set of non-dominated solutions generated during the optimisation.

A recent Evolutionary Multi-objective Simulated Annealing (EMOSA) algorithm, developed by Li and Landa-Silva (2008), has been tested for solving benchmark TSP problems. The EMOSA algorithm combines the strengths of the SA-based local search and the evolutionary search. In the EMOSA algorithm, a competitive mechanisim between similar members in the population and a two-phase strategy for adaptively tuning the search directions, have shown to be effective in comparison with other MOSA algorithms.

2.3.2.3 Multi-objective TS (MOTS)

Although tabu search has been successfully applied to many single objective problems, it has not been fully explored for solving multi-objective optimisation problems. As pointed out by Jones *et al.* (2002), among metaheuristic algorithms for multi-objective optimisation, only about 6% algorithms apply TS, while the majority (around 70%) algorithms employ GA, and almost 24% are developed by SA. Gandibleux *et al.* (1997) present a multi-objective tabu search algorithm. In the algorithm, the current solution is repeatedly replaced by a neighbouring solution with the highest value of weighted scalarizing function. The weight vector is modified periodically so that the search direction is diversified accordingly. Two tabu lists have been applied: one is used to prevent the algorithm from returning to previously visited solutions and another is designed for changing the weight vector.

Kulturel-konak *et al*. (2006) propose a multi-objective tabu search algorithm for the series parallel system redundancy allocation problem. Instead of using a priori weighting and scaling of objectives, the proposed algorithm randomly activates an objective using a multinomial probability mass function. Experiments show that the algorithm is effective in identifying a set of diverse Pareto optimal solutions.

2.3.2.4 Hybrid Multi-objective GA

In recent multi-objective optimisation research, some hybridizations of GA have been applied to different multi-objective optimisation problems, such as GA with local search algorithms, i.e. memetic algorithms (Ishibuchi and Murata, 1998; Jaszkiewicz, 2002; and Mendoza *et al*., 2010). These memetic algorithms have been shown to be very effective for complex multi-objective optimisation problems, due to the ability of local search to find local optima effectively over a relatively small part of the search space. Ishibuchi and Murata (1998) apply a local search procedure to each offspring generated by crossover and mutation operations, where a randomly generated weight vector is used to evaluate the fitness values of neighbourhood solutions. A characteristic feature of their algorithm is that only a small number of neighbourhood solutions are examined to reduce the computation time spent by the local search. Jaszkiewicz (2002) proposes a multi-objective genetic local search (MOGLS) algorithm for the travelling salesman problem. In each generation, the algorithm draws a random weighed linear utility function and constructs a temporary population composed of a number of best solutions among the current population. A local search is then applied to each offspring of two parents randomly selected from the temporary population. Their experiments show that the proposed algorithm outperforms Ishibuchi and Murata's algorithm (1998) and a Pareto-ranking-based GA.

2.3.2.5 Other Metaheuristic Approaches

Other metaheuristics such as ACO, SS, VNS and PSO, have also been applied, although with relatively less research effort, for multi-objective optimisation problems in recent work (e.g. Doerner *et al*., 2001; Beausoleil, 2005; Liu *et al*., 2006; and Adibi and Zandieh, 2010). For example, Beausoleil (2005) developed a hybrid Multi-objective Scatter Search (MOSS) with tabu search for nonlinear multi-objective optimisation problems. A multi-start tabu search is applied as the diversification method to generate diversified solutions in MOSS. A frequency memory is used to diversify the search to less visited areas and it is shared by the sequence of tabu searches. Weighted combinations of reference solutions are designed as the combination method in the scatter search phase.

2.4 Summary

This chapter first introduces some basic concepts related to metaheuristics, and then presents a brief overview of the principles of nine popular metaheuristics which have been widely used for combinatorial optimisation problems; finally it reviews various multi-objective metaheuristics in the area of multi-objective optimisation. Due to the effectiveness and efficiency of delivering near-optimal solutions for complex problems, metaheuristics play a very important role in the area of operational research. As reviewed in this chapter, metaheuristics have been investigated within many application areas such as logistics, transportation, telecommunication networks, manufacturing and heath care.

As stated in Chapter 1, the two case studies in the thesis, i.e. the DCLC multicast routing problem and the multi-objective multicast routing problem, belong to NP-complete problems, meaning that both problems are very challenging due to the high computational expenses for finding an optimal or a set of Pareto optimal solutions. Because of this, exact methods can be computationally expensive and impractical, especially for large-scale problems. In such circumstances, metaheuristic approaches should be more appropriate choices for solving these two problems, since many metaheuristic algorithms have already been successfully applied in recent years to various combinatorial optimisation problems and multi-objective optimisation problems. The summary of these metaheuristic algorithms in this chapter aims to provide a comprehensive overview of metaheuristics which may be adapted or hybridized for the DCLC multicast routing problem and the multi-objective multicast routing problem discussed in this thesis.

CHAPTER 3. The QoS Multicast Routing Problem

3.1 Introduction

The general problem of multicast routing has received increasing attention in the areas of computer communications and operational research (Salama *et al*., 1997; Yeo *et al*., 2004; Oliveira *et al*., 2005; and Fabregat *et al*., 2005). The QoS multicast routing is a valuable technique for sending messages from source(s) to a set of destinations that belong to the same multicast group while minimising the utilisation of network resources and satisfying certain QoS requirements. With the increasing demand of numerous multimedia applications, the QoS requirements of the multicast routing in the underlying computer networks take into account several attributes such as the cost, delay, delay variation, bandwidth and hops count. In this chapter, the underlying mathematic model of multicast routing, i.e. the Steiner tree problem, is firstly introduced. Secondly, an overview of the QoS multicast routing problem is presented including the network model and network simulator of the problem, and related work for various QoS multicast routing problems. Finally, the two case studies of the QoS multicast routing problem, the DCLC multicast routing and the multi-objective multicast routing problems, are formally defined and the related work is reviewed.

3.2 Basic Model of Multicast Routing

As mentioned in Chapter 1, the unconstrained multicast routing problem which tries to construct a multicast tree with the minimum tree cost without considering any other constraints can be modelled as the MStTG problem (Hwang and Richards, 1992), a well known NP-complete problem (Garey and Johnson, 1979). When solving real time multicast routing problems, more QoS constraints need to be considered in addition to the problem of finding a Steiner tree. In this case, the QoS multicast routing problem can be modelled as the constrained Steiner tree problem. As the underlying mathematical model of multicast routing problems, the Steiner tree problem and a number of related problems have been widely studied for decades, and still present great research challenges.

3.2.1 The Steiner Tree Problem

Given a weighted graph $G = (V, E)$, where V is a set of nodes, and E is a set of edges, respectively. Each edge $e(i, j) \in E$ between nodes i and j in V is associated with a weight $W(e)$: $E \to \Re^+$, where $\Re^+$ are nonnegative real numbers. For a subset of nodes $S \subseteq V$, the Steiner tree problem is defined as constructing a tree which connects all the nodes in S using a subset of edges in E. Extra nodes in $V \backslash S$ may be added to the Steiner tree, called the Steiner nodes. Based on the above definitions, the MStTG problem can then be formulated as follows:

The Minimum Steiner Tree Problem: *the objective of the MStTG problem is to find a minimum weighted tree T in G, $T \subseteq E$, which spans all nodes in S and some of the nodes in $V \backslash S$. The weight of the tree $W(T)$ is the sum of the weights of all edges in the tree T, i.e. $W(T) = \sum_{e \in T} W(e)$.*

3.2.2 Related Work for the Steiner Tree Problem

The MStTG problem and its variants have been widely investigated as a classical optimisation problem in the literature. A large amount of exact and heuristic algorithms have been developed for solving the typical NP-complete problems (Takahashi *et al*., 1980; Ko *et al*., 1981; Kou *et al*., 1981; Duin and Voß, 1997; Koch and Martin, 1998; Duin and Voß, 1999; Barahona and Ladanyi, 2006; and Costa *et al*., 2006). In recent years, many metaheuristic algorithms, such as tabu search (Ribeiro and Souza, 2000; Bastos and Ribeiro, 2001), VNS (Canuto *et al*., 2001; Gruber and Raidl, 2005), GRASP (Martins *et al*., 2000; Ribeiro *et al*., 2002), and PSO (Zhong *et al*., 2008; Consoli *et al*., 2008), have been applied for solving variants of MStTG problems. Surveys of Steiner tree problems can be found in Hwang and Richards (1992); Voß (1992); Duin and Voß (1997); Koch and Martin (1998); Zachariasen (1999); and Costa *et al*. (2006).

KMB (Kou-Markowsky-Berman) (Kou *et al*., 1981) is a representative heuristic for the Steiner tree problem, where the Prim's minimum spanning tree algorithm (Prim, 1957) is applied to construct the unconstrained Steiner tree. In the Prim's algorithm, for a connected weighted undirected network, the tree is repeatedly constructed from an arbitrary node until it spans all the nodes in the network, where the total weight of all edges in the tree is minimised. Koch and Martin (1998) use a branch and cut method with pre-processing, reduction techniques and primal heuristics to solve the Steiner tree problem. Such techniques have also been widely studied on other variants of the Steiner tree problems (Barahona and Ladanyi, 2006; Costa *et al.*, 2006).

Duin and Voß (1999) have successfully applied pilot methods with heuristic repetition to the MStTG problem. Four Steiner tree heuristics from the literature have been used as the sub-heuristics in the formulated pilot methods. Experiments show that a pilot method always delivers good solutions when using a more accurate sub-heuristic. Martins *et al*. (2000) and Ribeiro *et al*. (2002) have applied GRASP to solve the MStTG problem. Martins *et al*. (2000) describe a hybrid GRASP heuristic with two local search strategies for the MStTG problem. The proposed algorithms are tested on a group of parallel processors. Computational results show that their GRASP heuristic has high possibilities of finding the optimal solutions. Ribeiro *et al*. (2002) present a hybrid GRASP with weight perturbations and two adaptive path-relinking heuristics on a set of elite solutions for the MStTG problem. One is the path relinking with complementary move; the other is the path relinking with weight penalization. Experiment results on a broad set of benchmark problems illustrate the effectiveness and the robustness of their GRASP algorithm. Both GRASP algorithms are restricted to deal with Steiner tree problems with no constraints. Consoli *et al.* (2008) propose a novel jumping particle swarm optimisation algorithm to tackle the minimum labelling Steiner tree problem.

3.3 Overview of QoS Multicast Routing Problems

The rapid growth of group communications and multimedia applications over the Internet has accelerated the need for QoS multicast routing support in computer communication networks. The introduction of applications demanding QoS makes the multicast problem more challenging due to the varied QoS requirements for real time multicast applications. These QoS requirements can be classified into link constraints (e.g. bandwidth), path constraints (e.g. end-to-end delay or path cost), and tree constraints (e.g. delay variation). QoS multicast routing problems are NP-complete as they can be reduced to the constrained Steiner tree problems, a classical NP-complete problem. The nature of the QoS multicast routing problem and the variety of requirements that exist in the real time applications have attracted increasing research attention from the areas of both computer communications and operational research on developing various optimisation and search algorithms, including exact methods, heuristics and metaheuristics, for variants of QoS multicast routing problems (Noronha and Tobagi, 1994; Diot *et al*., 1997; Salama *et al*., 1997; Chen *et al*., 1998; Pasquale *et al*., 1998; Wang and Hou, 2000; Sahasrabuddhe *et al*., 2000; Youssef *et al*., 2001; Striegel and Manimaran, 2002; Paul and Raghavan, 2002b; Yeo *et al*., 2004; Oliveira *et al*., 2005; and Fabregat *et al*., 2005).

An early survey of problems, protocols and mechanisms related to multicast communication has been carried out by Diot *et al*. (1997). After analyzing the issues and challenges of multicast routing, the mechanisms in different levels including node-level, hop-by-bop and end-to-end for providing multicast transmission are discussed. The paper also presents some existing multicast routing applications and algorithms.

A detailed simulation comparison of some of the most important multicast routing algorithms is made by Salama *et al*. (1997). The comparison is based on the quality of the generated multicast trees and the efficiency in managing the network resources of these algorithms. Simulation results show that most of the early heuristic algorithms for tackling the multicast routing problems are deterministic, and these algorithms either need long computational time or generate low quality solutions. Algorithms which are more scalable and efficient need to be developed.

Pasquale *et al*. (1998) present an overview of research in multimedia multicasting. The authors discuss the issues arising from the multicasting in computer networks to support real time multimedia applications, including quality of service, resource reservations, routing, error and traffic control, and heterogeneity.

Wang and Hou (2000) provide a comprehensive overview of existing multicast routing algorithms, protocols, and the issues and challenges of providing QoS support to multicast routing protocols. In the paper, multicast routing problems are classified according to their objective functions and QoS constraints. Based on the classification, some multicast routing algorithms have been described and their advantages and disadvantages have been discussed. The authors categorize existing multicast routing protocols into two types: the source-based and core-based protocols. In addition, issues related to multicast routing protocols, such as the collection and updating of state information, the construction of multicast trees, state and tree maintenance, and scalability in large networks, have been analysed. Furthermore, the challenges in providing QoS in multicast routing have also been outlined and some of the approaches to solutions have been illustrated.

Almost at the same time, Sahasrabuddhe *et al*. (2000) provide a tutorial about two important topics in multicasting, including multicast routing algorithms and protocols employed on the Internet. In this tutorial, some of the important issues in multicasting, including the classification of multicast routing algorithms, different QoS properties, dynamic multicast groups, scalability, survivability and fairness within multicast routing are illustrated. Several existing multicast routing protocols are then discussed in detail.

Striegel and Manimaran (2002) present a survey of QoS multicasting issues. In the article a multicast "life cycle" model is defined, which consists of four steps: multicast group (session) creation, multicast tree construction with resource reservation, data transmission and multicast session teardown. Then some important issues involved in a typical multicast session, such as multicast group dynamics, network dynamics and traffic dynamics, are discussed. To provide QoS multicasting support and to solve various issues that have occurred during the multicast session, several multicast routing protocols have been proposed on the Internet. The authors introduce and analyse these protocols.

Paul and Raghavan (2002b) review a number of multicast routing algorithms and protocols. The article introduces some multicast routing algorithms according to the different types of multicast trees, such as source tree, shared tree and Steiner tree, generated by these algorithms. The features of some existing multicast routing protocols are analysed. Some of the issues relating to the implementation and deployment of multicast on the Internet are discussed.

A recent overview on applications and combinatorial optimisation problems in the area of multicast routing is presented by Oliveria and Pardalos (2005). The paper discusses different combinatorial problems associated with multicast routing in terms of varied constraints, such as the Steiner tree in graphs, Steiner tree with delay constraints, dynamic multicast routing and the degree-constrained Steiner tree problem, and then compares the related algorithms for each type of the main problems.

At the early stage of research, the multicast routing problems were defined and solved mainly as a single-objective optimisation problem subject to some QoS constraints. When a range of inter-dependent and conflicting QoS objectives and constraints are considered simultaneously, the QoS multicast routing problems can be more appropriately defined as multi-objective optimisation problems. With the recent advances in multi-objective research, multi-objective optimisation algorithms have been investigated for multicast routing problems with more realistic constraints and objectives. A survey of multi-objective optimisation heuristics for a variety of multicast routing problems is presented by Fabregat *et al.* (2005). The authors review and classify multi-objective multicast routing heuristics according to their objective functions, constraints and methods used in the heuristics. Based on the classification, a General Multi-objective Multi-tree model is proposed, which considers multi-tree multicast (i.e. different trees can be built from a source to the same multicast group) with splitting (i.e. a flow can be split into sub-flows) in a multi-objective context. A multi-objective evolutionary algorithm inspired by the Strength Pareto Evolutionary Algorithm (Zitzler and Thiele, 1999) has been implemented for the multicast routing problem with multiple objectives and constraints.

3.3.1 Network Model for QoS Multicast Routing Problems

To model the general multicast routing problem, a computer network is represented by a connected, directed graph $G = (V, E)$ with $|V| = n$ nodes and $|E| = l$ links, where V is a set of nodes and E is a set of links, respectively. The nodes in V include a source node s, a set of destination nodes which receive data stream from the source, denoted by $R \subseteq V - s$, called the multicast group. A multicast routing problem is to build a multicast tree rooted at the source which spans all the destination nodes in the multicast group. The $|R|$ (the cardinality of R) represents the number of destinations, called the group size. Relay nodes in a multicast tree are intermediate nodes, excluding the source and destination nodes, along the paths from the source to destinations. The following notations are used in the rest of the thesis:

$(i, j) \in E$: Link from node i to node j, $i, j \in V$.

$c_{ij} \in \Re^+$: Cost of link (i, j), such as the bandwidth utilisation or communication cost.

$d_{ij} \in \Re^+$: Delay of link (i, j), sum of queuing delay, transmission delay and propagation delay of the link. Queuing delay is the waiting time of the data in the routing queue. Transmission delay is the time of pushing the data onto the network link. Propagation delay is the transmitting delay of the data before reaching its destination.

$z_{ij} \in \Re^+$: Capacity of link (i, j), measured in Mbps.

$t_{ij} \in \Re^+$: Current traffic of link (i, j), measured in Mbps.

$\phi \in \Re^+$: Traffic demand (bandwidth requirement) of a multicast request, measured in Mbps.

$P(u, v) = \{(u, i), (i, j), \ldots, (k, v)\}$: A path from node u to v as an ordering set of links.

$T(s, R)$: Multicast tree rooted at source s and spanning all members of R.

$Cost(T)$: Cost of the tree, given by $Cost(T) = \sum_{(i,j) \in T} c_{ij}$, i.e. the sum of the costs of all links in the tree.

$p_T(s, r_d) \subseteq T(s, R)$: Path connecting source s and a destination $r_d \in R$ in the multicast tree T.

$d(p_T(s, r_d))$: Delay of path $p_T(s, r_d)$, given by $d(p_T(s, r_d)) = \sum_{(i,j) \in p_T(s, r_d)} d_{ij}$, $r_d \in R$.

$Delay(T)$: Delay of the tree, given by $Delay(T) = Max\{d(p_T(s, r_d)), r_d \in R\}$, i.e. the maximum delay among all the paths from source to each destination.

In the network model, 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 $c_{ij} \neq c_{ji}$, $d_{ij} \neq d_{ji}$, $z_{ij} \neq z_{ji}$, and/or $t_{ij} \neq t_{ji}$.

For the ease of understanding, Figure 3.1 presents a simple example of a random directed network graph with $|V| = 9$ nodes and $|E| = 14$ links; the numbers beside each link are the cost and delay of the directed link, i.e. c_{ij} / d_{ij}, the source node $s = 5$, the multicast group $R = \{0, 2, 7\}$. An example multicast tree connected by bold arrow lines is shown in the figure below.

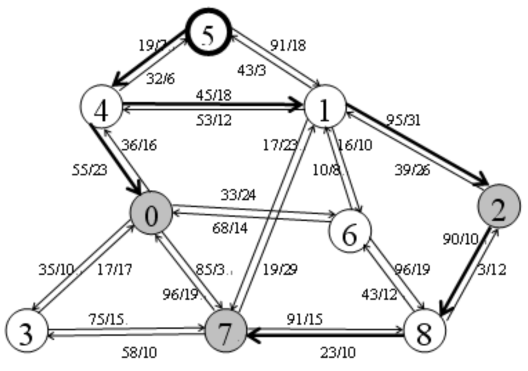


Figure 3. 1 An example of a random network graph and a multicast tree

Generally, given a network topology, a source node s, a multicast group R, and a set of optimisation objective functions, the multicast routing problem is to construct a multicast tree that optimises the objective functions. One important and common optimisation objective is defined as minimising the cost of a multicast tree. In order to support QoS requirements for the real time applications, a set of constraints, such as the end-to-end delay bound, delay jitter, bandwidth and packet loss ratio, is given to the constrained multicast routing problem. Depending on the constraints considered and the objective function used, various multicast routing problems exist in the literature. These include the bandwidth-constrained minimum cost multicast routing problem; the delay-constrained least-cost (DCLC, also called the delay-constrained minimum cost) multicast routing problem; the delay-bandwidth-constrained minimum cost multicast routing problem; and the delay-delay jitter-constrained minimum cost multicast routing problem. None of these problems are polynomial solvable, because of this, numerous algorithms have been proposed in the literature. This thesis focuses on the QoS multicast routing problem, i.e. the constrained multicast routing problem.

3.3.2 Network Simulator for QoS Multicast Routing Problems

In order to evaluate the performance of the proposed QoS multicast routing algorithms, researchers normally use network simulator to run experiments. There are several tools available which can be used as the network simulators, such as the OPNET, NS (Network Simulator), Matlab, and other random network topology generators. In this thesis, a QoS multicast routing simulator (MRSIM) has been implemented in C++ based on Salama's generator (Salama *et al.*, 1997). MRSIM generates random network topologies using the Waxman model described by Waxman (1988). The Waxman model has been widely used for algorithm evaluations in communication networks. The positions of the nodes in MRSIM are fixed in a rectangle of size $4000 \times 4000 km^2$. The link delay is defined as the propagation delay of the link, and queuing and transmission delays are negligible. The link cost is assigned as the current total bandwidth reserved on the link. The Euclidean metric is then used to determine the distance between pairs of nodes (u, v) with a probability is given by

$$P(u, v) = \beta \exp(-l(u, v)/\alpha L) \qquad \alpha, \beta \in (0,1] \tag{3.1}$$

where $l(u, v)$ is the distance from node u to node v. The parameters α and β can be set to obtain desired characteristics in the graph. For example, a large β gives nodes with a high average degree, and a small α gives long connections. L is the maximum distance between two nodes.

3.3.3 Summary of QoS Multicast Routing Algorithms

The QoS multicast routing problem has attracted the attention of many researchers over the past twenty years and consequently, many algorithms have been proposed to solve diverse QoS multicast routing problems. This section gives a brief summary of a number of QoS multicast routing algorithms in the literature.

The QoS multicast routing problems are initially considered within the single-objective context by minimising the cost of the multicast tree subject to certain constraints. Most of the early algorithms are based on heuristics, which can be classified into two main categories: source-based (or centralized) and distributed multicast routing algorithms. The source-based multicast routing algorithms assume that each node has all the network status information necessary to construct the multicast tree (e.g. Kompella *et al.*, 1993a; Widyono, 1994; Sun and Langendoerfer, 1995; Zhu *et al.*, 1995; Haberman and Rouskas, 1997; Sun and Langendoerfer, 1998; and Zhang *et al.*, 2001). Centralized multicast routing algorithms are not practical for large-scale networks, since they assume each node has the complete information of the network for constructing the multicast tree. Some distributed multicast routing algorithms have been proposed. In the distributed multicast routing algorithms, it is not necessary to maintain the entire network status information in each node, and every intermediate node participates in constructing the multicast tree (e.g. Kompella *et al.*, 1993b, Bauer and Verma, 1995; Shaikh and Shin, 1997; Jia, 1998; and Guo and Matta, 1999). Distributed multicast routing algorithms are normally problem-specific and have some shortcomings, such as being time consuming or complex to implement. With the emergence of metaheuristics in the area of operational research for solving a wide range of optimisation problems, some metaheuristic algorithms, including SA, GA, TS, PR and GRASP have in the past decade been investigated for various multicast routing problems.

Table 3.1 summarizes several heuristic and metaheuristic algorithms for the single-objective multicast routing problems with diverse QoS constraints in the literature, categorized by the type of heuristics, and ordered in the year of publication. From the table, it can be seen that one of the widely studied QoS multicast routing problems is the DCLC (delay-constrained least-cost) multicast routing problem, which concerns two of the commonest and most important QoS requirements, the cost and delay of the multicast tree. The cost of a multicast tree is defined as the sum of costs on all the edges in the tree. The tree delay is the maximal end-to-end delay of the paths from the source to each destination. The DCLC multicast routing problem is thus chosen as one of the case studies; the related algorithms for the problem will be discussed extensively in a later section.

Table 3. 1 Summary of single-objective QoS multicast routing algorithms in the literature

Algorithms		**Description**
Heuristics	Kompella *et al.* (1993a)	KPP, the first centralized heuristic for the delay-constrained least-cost multicast routing problem, based on Floyd's shortest path algorithm (Floyd, 1962)
	Kompella *et al.* (1993b)	DKPP, distributed heuristic for the delay-constrained least-cost multicast routing problem, based on Prim's minimum spanning tree algorithm(Cormen et al., 1997)
	Widyono (1994)	CAO, centralized heuristic for the delay-constrained least-cost multicast routing problem, based on Bellman-ford's shortest path algorithm (Cormen et al., 1997)
	Zhu *et al.* (1995)	BSMA, centralized heuristic for the delay-constrained least-cost multicast routing problem, based on based on Dijkstra's shortest path algorithm (Bertsekas and Gallager, 1992) and a *K*th shortest path algorithm (Eppstein, 1998)
	Sun and Langendoerfer (1995)	CDKS, centralized heuristic for the delay-constrained least-cost multicast routing problem, based on Dijkstra's shortest path algorithm
	Haberman and Rouskas (1997)	CCDVMA, centralized heuristic for the delay-delay jitter-constrained minimum cost multicast routing problem
	Sun and Langendoerfer (1998)	CKMB, centralized heuristic based on the KMB heuristic for Steiner trees, uses Dijkstra's shortest path algorithm
	Bauer and Varma (1995)	Distributed heuristics for the degree-constrained minimum cost multicast routing problem
	Shaikh and Shin (1997)	DDMC, distributed heuristic for the delya-constrained least-cost multicast routing problem, uses Prim's minimum spanning tree and Dijkstra's shortest path tree algorithms
	Jia (1998)	DSPH, distributed heuristic for the delay-constrained least-cost multicast routing problem with dynamic group changes, based on Prim's minimum spanning tree algorithm and Bellman-ford's shortest path algorithm
	Sriram *et al.* (1999)	CRCDM, centralized heuristic for the delay-constrained least-cost multicast routing problem with dynamic group changes
	Guo and Matta (1999)	QDMR, distributed heuristic for the delay-constrained least-cost multicast routing problem, based on DDMC
	Sun and Langendoerfer (1999)	Distributed heuristic for the delay-constrained least-cost multicast routing problem with dynamic group changes
	Zhang *et al.* (2001)	DCMA, centralized heuristic for the delay-constrained least-cost multicast routing problem
	Wang *et al.* (2002)	STH and CSH, group multicast routing heuristics with multiple multicast groups under a capacity limited constraint
ACO Algorithms	Lu and Liu (2000)	Distributed delay and delay-variation constrained minimum cost multicast algorithm
	Huang *et al.* (2007)	Source-based ant algorithm for the delay-constrained least-cost multicast routing problem
GA-based Algorithms	Wang *et al.* (2001)	Bandwidth-delay-constrained least-cost multicast algorithm
	Tran and Harris (2003)	Delay and delay-variation constrained least-cost algorithm for dynamic multicast routing
	Haghighat *et al.* (2004)	Delay and delay-variation constrained minimum cost multicast algorithm
	Hamdan and El-Hawary (2004)	Delay and delay-variation constrained minimum cost multicast algorithm
	Randaccio and Atzori (2006)	Group multicast routing algorithm under the bandwidth constraint
	Zahrani *et al.* (2008)	Capacity-constrained group multicast routing genetic local search algorithm
TS-based Algorithms	Youssef *et al.* (2002)	TS, tabu search algorithm for the delay-constrained least-cost multicast routing problem based on Dijkstra's shortest path algorithm

	Skorin-Kapov and Kos (2003)	TS-CST, tabu search algorithm for the delay-constrained least-cost multicast routing problem based on Prim's spanning tree algorithm
	Wang *et al.* (2004)	TSDLMRA, tabu search algorithm for the delay-constrained least-cost multicast routing problem based on the *K*th shortest path algorithm
	Ghaboosi and Haghighat (2006)	TS-based, tabu search algorithm for the delay-constrained least-cost multicast routing uses the *K*th shortest path and Prim's algorithm
SA-based Algorithms	Wang and Jiang (2004)	Delay-constrained least-cost multicast routing algorithm
	Zhang *et al.* (2005)	Distributed delay and delay-variation constrained minimum cost dynamic multicast algorithm
GRASP Algorithms	Skorin-Kapov and Kos (2006)	GRASP-CST, GRASP algorithm for the delay-constrained least-cost multicast routing problem applies a tabu search in the local search phase
PSO Algorithms	Sun *et al.* (2006)	QPSO, for the minimum cost multicast routing problem with multiple constraints including delay, bandwidth, delay jitter and package loss
PR Algorithm	Ghaboosi and Haghighat (2007)	Path relinking algorithm for the delay-constrained least-cost multicast routing problem based on the prüfer number
Hybrid Algorithms	Wang *et al.* (2005)	Hybrid PSO and GA algorithm for the minimum cost multicast routing problem with multiple constraints (bandwidth,delay,delay jitter, and loss rate)
	Zhang *et al.* (2009)	Hybrid GA and SA algorithm for the minimum cost multicast routing problem with multiple constraints including delay, bandwidth, and delay jitter

In addition to the DCLC multicast routing problem, algorithms for tackling multicast routing with diverse QoS requirements have been proposed. Bauer and Varma (1995) propose two approaches for the degree-constrained minimum cost multicast routing problem. The first approach is to design two distributed heuristic algorithms for the problem, namely, the shortest path heuristic (SPH), and the Kruskal-based shortest path heuristic (K-SPH) (Kruskal, 1956). Both heuristics are distributed by relying on the shortest path under degree constraint information available at each node. Distributed SPH starts from the source node and repeatedly connects other destination nodes to the sub-tree by the shortest degree-constrained path until all the destination nodes are included in the tree. Distributed K-SPH initializes the multicast tree from a forest of sub-trees, each sub-tree including the source or a destination node. The algorithm then repeatedly connects sub-trees into larger sub-trees by the shortest degree-constrained path until a single tree remains or until no further connections are possible. The second approach is a post-processing heuristic. It first constructs a Steiner tree with no degree constraint and then modifies the solution to satisfy the degree constraint. This is realized by removing the edges which connects the nodes over the degree constraints and finding alternative paths to reconnect the sub-trees so that the degree constraints are not violated. CCDVMA (Cost Conscious Delay and Delay Variation Multicast Algorithm) (Haberman and Rouskas, 1997) is a heuristic for the delay and delay-variation constrained minimum cost multicast routing problem. It first constructs a feasible tree of least cost paths from source to all destination nodes. Second, for each destination node, CCDVMA constructs a tree that includes the path from source to the destination in the feasible tree. Then the algorithm repeatedly adds more "good" paths, i.e. the path that satisfies the constraints, from on-tree nodes to off-tree destinations, until all destination nodes are included in the tree. The feasible tree with the least cost is eventually selected as the result. Lu and Liu (2000) present a distributed multicast routing algorithm based on ant colony optimisation for the delay and delay-variation constrained minimum cost multicast routing problem. A pheromone table is maintained at each node which gives the probabilities of selecting neighbouring links. An ant moves from one node to next node depending on the probability, i.e. the pheromone strength, for each neighbourhood link. The pheromone is updated to allocate more probability to the paths with smaller delay and delay variation. Hamdan and El-Hawary (2004) address the delay and delay-variation constrained low cost multicast routing problem using a genetic algorithm. The chromosome of each solution is represented by a string of genes; each gene is associated with a destination and represents a possible path from the source to the corresponding destination. After forming the initial population, the evolution process including reproduction, crossover and mutation is applied to evolve the current population to the next in a given number of generations. Zhang *et al*. (2005) propose an efficient distributed simulated annealing based algorithm for the delay and delay-variation constrained multicast routing problem. After constructing an initial feasible multicast tree that satisfies the delay and delay-variation constraints, a simulated annealing phase is designed to reduce iteratively the total tree cost without violating the two constraints. In addition, the algorithm supports dynamic changes in the multicast group. Sun *et al*. (2006) consider a QoS multicast routing problem to find a minimum cost multicast tree subject to a number of constraints including the delay, bandwidth, delay jitter and package loss. A quantum-behaved particle swarm optimisation (QPSO) algorithm is proposed to solve the problem. The particles in their QPSO algorithm use an integer coding that associates a position with a list of nodes from the source to the destination. The QPSO outperforms a GA and another PSO when applied to a small problem instance of 23 nodes.

Some recent novel hybridised algorithms have been proposed for solving the QoS multicast routing problem with multiple constraints. Wang *et al*. (2005) develop a hybrid algorithm combining PSO and GA to tackle the multi-constrained multicast routing problem, where the objective is to minimise the tree cost while satisfying a number of constraints including the bandwidth, delay, delay jitter and error rate in non-deterministic scenarios. The tests carried out show that their hybrid algorithm outperforms a single-GA algorithm on both the tree cost and the converge rate. Zhang *et al*. (2009) propose a hybrid genetic simulated annealing algorithm for the least cost QoS multicast routing problem with a number of constraints including the delay, delay jitter and bandwidth. The chromosome of a multicast tree is represented by a tree structure coding. The crossover operation in GA uses an adaptive crossover probability to improve the evolutionary efficiency. The mutation operation of GA is based on SA, where a neighbouring solution is accepted with the probability calculated by SA. Their simulation results show that the hybrid algorithm can solve the QoS multicast routing problem efficiently.

Some algorithms for the multicast routing problem with dynamic changes in the multicast group have also been proposed (Jia, 1998; Sriram *et al*., 1999; and Sun and Langendoerfer, 1999). The dynamic multicast routing problem presents a greater challenge, since the members in the multicast group can join in or leave the group at different times. For example, CRCDM (Controlled Rearrangement for Constrained Dynamic Multicasting) is proposed by Sriram *et al*. (1999) for the delay-constrained minimum cost multicast routing problem with dynamic group changes. The algorithm includes a new destination node by a delay-constrained shortest path to the existing multicast tree. When deleting a destination node, CRCDM takes actions depending on the position of the destination node. If the destination node is a leaf node, the destination node with unnecessary edges will be removed from the tree. If the destination node is an intermediate node and cannot be removed, the node will be marked. CRCDM uses a quality factor to represent the usefulness of a portion of the multicast tree to the overall dynamic routing process. Once the usefulness of a particular part of the tree drops below a threshold, a rearrangement technique is applied to modify the tree. Tran and Harris (2003) propose a genetic algorithm to address the dynamic multicast routing problem. By rearranging a part of the existing multicast tree, the algorithm reduces the tree cost, while the end-to-end delay and delay variation constraints remain satisfied.

In reality, a number of multicast applications may try to use the network to send information to multiple destinations at the same time. This creates a new and more complex network optimisation problem, namely the group multicast routing problem, which consists of multiple multicast sessions concurrently. In this case, the network resources, such as capacity (bandwidth), must be shared accordingly with the requirements of a group of multicast sessions. As a result, this group multicast routing problem with capacity constraints can be modelled as the multicast packing problem and has attracted the attention of some researchers (Chen *et al*., 1998; Wang *et al*., 2002; Randaccio and Atzori, 2006; and Zahrani *et al*., 2008), as described in the literature.

For example, Wang *et al*. (2002) consider the multicast packing problem with multiple multicast groups under the limited capacity constraint. The authors propose two heuristic algorithms, the Steiner-tree-based heuristic and the cut-set-based heuristic, to solve the problem. However, the simulation results show that if the available bandwidth is only just enough, the two algorithms may fail to find a solution even if the solution exists. Randaccio and Atzori (2006) propose a genetic algorithm to build a set of multicast trees simultaneously for the group multicast routing problem under the bandwidth constraint. A single cost function is used by combining the transmission delay and the bandwidth usage. Experimental results show that the proposed algorithm is able to obtain a better distribution of multiple session traffics. Zahrani *et al*. (2008) design a genetic local search algorithm with pre-processing by logarithmic simulated annealing for the group multicast routing problem for cost minimisation subject to the capacity constraint. In the algorithm, a solution is represented in a sequence of the group of multicast requests. The partially mixed crossover operation is applied to pairs of parents. A logarithmic simulated annealing procedure is used to estimate the maximum depth of the deepest local optima in the landscape. Based on the estimation, a local search is further applied to improve each individual. Their experimental results demonstrate that the proposed genetic local search algorithm can achieve better results.

In contrast to the traditional single-objective algorithms, some multi-objective algorithms have been developed in recent research for solving multicast routing problems with more realistic features. When different conflicting objectives are considered simultaneously, multicast routing problems become much more complicated multi-objective optimisation problems, as already recognized by many researchers (Fabregat *et al.*, 2005). Table 3.2 summarizes a number of algorithms developed for solving multi-objective multicast routing problems in the literature, categorized by the objectives and constraints considered in the work, and ordered by the year of the publication. It can be seen that different metaheuristics, e.g. genetic algorithm, ant colony algorithm, artificial immune algorithm and particle swarm optimisation, have been investigated for multi-objective multicast routing problems with various objectives. Due to the nature of multi-objective optimisation, where a set of alternative solutions is considered, it is not surprising that genetic algorithms, one of the most studied population-based algorithms, have been adapted in most multi-objective multicast routing algorithms. Other population-based algorithms, such as an ant colony optimisation algorithms designed by Pinto and Baran (2005), and a hybrid genetic algorithm with particle swarm optimisation algorithm proposed by Li *et al*. (2007), have also been developed in the multi-objective multicast routing literature. Detailed review of these multi-objective multicast routing algorithms is presented in Section 3.5, where the multi-objective multicast routing problem will be modelled and discussed as a case study.

Table 3. 2 Categorized multi-objective multicast routing algorithms by the objectives and constraints considered in the problem (MOGA: multi-objective genetic algorithm)

Algorithms	**Objectives**								**Constraints**		
	Cost	**Delay**	**Average Delay**	**Delay Jitter**	**Link Utilisation**	**Bandwidth**	**Packet Loss**	**Hop Count**	**Bandwidth**	**Delay**	**Delay Jitter**
MOGA (Roy *et al.*, 2002) based on Non-dominated Sorting genetic algorithm		x			x	x					
MOGA (Cui *et al.*, 2003) based on Pareto dominance		x			x		x				x
MOGA (Roy and Das, 2004) based on Niched Pareto Genetic Algorithm for the QoS-based mobile multicast routing		x			x		x				
MOGA (Crichigno and Baran, 2004a) and an improved MOGA (Crichigno and Baran, 2004b) based on Strength Pareto Evolutionary Algorithm	x	x	x		x				x		
A heuristic algorithm (Donoso *et al.*, 2004a) for the multi-objective multicast routing problem		x			x	x		x	x		
Multi-objective optimisation model (Donoso *et al.*, 2004b) for dynamic multicast routing		x			x	x		x	x		
MOGA (Fabregat *et al.*, 2004) based on Non-dominated Sorting genetic algorithm		x			x			x			
Multi-objective ant colony optimisation system and multi-objective Max-Min ant system (Pinto and Baran, 2005)	x	x	x		x				x		
MOGA (Fabregat *et al.*, 2005) based on Strength Pareto Evolutionary Algorithm		x		x	x	x		x			
Multi-objective immune algorithm (Wang *et al.*, 2006)	x	x				x			x	x	
A hybrid genetic algorithm and a particle swarm optimisation algorithm (Li *et al.*, 2007)	x	x	x		x				x		
MOGA (Rai *et al.*, 2010) with Fuzzy based parameter setting for QoS multicasting in wireless ad hoc networks		x		x			x				
MOGA (Huang and Liu, 2010) for QoS multicasting in wireless ad hoc networks		x		x		x	x				

3.4 The DCLC Multicast Routing Problem

3.4.1 Introduction

In real-time applications, two of the most important QoS requirements for constructing multicast trees are the delay and cost. The end-to-end delay from the source to each destination is the sum of delays along the path. The cost of the multicast tree is the sum of cost on the edges in the multicast tree. The minimum cost tree is called a Steiner tree (Hwang and Richards, 1992). The problem of finding a Steiner tree is known to be NP-complete (Garey *et al*., 1977; Garey and Johnson, 1979). The DCLC multicast routing problem is equivalent to the Delay-Constrained Steiner tree (DCST) problem, which is also NP-complete (Guo and Matta, 1999).

3.4.2 Problem Definition

Applications may assign different delay bounds for each destination $r_d \in R$. In this thesis, the delay bound for all destinations is assumed to be the same, and is denoted by Δ. Given the definitions in Section 3.2.1, the DCLC multicast routing problem, i.e. the DCST problem, can be formally defined as follows:

The Delay-Constrained Steiner Tree* (*DCST*) *Problem: *Given a network G, a source node s, a destination node set R, a link delay function d*(.)*, a link cost function c*(.)*, and a delay bound Δ, the objective of the Delay-Constrained Steiner Tree* (*DCST*) *Problem is to construct a multicast tree T∈T*(*s, R*) *such that the tree delay Delay*(*T*) *is within the delay bound, and the tree cost Cost*(*T*) *is minimised. So we can define the objective function:*

$$\min\{Cost(T) \mid Delay(T) \leq \Delta;\ T \in T(s, R)\} \tag{3.2}$$

3.4.3 Related Work for the DCLC Multicast Routing Problem

As previously reviewed, the DCLC multicast routing problem has been well studied, resulting in many heuristics and metaheuristics that construct the delay-constrained minimum cost multicast tree proposed in the literature. Most heuristic algorithms are classified as source-based if the source is assumed to have all the information necessary to construct the multicast tree, while others are classified as distributed algorithms, if they are not required to maintain the entire network status information in each node, and multiple nodes are participating in constructing the multicast tree. This subsection reviews the related heuristics including source-based and distributed heuristics, and metaheuristic algorithms for the DCLC multicast routing problem. The aim is to give a brief overview of the existing approaches proposed for this problem in the literature. The summary of algorithms reviewed in the following subsections is not intended to be exhaustive, as only some representative or well-known algorithms are included.

3.4.3.1 Source-based Heuristics for the DCLC Multicast Routing Problem

Some typical source-based heuristic algorithms include CAO (Widyono, 1994), KPP (Kompella *et al*., 1993a), CDKS (Sun and Langendoerfer, 1995), BSMA (Zhu *et al*., 1995) and CKMB (Sun and Langendoerfer, 1997).

KPP (Kompella-Pasquale-Polyzos) is the first source-based heuristic for DCLC multicast routing problems (Kompella *et al*., 1993a). It computes constrained paths for Steiner tree problems, assuming that the link delays and delay bounds are integers, while the link cost may take any non negative real value. KPP extends the KMB heuristic by taking the delay constraint into consideration. KPP starts with reducing the network to a complete graph over the source node and all the destinations nodes in the multicast group, where each link in the graph represents a delay-constrained shortest path between the two nodes and its cost is the length of the shortest path in the original network. KPP then uses the Prim's algorithm to construct a minimum spanning tree for the complete graph under the delay constraint. Finally, KPP replaces the links in the minimum spanning tree with the paths in the original network and prunes the non-destination leaves to obtain a multicast tree. The time complexity of KPP is $O(\Delta n^3)$, where Δ is the delay bound, n is the number of nodes in the network.

CAO (Constrained Adaptive Ordering) proposed by Widyono (1994) is based on the Bellman-Ford shortest path algorithm by considering the delay constraint in the heuristic. It incrementally adds a new tree branch by merging in the delay-constrained minimum-cost unicast path from the current sub-tree leading to an unconnected destination node until all destination nodes are included in the tree. The time complexity of CAO is $O(n^2)$. However, this complexity can be exponential in many cases and thus makes CAO is not suitable for large-scale networks.

CDKS (Constrained Dijkstra's) (Sun and Langendoerfer, 1995) is a heuristic for constructing delay-constrained multicast trees based on the Dijkstra's shortest path algorithm. It first constructs a delay-constrained low cost tree and then computes the delay-constrained shortest path for those destination nodes which are not yet included in the constructed multicast tree.

BSMA (Bounded Shortest Multicast Algorithm) (Zhu *et al*., 1995), one of the best known delay-bounded multicast routing algorithms developed in the 1990s, starts by computing a delay-bounded shortest path tree for the given source and multicast group. It defines a *superedge* as the longest simple path between two end nodes such that all internal nodes are relay nodes (i.e. nodes in the multicast tree excluding the source and the destination nodes) with the node degree equals to 2. It iteratively replaces a *superedge* in the tree with delay-constrained lower cost path until the tree cost cannot be further reduced. A *k*-Shortest path algorithm (Eppstein, 1998) is used to find the lower cost path. Due to its efficient performance on tree cost, it was regarded as the best deterministic DCLC multicast algorithm and is still being frequently used to compare the performance of many recent multicast routing algorithms. However, BSMA requires excessive execution time for large size networks as it uses the *k*-shortest path algorithm. This can be seen in its high time complexity $O(kn^3\log(n))$, where k is the parameter used in the *k*-shortest path algorithm.

Like the KPP heuristic, CKMB (Constrained KMB), proposed by Sun and Langendoerfer (1998), also extends KMB by adding the delay constraint into the KMB heuristic. CKMB starts by defining a closure graph over the source and the destinations in the multicast group. From the closure graph, CKMB ends with a delay constrained multicast tree. The main goal of CKMB is to reduce the time complexity of KPP. Instead of using the Prim's algorithm in KPP, a greedy approach similar to the Dijkstra's shortest path algorithm is used in CKMB to construct a constrained spanning tree of the closure graph. As a result, the time complexity of CKMB is reduced to $O(mn^2)$, where m is the number of destination nodes.

3.4.3.2 Distribution Heuristics for the DCLC Multicast Routing Problem

Beside the source-based heuristics, some distributed heuristics have been proposed, such as DKPP (Kompella *et al.*, 1993b), DDMC (Shaikh and Shin, 1997), DSPH (Jia, 1998), and QDMR (Guo and Matta, 1999).

DKPP (Distributed KPP) is a distributed minimum spanning tree heuristic based on the Prim's algorithm. DKPP starts from the source node and spans the tree by adding destinations one at a time until all destination nodes are included in the tree. One destination is added to the tree by a shortest path with extra checking to ensure that the delay constraint is satisfied during the expansion of the tree.

DSPH (Distributed Shortest Path Heuristic) is a shortest path based distributed algorithm for the DCLC multicast routing problem. It sequentially expands the multicast tree by a delay bounded shortest path from the tree to one destination until all destination nodes are covered. Although DSPH has good performance on tree cost, its computational time is extremely high, especially for large networks.

DDMC (Destination-Driven MultiCasting) uses a greedy strategy combining the Prim's minimum spanning tree and the Dijkstra's shortest path algorithms. It operates in a similar way to the Prim's algorithm. The primary difference is that the incremental cost of a destination node to the adjacent node is set to zero, so that DDMC reduces the tree cost by biasing toward paths that pass through destination nodes. DDMC shows significant improvement in terms of tree cost compared with other shortest path tree algorithms.

QDMR (QoS Dependent Multicast Routing) extends the DDMC algorithm to obtain delay-constrained low-cost multicast trees. Similar to DDMC, QDMR uses a weight function to adjust dynamically how far the current on-tree node is from violating the delay bound. QDMR expands the current tree by adding the node with the lowest weight until all destination nodes are mounted in the tree.

3.4.3.3 Metaheuristics for the DCLC multicast routing problem

In recent years, metaheuristic algorithms such as genetic algorithm (Wang *et al.*, 2001); tabu search (Youssef *et al.*, 2002; Skorin-Kapov and Kos, 2003; Wang *et al.*, 2004; Yang and Wen, 2005; Ghaboosi and Haghighat, 2006); simulated annealing (Wang and Jiang, 2004), GRASP (Skorin-Kapov and Kos, 2006); ant colony optimisation (Huang *et al.*, 2007), and path relinking (Ghaboosi and Haghighat, 2007) have been investigated for the DCLC multicast routing problem.

Youssef *et al.* (2002) 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 the Dijkstra's shortest path algorithm. In a given iteration, the algorithm refines the tree to a lower cost. Skorin-Kapov and Kos (2003) propose a tabu search-based algorithm called TS-CST (Tabu Search - Constrained Steiner Tree) for the DCLC multicast routing problem. TS-CST uses a binary string to represent a solution (a multicast tree). The neighbouring solutions include all the solutions whose binary representation is exactly one bit different in the binary string from the current solution. A move in the TS-CST is to choose the best new neighbouring solution as the current solution in the next iteration. A tabu list of length one prevents the algorithm from repeating the last performed move. TS-CST stops after a pre-defined number of iterations without improvements. Wang *et al*. (2004) propose an efficient DCLC Multicast Routing algorithm based on tabu search. The main disadvantage of the algorithms proposed in Youssef *et al*. (2002) and Wang *et al.* (2004) is that they randomly select the paths to be added or deleted, which may lead to a disjointed multicast tree. Yang and Wen (2005) apply tabu search to the problem of pre-planning delay-constrained backup paths for multicast trees to minimise the total cost of all the back up paths. Ghaboosi and Haghighat (2006) propose a tabu search-based algorithm for the DCLC multicast routing problem. The tabu search algorithm creates initial solutions based on the 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.

Wang and Jiang (2004) propose a simulated annealing algorithm for the DCLC multicast routing problem. Compared with a genetic algorithm, their experimental results show that the simulated annealing algorithm has the ability to converge to the optimal solution in a short time. This indicates that the simulated annealing algorithm is a high-efficient approach in multicast routing.

Skorin-Kapov and Kos (2006) develop a GRASP metaheuristic for the DCLC 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 TS-CST (Tabu Search - Constrained Steiner Tree) (Skorin-Kapov and Kos, 2003) 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.

Ghaboosi *et al*. (2007) present the first path relinking approach for the DCLC multicast routing problem. Their algorithm applies prüfer numbers to represent multicast trees, i.e. solutions. After generating a reference set of random initial solutions, the path relinking algorithm operates repeatedly on pairs of randomly chosen elite solutions. A repair procedure deals with 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 during the path relinking phase, especially when the network size increases and many infeasible solutions may occur.

Huang *et al*. (2007) propose a source-based ant algorithm for the DCLC multicast routing problem. The algorithm starts from constructing a set of backup paths from the source to each destination by using the Dijkstra's shortest path algorithm. The movement of an ant from one node to another depends on the corresponding pheromone strength of each neighbouring link given by a probability function. At the end of each iteration, the pheromone will be updated to leave more pheromone on the link with shorter journeys by an ant. Simulation results show the ant algorithm is effective in terms of cost and convergence speed. One shortcoming of the source-based algorithm is its scalability for large-scale networks.

3.4.3.4 Landscape of the DCLC Multicast Routing Problem

As reviewed above, many heuristic and metaheuristic algorithms have been proposed for the DCLC multicast routing problem, while most of these studies focus mainly on the performance of the proposed algorithm by showing that the algorithm is effective and superior to other algorithms. The performance of a given algorithm is determined by its efficiency and its effectiveness. The quality of a solution found by the algorithm reflects the effectiveness of an algorithm, and the computational time is usually a measure of the efficiency of the algorithm. Instead of applying a specific heuristic or metaheuristic for a particular optimisation problem, a useful research direction in operational research is to analyse the fitness landscape of the problem before designing a certain search algorithm. The theoretical analysis of fitness landscape has shown to be useful for observing the behaviour of search algorithms and thus can help in predicting their performance (Wright, 1932). However, few efforts have been made in establishing the theoretical foundations of the landscape analysis of the QoS multicast routing problem. There are only three related studies of the landscape analysis for the multicast routing problem conducted by Zahrani *et al*. (2006; 2007; 2008). In (Zahrani *et al*., 2006), a logarithmic simulated annealing (LSA) is used to perform more detailed landscape analysis on three instances. They emphasise on the approximation of optimal parameter settings. The difference between two parameters; the estimation of the maximum value of the minimum escape height from the local optima; and the maximum increase value of the objective function between two successive improvements of the best objective function value found during the local search time, are observed in LSA by a number of experiments. The results show that the difference depends mainly on the cost function and the capacity constraint, while only slightly on the particular network structure. Zahrani *et al*. have further extended their work (2007), by introducing a LSA based genetic local search (GLS) algorithm for the landscape analysis on the multicast routing packing problem, where a sequence of multicast requests is scheduled. In the algorithm, a solution is represented as an ordered sequence of a group multicast requests. The simple KMB (Kou *et al*., 1981) heuristic is used to construct the single multicast tree for each multicast request, where two QoS requirements (the cost and capacity) are considered. The GLS applies the partial mixed crossover (PMX) operation to pairs of individuals. As the pre-processing step in the GLS, a simulated annealing procedure with a logarithmic cooling schedule performs the landscape analysis by estimating the depth of the deepest local minima. Experimental results on two benchmark instances show that the proposed LSA-based GLS together with the PMX operation outperforms two variants of GLS algorithms with either LSA or PMX only. The same GLA algorithm with a LSA as the pre-processing step is applied by Zahrani *et al*. (2008) on more instances for the group multicast routing problem with the capacity constraint. The PMX operation is used on pairs of parents. Similarly, a simulated annealing with the logarithmic cooling schedule is used to estimate the value of maximum depth of local optima in the landscape. A local search procedure is then applied by swapping two random points in each individual. The same conclusion that the GLS combined with LSA and PMX outperforms the algorithms using either LSA or PMX is obtained as that drawn by Zahrani *et al*. (2007).

To the best of my knowledge, no research has been carried out on the landscape analysis for the DCLC multicast routing problem. In order to give a better understanding of landscape properties for the DCLC multicast routing problem, the first landscape analysis on the problem will be presented in Chapter 5. This may contribute to more advanced search algorithms for various related applications of the DCLC multicast routing problem in the future by providing the method to predict the performance of the designed algorithms.

3.4.4 Benchmark Instances of the DCLC Multicast Routing Problem

In this subsection, some benchmark instances of the DCLC multicast routing problem in the literature are summarised. These instances will be used to test the proposed algorithms.

1) Steinb Problems from the OR-library

Steinb is a set of 18 small and medium sized (50-100 nodes) benchmark instances from SteinLib in the OR library (Beasley, 1990). SteinLib is a publicly available library of test instances for the MStTG problem. Characteristics of Steinb instances are given in Table 3.3.

Table 3. 3 The problem characteristics of Steinb from the OR-library (|*V*|, |*E*|, |*R*| denote the number of nodes, the number of edges and the number of destinations in the instances, respectively, 'OPT' denotes the optimal solution)

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

2) Steinc Problems from the OR-library

Steinc is a set of large and hard networks with 500 nodes in the OR-library. Details of Steinc instances are given in Table 3.4.

Table 3. 4 The problem characteristics of Steinc from the OR-library (|*V*|, |*E*|, |*R*| denote the number of nodes, the number of edges and the number of destinations in the instances, respectively, 'OPT' denotes the optimal solution)

No.	**|V|**	**|E|**	**|R|**	**OPT**	**No.**	**|V|**	**|E|**	**|R|**	**OPT**	**No.**	**|V|**	**|E|**	**|R|**	**OPT**	**No.**	**|V|**	**|E|**	**|R|**	**OPT**
C01	500	625	5	85	**C06**	500	1000	5	55	**C11**	500	2500	5	32	**C16**	500	12500	5	11
C02	500	625	10	144	**C07**	500	1000	10	102	**C12**	500	2500	10	46	**C17**	500	12500	10	18
C03	500	625	83	754	**C08**	500	1000	83	509	**C13**	500	2500	83	258	**C18**	500	12500	83	113
C04	500	625	125	1079	**C09**	500	1000	125	707	**C14**	500	2500	125	323	**C19**	500	12500	125	146
C05	500	625	250	1579	**C10**	500	1000	250	1093	**C15**	500	2500	250	556	**C20**	500	12500	250	267

As can be seen from the tables, the optimal solution for each instance on both Steinb and Steinc is known in the OR-library. The network topology of each instance is generated by using the simulator MRSIM. Since Steinb and Steinc instances are for the Steiner tree problem, only a cost function is assigned to the links in each instance. The delays of the links are randomly assigned for each instance in the experiments.

3.5 The Multi-Objective Multicast Routing Problem

3.5.1 Introduction

In the literature of the QoS multicast routing problem, most early algorithms treat the problem as a single-objective problem with several constraints. Lately, the multi-objective context of the multicast routing problem has been gradually recognized (Roy *et al*., 2002), and since 2003 an increasing number of algorithms have been proposed as shown in Table 3.2. When a range of inter-dependent and conflicting multiple QoS objectives and constraints (e.g. the cost, delay, bandwidth, link utilisation, delay variation, packet lost ratio and hop count) in real world applications are considered simultaneously, the QoS-based multicast routing problems can be more appropriately defined as multi-objective optimisation problems.

3.5.2 Problem Definition

As can be seen in Table 3.2, a range of QoS requirements exist in the multicast routing problem. If all these requirements are considered simultaneously, the QoS multicast routing problem will become extremely complicated to be solved. This study concerns the multi-objective multicast routing problem with five common and important objectives for static computer networks where the status of the networks is known and remains unchanged. In addition to the four objectives, i.e. the cost, the maximum end-to-end delay, the link utilisation, and the average delay of the multicast tree, considered in (Crichigno and Baran, 2004a), another important QoS requirement, the delay variation of the multicast tree, is chosen as the fifth objective. These five QoS requirements are selected due to the following reasons. The cost of the multicast tree is the main concern of service providers since it directly influences the network resource utilisation. For many delay-sensitive interactive multimedia applications, the maximal end-to-end delay, average delay, and delay variation of the multicast tree should be minimised to make sure the transmitted information arrive at the receivers in a real-time manner. The link utilisation is a basic requirement for transmitting information, and directly reflects the congestion of traffic transmission. Based on the notations defined in Section 3.3.1, the multi-objective multicast routing problem concerned in this thesis is then defined to find a multicast tree while minimising the values of the following five objectives:

The cost function of the multicast tree: $C(T) = \phi \cdot \sum_{(i,j)\in T} c_{ij}$ (3.3)

The maximal end-to-end delay of the multicast tree:

$$DM(T) = Max\{d(p_T(s, r_d))\},\ r_d \in R \tag{3.4}$$

The maximal link utilisation:

$$\alpha(T) = Max\left\{\frac{\phi + t_{ij}}{z_{ij}}\right\},\ (i, j)\in T \tag{3.5}$$

The average delay of the multicast tree:

$$DA(T) = \frac{1}{|R|}\sum_{r_d\in R} d\,(p_T(s, r_d)) \tag{3.6}$$

The delay variation of the multicast tree:

$$DV(T) = Max\{d(p_T(s, r_d))\} - Min\{d(p_T(s, r_j))\},\ r_d,\ r_j \in R \tag{3.7}$$

Objective (3.3) aims to minimise the cost function of the multicast tree *T*, which is a combined measure of transmission costs and the bandwidth requirement of the multicast request. Objective (3.4) minimises the maximal delay time of sending the data via the multicast tree so that they arrive destinations within a shortest bounded time. Objective (3.5) tries to minimise the maximal link utilisation, i.e. traffic demand over the available bandwidth on the links. Objective (3.6) minimises the average delay time of sending the data so they arrive all $|R|$ destinations in the shortest possible time. Objective (3.7) minimises the delay variation of the multicast tree, which is defined as the difference between the maximum and minimum delays among all the path delays from the source to each destination. Note that objective (3.4) concerns the maximal end-to-end delay within the multicast tree, while objectives (3.6) and (3.7) minimise the average delay and the delay variation, respectively, concerning the delay to all destinations in the network. As indicated by the literature, these objectives represent the most common requirements in communications. It remains interesting future work to formulate a wide range of various objectives in the above defined problem for different applications with specific requirements.

In communication networks, the total bandwidth of datagram on a link must not exceed the limited bandwidth available. Thus, the total traffic on link (i, j), i.e. the traffic demand ϕ of a multicast request plus the current traffic t_{ij}, is subject to a link capacity z_{ij}:

$$\phi + t_{ij} \le z_{ij},\ \forall (i, j)\in T(s, R) \tag{3.8}$$

3.5.3 Related Work for the Multi-objective Multicast Routing Problem

As shown in Table 3.2, most of the algorithms for various multi-objective multicast routing problems with multiple objectives and constraints are based on metaheuristics. To the best of my knowledge, there are only two exceptions in studies by Donoso *et al*. (2004a; 2004b). Donoso *et al*. (2004a) propose a heuristic algorithm for the multi-objective multicast routing problem. The algorithm searches a set of multicast trees by optimising the weighted sum function composed of four QoS requirements: the maximal link utilisation, hop count, bandwidth consumption and end-to-end delay in the multicast tree. Donoso *et al*. (2004b) extend their work by proposing a multi-objective optimisation model for solving the same problem with dynamic multicast groups. In the case of dynamic membership changes in the multicast group, the model rearranges the previously constructed tree while considering the multiple available paths in order to optimise the defined multi-objectives.

The majority of metaheuristic approaches for the multi-objective multicast routing problem are based on MOGA. Some MOGA algorithms have been proposed for the problem in wireless ad hoc networks. For wireless networks, the situation becomes more sophisticated due to the unpredictable mobility of users and their variations in QoS requirements. Roy *et al*. (2002) adapt the multi-objective non-dominated sorting based genetic algorithm (Srinivas and Deb, 1994) to optimise simultaneously the end-to-end delay, bandwidth and residential bandwidth utilisation rather than combining them into a single weighted sum objective function for the multi-objective multicast routing problem in wireless networks. In the algorithm, each individual, i.e. the generated multicast tree, is encoded to a chromosome consisting of the sequence of nodes along the path from the source to each destination. The population is ranked based on the individual's non-dominance. After the tournament selection using the non-dominated sorting, the basic GA operations are applied to the selected parents to generate offspring solutions. Roy and Das (2004) present a QoS-based mobile multicast routing protocol based on MOGA. Simulation results demonstrate the proposed protocol is able to discover a set of near-optimal solutions within a few iterations and it is fast and efficient in tackling dynamic changes in the multicast group and imprecise network information in wireless networks. Rai *et al*. (2010) propose a MOGA algorithm for the QoS multicast routing problem in wireless networks. In the algorithm, a fuzzy-based selection technique is used for the initialization of multiple QoS parameters, including the bandwidth, end-to-end delay, delay jitter, packet loss rate and blocking probability. Their simulation results show the proposed algorithm is able to obtain near-optimal solutions for QoS multicast routing in varying network conditions. Huang and Liu (2010) present a MOGA algorithm based on the CBT (Core Based Tree, Ballardie *et al*., 1993) protocol. Experiment results show the proposed algorithm is capable of achieving fast convergence and obtaining a set of diversified non-dominated solutions.

For the multi-objective multicast routing problem within the scenario of static computer networks considered in this thesis, several algorithms based on MOGA have been proposed in the literature. Cui *et al*. (2003) propose a MOGA algorithm based on the Pareto dominance for the multi-objective multicast routing problem. The chromosome of each individual is encoded using the same representation as that used by Roy *et al*. (2002). The fitness of an individual is based on Pareto ranking and density calculation. The crossover and mutation operations are designed as those in standard GA. Experiments demonstrate that the proposed algorithm is capable of discovering a set of non-dominated solutions within a finite evolutionary generation. Two multi-objective evolutionary algorithms with an external population of Pareto optimal solutions have been proposed by Crichigno and Baran (2004a, 2004b) using the strength Pareto evolutionary algorithm (Zitzler and Thiele, 1999). Experimental analysis shows that the MOGA algorithm proposed by Crichigno and Baran (2004b), with a binary tournament selection, outperforms the MOGA algorithm proposed by Crichigno and Baran (2004a), with a roulette wheel selection.

A variety of other population-based metaheuristics also appear in the multi-objective multicast routing literature. Two ant colony optimisation algorithms, namely MOACS (Multi-Objective Ant Colony System) and M-MMAS (Multi-objective Max-Min Ant System), have been developed by Pinto and Baran (2005). Compared with Crichigno and Baran's traditional MOEA algorithm (2004b), both MOACS and M-MMAS are able to find more non-dominated solutions on some benchmark problems. Furthermore, MOACS can find generally better non-dominated solutions than M-MMAS and MOEA. Wang *et al.* (2006) propose a multi-objective multicast routing model based on an artificial immune system with a gene library and a clone search operator to search for better solutions. The algorithm can effectively identify a set of Pareto optimisation solutions compromising multiple QoS objectives. A Hybrid Genetic Algorithm and Particle

Swarm Optimisation algorithm, called HGAPSO, is investigated by Li *et al.* (2007) for the multi-objective multicast routing problem. In HGAPSO, new generations are created not only by the crossover and mutation operations in GA but also by PSO. The upper-half individuals in the population are chosen as elites and will be regarded as a swarm and enhanced by PSO before reproducing to the next generation. The hybrid algorithm outperforms the standard GA on randomly generated problems.

3.5.4 Benchmark Instances of the Multi-objective Multicast Routing Problem

Two commonly used benchmark instances of the multi-objective multicast routing problem in the literature are summaried as follows:

1) The NSF (National Science Foundation) Network

The NSF network was a major part of early 1990s Internet backbone and has been tested in the existing literature by a number of researchers as a benchmark problem (Cui *et al.*, 2003). The NSF network with 14 nodes is shown in Figure 3.2.

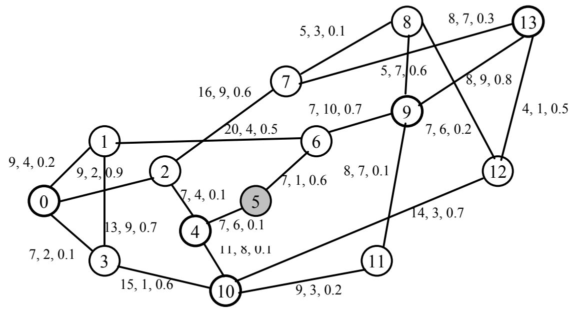


Figure 3. 2 The NSF network. d_{ij}, c_{ij} and t_{ij} are shown on each link.
$s = 5$, $R = \{0, 4, 9, 10, 13\}$, $\phi = 0.2$Mbps, $z_{ij} = 1.5$Mbps.

2) The NTT (Nippon Telephone Telegraph of Japan) Network

The NTT network is also a benchmark problem for testing the multi-objective multicast routing problem in the literature (see Crichigno and Baran, 2004b). The NTT network topology with 55 nodes and 142 links is shown in Figure 3.3. The cost c_{ij} of each link is set to 1, and the current traffic t_{ij} on each link is set as randomly loaded with around 50% of its total link capacity, which are the same as those in (Diego and Baran, 2005).

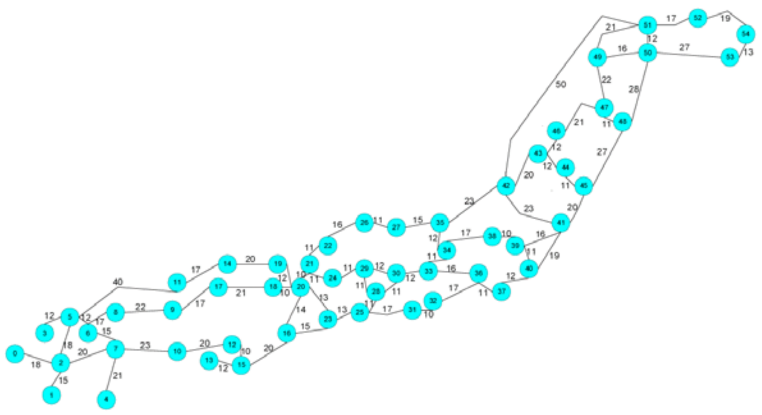


Figure 3. 3 The NTT network. Over each link (i, j), a delay d_{ij} is shown

3.6 Summary

This chapter has overviewed various QoS multicast routing problems and algorithms. Firstly, the Steiner tree problem as the underlying model of the multicast routing problem is introduced. Several related studies for the classical NP-complete problem are also discussed. Secondly, the general network model for the QoS multicast routing problem is presented. The network simulator used in this thesis for generating network topology and evaluating the performance of QoS multicast routing algorithms is introduced. A number of QoS multicast routing algorithms proposed in the literature are then categorized and briefly summarized. Thirdly, two QoS multicast routing problems, the DCLC multicast routing problem and the multi-objective multicast routing problem, are chosen as the case studies. The two NP-complete problems are then in turn formally defined. Related heuristic and metaheuristic algorithms selected from the literature for these two problems are reviewed. In addition, some benchmark instances in the literature for both problems are also introduced.

The main purpose of this chapter is to give an overview of the current research advances for the QoS multicast routing problem in computer networks. The importance and challenge of the QoS multicast routing problem in real world applications have attracted significant research attention in the areas of both computer networks and operational research. Many algorithms have therefore been proposed in the past two decades for solving various QoS multicast routing problems. Due to the efficiency and effectiveness of metaheuristics for tackling combinatorial optimisation problems, most of the state-of-the-art algorithms are based on metaheuristic methods, as reviewed in this chapter. However, as far as I know, some recent metaheuristic methods, such as variable neighbourhood search and Scatter Search, have not been investigated by researchers. In addition, some hybrid approaches have not been explored for the multi-objective multicast routing problem. Furthermore, few researches have been carried out into the analysis of the landscape of the QoS multicast routing problem in the literature. Based on the above observations, the motivation of the research in this thesis is to investigate effective and efficient metaheuristic approaches for solving the two selected multicast routing problems and to analyse the landscape properties of the DCLC multicast routing problem such that more advanced search algorithms for various related applications of the problem may be proposed in the future.

CHAPTER 4. Metaheuristic Approaches for the DCLC Multicast Routing Problem

4.1 Introduction

The emergence of metaheuristics for solving a variety of combinatorial optimisation problems is a rapid growing field in operational research. As reviewed in Chapter 2, metaheuristics are powerful algorithmic approaches which have been successfully applied to many difficult combinatorial optimisation problems. The DCLC multicast routing problem, as defined in Chapter 3, is a NP-complete combinatorial optimisation problem and numerous algorithms have been proposed in the literature for tackling the problem. In recent years, metaheuristic approaches for the DCLC multicast routing problem are gaining an increasing attention by some researchers. However, the applications of metaheuristics for the problem have not been fully explored. As far as I am concerned, some recent metaheuristics still remain unexplored for the problem. Motivated by this, four new metaheuristic approaches, including VND search algorithms, a hybrid GRASP algorithm, a JPSO algorithm, and a SS with PR algorithm, have been proposed for the DCLC multicast routing problem. A large amount of simulations has been carried out to test the performance of these proposed algorithms. Experimental results demonstrate the effectiveness and efficiency of these algorithms compared with some existing heuristics and algorithms. These four proposed algorithms will be presented in details in the following subsections.

4.2 Variable Neighbourhood Descent Search Algorithms

The basic idea of VNS is to systematically change the employment of different neighbourhoods within a local search. Unlike many standard metaheurisitcs where only a single neighbourhood is employed, VNS can traverse among different areas of the search space which are defined by different neighbourhood structures. This is due to that a local optimum defined by one neighbourhood structure is not necessarily the local optimum of another neighbourhood structure. The search of VNS is thus more flexible within the solution space of the problem, and potentially leads to better solutions which are difficult to obtain by using single neighbourhood based local search algorithms.

In this section, the first VNS approach is investigated for the DCLC multicast routing problem. Although VNS has been applied to Steiner tree problems, e.g. VNS as a post-optimisation procedure to the prize-collecting Steiner tree problem (Canuto *et al.*, 2001), and the bounded diameter minimum spanning tree problem (Gruber and Raidl, 2005), as far as I am aware, no other research has been carried out on using VNS for the DCLC multicast routing problem.

Let us denote by N_ℓ, $\ell = 1, ..., \ell_{max}$, a finite set of pre-designed neighbourhood structures, and by $N_\ell(x)$ the set of solutions by employing the ℓ^{th} neighbourhood operator upon an incumbent solution x. The proposed algorithm is based the variable neighbourhood descent (VND) search (Mladenović and Hansen, 1997), a deterministic version of VNS where the best neighbouring solution in each neighbourhood structure is always selected. The pseudo-code of the basic VND search is presented in Figure 4.1.

```
Initialization:
Define the set of neighbourhood structures $N_\ell$, $\ell = 1, \ldots, \ell_{max}$; generate an
initial solution $x$;
Repeat
    (1) Set $\ell = 1$;
    (2) Repeat the following steps until $\ell = \ell_{max}$:
        (a) Exploration of neighbourhood:
            Find the best neighbour $x'$ of $x$, $x' \in N_\ell(x)$;
        (b) Move or not:
            If the solution $x'$ is better than x, then $x \leftarrow x'$, $\ell = 1$;
            otherwise, set $\ell = \ell + 1$;
Until no improvement is obtained
```

Figure 4. 1 The pseudo-code of the basic VND

4.2.1 Algorithm Description

4.2.1.1 Initialization

In the proposed VND Multicast Routing (VNDMR) algorithm, firstly an initial solution T_0 is created and then iteratively improved while the delay constraint is satisfied, by employing three neighbourhoods until the tree cost cannot be reduced. As in (Zhu *et al.*, 1995), each solution, i.e. a multicast tree, is represented by using a predecessor array. For the network with $|V| = n$ nodes, there are n bits in the array and each bit (from 0 to $n-1$) represents one node in the network. The value of each bit in the array is set to the bit number of the current node's predecessor. To investigate the effects of different initial solutions within the VNDMR algorithm, two variants of the algorithm (namely VNDMR0 and VNDMR1) with the same neighbourhood structures are designed, but starting from different initial solutions produced by using the following two heuristics:

- Initialization by DKSLD (VNDMR0): the Dijkstra's shortest path algorithm is used to construct the least delay multicast tree;
- Initialization by DBDDSP (VNDMR1): The Destination-Driven Shortest Path (DDSP) algorithm in (Gruber and Raidl, 2005), a distributed shortest path unconstrained multicast tree algorithm, is extended by considering the delay bound in DDSP. This modified DBDDSP (Delay-Bounded DDSP) algorithm is used as the initialization in VNDMR1.

4.2.1.2 Neighbourhood Structures within the VNDMR Algorithms

The main characteristic of the VNDMR algorithm is to design three simple yet effective neighbourhood structures. Each neighbourhood aims to reduce the tree cost and at the same time satisfying the delay constraint. The pseudo-code of VNDMR is presented in Figure 4.2. The neighbourhood structures within VNDMR0 and VNDMR1 are based on an operation called path replacement, where a path in tree T_i is replaced by another new path not in tree T_i, resulting in a new tree T_{i+1}. The delay-bounded path replacement operation guarantees that tree T_{i+1} is always a delay-bounded and loop free tree. To present the candidate paths chosen in path replacement, a *superpath* is defined as any longest simple path between two nodes in tree T_i, in which all internal nodes, except the two end nodes of the path, are relay nodes and each relay node connects exactly two tree edges. The *superpath* is the same as *superedge* defined in BSMA (Zhu *et al*., 1995). The three neighbourhood structures of VNDMR0 and VNDMR1 are described as follows:

1. ***Neighbour1***: all the most expensive edges on each *superpath* in tree T_i are the candidates of path replacement. At each step, one chosen edge is deleted, which divides the tree T_i into two sub-trees T_i^1 and T_i^2. The Dijkstra's shortest path algorithm is then used to find a new delay-bounded shortest path that connects the two sub-trees and reduce the tree cost;
2. ***Neighbour2***: this operator operates on all *superpaths* in tree T_i. At each step, one *superpath* is replaced by a cheaper delay-bounded path using the same path replacement strategy in ***Neighbour1***;

3. ***Neighbour3***: all *superpaths* connect with destinations 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 sub-tree T_i' and a destination node $r_d \in R$. The same path replacement strategy is then used to search for a new delay-bounded shortest path from r_d which reconnects r_d to T_i'.

```
VNDMR($G = (V, E), s, R\ \Delta, \ell_{max}, N_\ell, \ell = 1,..., \ell_{max}$)
{   // $s$: source node; $R$: destination set; $\Delta \geq 0$ is the delay bound;
    // $\ell_{max} = 3$ is the number of neighbourhoods structures;
    // $N_\ell$: the $\ell^{th}$ neighbourhood of the set of neighbourhoods;
    Create initial solution $T_0$; // DKSLD or DBDDSP
    if $T_0 = NULL$ then return FAILED; // a feasible tree does not exist
    else
    {
      $T_{best} = T_0$; $\ell = 1$;
      while $\ell \leq \ell_{max}$ do {
           select the best neighbour $T'$, $T' \in N_\ell(T_{best})$;
           if((($Cost(T') < Cost(T_{best})$) & ($Delay(T') \leq \Delta$)) ||
              (($Cost(T') == Cost(T_{best})$) & ($Delay(T') < Delay(T_{best})$)))
           then { $T_{best} = T'$; $\ell = 1$; }
           else $\ell$++;
      end of while loop }
    }
    return $T_{best}$;
}
```

Figure 4. 2 The pseudo-code of the VNDMR Algorithm

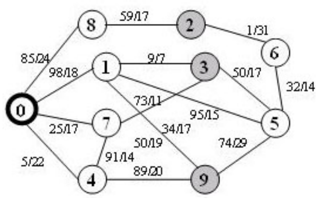
In order to investigate the influence of different neighbourhood structures on the performance of the VNDMR algorithm, another node-based neighbourhood structure is designed in VNDMR2. Both VNDMR1 and VNDMR2 start from the same initial solution (DBDDSP) but have different neighbourhood structures. The three neighbourhood structures of VNDMR2 are as follows:

1. ***Neighbour1***: one neighbourhood tree is defined as deleting or adding a non-destination node from the current multicast tree and creating a minimum spanning tree which spans the remaining nodes by using the Prim's spanning tree algorithm. Once a better tree is found, the current tree is updated. These steps are repeated until no better tree can be found more than 2 iterations;
2. ***Neighbour2***: the same as ***Neighbour2*** in VNDMR1;
3. ***Neighbour3***: the same as ***Neighbour3*** in VNDMR1.

4.2.1.3 An Illustrative Example

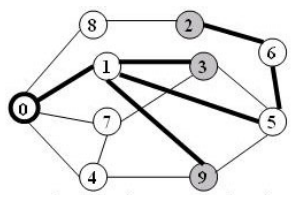
For the ease of understanding, an example network is presented in Figure 4.3(a), where $s = \{0\}$ and $R = \{2, 3, 9\}$, $\Delta = 82$ is the delay bound and the values on each link $e = (i, j)$ are the cost and delay of the link, i.e. c_{ij}/d_{ij}. The search results by applying BSMA and the VNDMR1 algorithm are shown in Figure 4.3(b) and Figure 4.3(d), respectively. The initial solution constructed by DBDDSP is shown in Figure 4.3(c).

According the definition of *superpath*, there are three *superpaths* in the multicast tree in Figure 4.3(c), path $P(0, 2) = \{(0, 8), (8, 2)\}$, path $P(0, 3) = \{(0, 7), (7, 3)\}$, and path $P(0, 9) = \{(0, 4), (4, 9)\}$. By using VNDMR1, $P(0, 3)$ is replaced by path $P(9, 3) = \{(9, 1), (1, 3)\}$, generating a new tree in Figure 4.3(d). The tree cost is reduced from 336 in Figure 4.3(c) to 281 Figure 4.3(d), while the tree delay ($Delay(T_{VNDMR1}) = 66$) still satisfies the delay constraint $\Delta = 82$. Comparing the trees generated by BSMA and VNDMR1, it can be seen that the tree cost found by VNDMR1 (281) is lower than that of BSMA (285). This is due to the flexibility of the VNDMR1 algorithm which explores the search space defined by different neighbourhood structures.



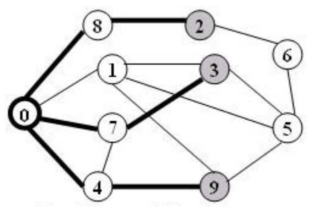
$c_{19}/d_{19} = 50/19$, $c_{91}/d_{91} = 34/17$

(a) An example network



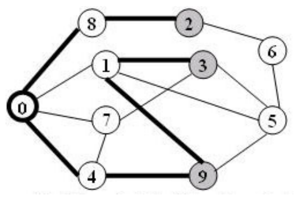
$Cost(T_{BSMA}) = 285$, $Delay(T_{BSMA}) = 78$

(b) Search result of BSMA



$Cost(T_0) = 336$, $Delay(T_0) = 42$

(c) Initial solution of VNDMR1



$Cost(T_{VNSMR}) = 281$, $Delay(T_{VNSMR}) = 66$

(d) Search result of VNDMR1

Figure 4. 3 An example network and solutions obtained by BSMA and VNDMR1

4.2.1.3 Time Complexity of the VNDMR Algorithm

Consider a network with n nodes, the proof of the probability of transition from spanning tree s_i to another tree s_j of the network has been given in (Zhu *et al*., 1995):

According to Cayley's Theorem (Cayley, 1989), there are n^{n-2} possible spanning trees on n nodes. The number of Steiner trees spanning $n_s \le n$ is thus bounded by n^{n-2}. Let us consider a Markov chain of n^{n-2} states, where each state corresponds to a spanning tree. These states are sorted in a non-increasing order from left to right with respect to the cost of the Steiner tree. Replace each state in the sorted list with n copies of itself to get a total 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 an equally likely event. The probability of a transition from s_i to s_j is thus as follows:

$$P_{ij} = \frac{1}{i-1} \quad (1 \le j < i, P_{11} = 1) \tag{4.1}$$

The proof of time complexity of VNDMR is based on the method used in (Zhu *et al*., 1995). Let m_i be the number of transitions needed to go from state s_i to s_1, the expected value $E[m_i] = \log(i)$ (see Zhu *et al*., 1995). Therefore, if the VNSMR algorithm starts from the most expensive state, i.e. $i = n^{n-1}$, 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 neighbourhood structures in VNSMR is $O(n\log(n))$. The VNSMR algorithm includes three neighbourhood structures (N_1, N_2, N_3), thus the time complexity of VNDMR is:

$$O(n\log(n)(\ O(N_1)+\ O(N_2)+\ O(N_3)))$$

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

$$O(n\log(n)(3\times l\times l\log(n))) = O(l^2 n\log^2(n)) \tag{4.2}$$

For VNDMR2, the first neighbourhood structure N_1, the time consumed by each operation of constructing the Prim's spanning tree (Cormen *et al*., 1997) is $O(l\log(n))$. In the worst case, the maximum number of expected iterations from one spanning tree to another is $n\log(n)$ (Zhu *et al.*, 1995). So the time complexity of N_1 is:

$$O(N_1) = O(n\log(n)\times l\log(n)) = O(nl\log^2(n)) \tag{4.3}$$

The path-replacement operation of both N_2, and N_3 in VNDMR2 is based on the Dijkstra's shortest path algorithm which takes $O(l\log(n))$. Neighbourhood N_2, and N_3 require replacing at most $O(l)$ *superpaths* in the worst case. Therefore, the time complexity of VNDMR2 is:

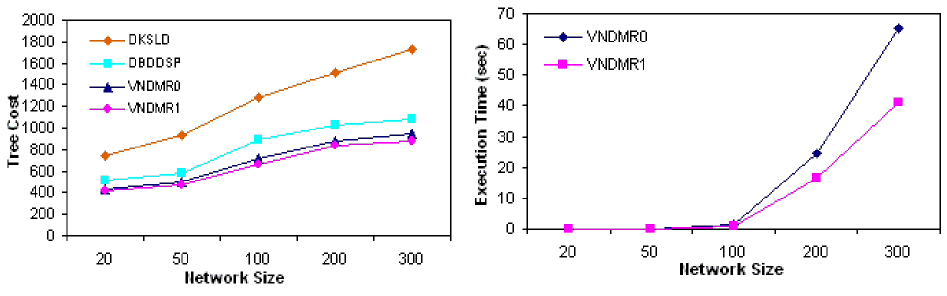
$$O(n\log(n)(\ nl\log^2(n)+2\times l\times l\log(n))) = O(nl\ \log^2(n)(n\log(n)+l)) \tag{4.4}$$

4.2.2 Experiments and Results

To evaluate the efficiency of the VNDMR algorithms, the multicast routing simulator MRSIM is used to generate a set of random network topologies. In the simulations, parameters are set as $\alpha = 0.25$, $\beta = 0.40$, and the average node degree = 4 in MRSIM (see Section 3.3.2 for more details). 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. VNDMR algorithms have been implemented and integrated in C++ in the simulator. A series of experiments are designed as follows to test the performance of the proposed VNDMR algorithms. All simulations were run on a Windows XP computer with PIV 3.4GHZ, 1G RAM.

4.2.2.1 VNDMR with Different Initializations

In the first group of experiments, 20 different network topologies for each size of networks with 20, 50, 100, 200 and 300 nodes are randomly generated. For each network topology, the source node and the destination nodes are randomly selected. The delay bound in the experiments for each network topology is set to be 2 times the delay of the least delay tree generated by DKSLD (see Section 4.2.1.1) , i.e. $\Delta = 2\times\ Delay(T_{DKSLD})$. The performance of VNDMR with two different initialization heuristics, e.g. VNDMR0 with DKSLD and VNDMR1 with DBDDSP (see Section 4.2.1.1), are investigated. Both variants have the same neighbourhood structures as defined in Section 4.2.1.2. The simulation was run 50 times on each network topology.



(a) tree cost (b) execution time

Figure 4. 4 Tree cost and execution time vs. network size (group size = 10)

Figure 4.4 presents the average tree cost and execution time of VNDMR0 and VNDMR1 for the problems of different network sizes with group size (the number of destinations) of 10. The tree cost of the initial solutions obtained from DBDDSP and DKSLD are also presented in Figure 4.4(a), where we can see that the initial solutions generated by DBDDSP are better than those generated by DKSLD and both VNDMR algorithms employing the three predefined neighbourhoods can further improve the initial solutions. Affected by different initializations, VNDMR1 (with initial solutions obtained from DBDDSP) performs better than VNDMR0 (with initial solutions obtained from DKSLD) in terms of both the average tree cost and computational time.

Table 4. 1 Tree cost vs. group size (network size = 50)

Group size	DKSLD	VNDMR0	DBDDSP	VNDMR1
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**

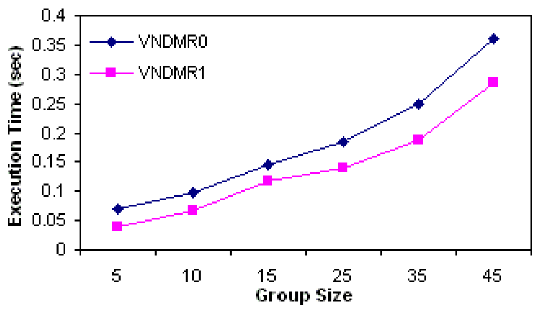


Figure 4. 5 Execution time vs. group size (network size = 50)

The above experiments show that the VNDMR algorithms can always improve the initial solutions when constructing the DCLC multicast trees. The quality of initial solutions affects the performance of the VNDMR algorithm. It shows that the better initial solution from a more intelligent heuristic leads to better final results, and also reduces the execution time of the VNDMR algorithm.

4.2.2.2 VNDMR with Different Neighbourhood Structures

In the second group of experiments, VNDMR1 and VNDMR2 are tested on the same random network topologies generated in Section 4.2.2.1. Both VNDMR1 and VNDMR2 start from the same initial solution from DBDDSP, whereas they apply different neighbourhood structures described in Section 4.2.1.2.

Table 4. 2 Average tree cost and execution time vs. network size (group size = 10)

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

The average tree cost and execution time of VNDMR1 and VNDMR2 on 5 different network sizes with group size of 10 are shown in Table 4.2. VNDMR2 always obtains better average tree cost than VNDMR1 on these networks of different sizes. It also shows that VNDMR2 spends longer computing time than VNDMR1.

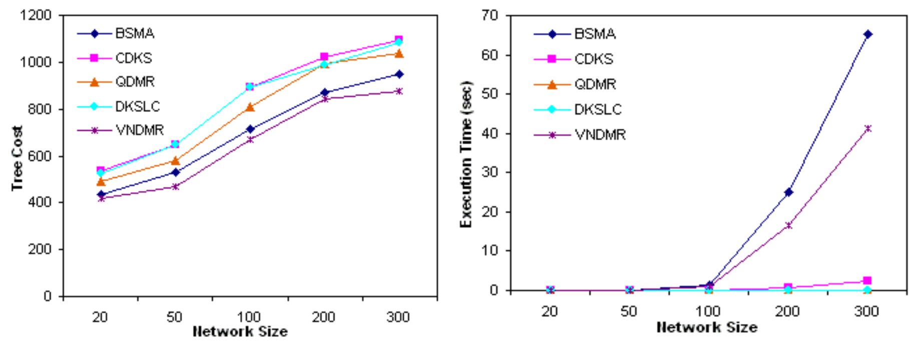
Table 4.3 presents the average tree cost and execution time of VNDMR1 and VNDMR2 on the same group of 50-node networks with different group sizes. We can see that VNDMR2 obtains better tree costs on the networks with small group sizes (5, 10 and 15 nodes), while VNDMR1 performs better than VNDMR2 when the group size increases (25, 35 and 45 nodes). It means the design of the neighbourhood structures affects the performance of the VNDMR algorithm. The ***Neighbour1*** of VNDMR2 is based on an operation on the nodes in the multicast tree. With the increasing group size (the number of destination nodes) in these 50-node networks, the possible neighbourhood trees of the current tree that can be explored are decreasing. Since the number of nodes which can be deleted from or added to the current tree is decreased, the ***Neighbour1*** plays no much role when exploring the solution space. However, the edge-based VNDMR1 still performs well even on the networks with large group sizes. On the other hand, VNDMR1 spends less computing time than VNDMR2 on the tested problems.

Table 4. 3 Average tree cost and execution time vs. group size (network size = 50)

Group size	**Initial Solution (DBDDSP)**	**VNDMR1**		**VNDMR2**	
		Cost	**Time (s)**	**Cost**	**Time (s)**
5	330.05	280.75	0.038	**280.15**	0.075
10	583.1	466.5	0.067	**456.85**	0.214
15	809.1	652.95	0.117	**643.65**	0.373
25	1077.75	**840.25**	0.141	845.15	0.473
35	1359.95	**1055.75**	0.187	1063.45	0.583
45	1591.45	**1214.75**	0.287	1224.95	0.595

4.2.2.3 Comparisons with Existing Algorithms

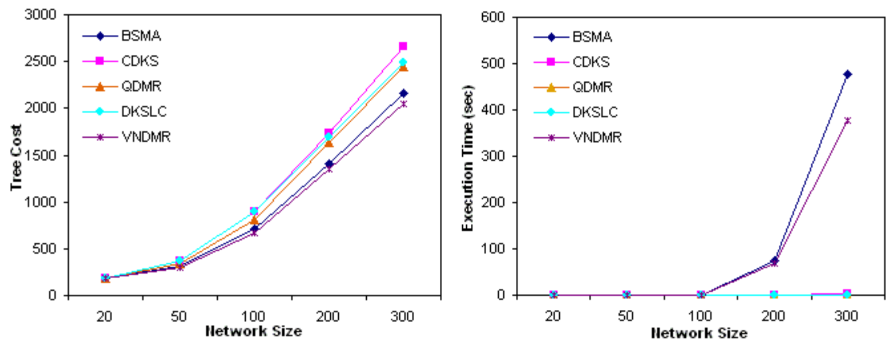
In the third group of experiments, VNDMR1 is compared with four other existing multicast routing algorithms in terms of both the solution quality and computational time using the same simulations designed in Section 4.2.2.1. The four algorithms include three DCLC multicast routing algorithms BSMA, CDKS and QDMR in the literature, and an algorithm named DKSLC which uses the Dijkstra's algorithm to construct the least cost multicast tree without the delay constraint.



(a) tree cost (b) execution time

Figure 4. 6 Tree cost and execution time vs. network size (group size = 10)

Figure 4.6 shows the average tree cost and execution time of these four algorithms compared with the VNDMR algorithms. It can be clearly seen in Figure 4.6(a) that VNDMR1 outperforms all the other four algorithms upon tree cost. CDKS and DKSLD have the worst and similar tree cost; BSMA is better than QDMR but worse on tree cost than VNDMR1. In addition, Figure 4.6(b) shows that VNDMR1 requires less execution time than that of BSMA. The other three algorithm CDKS, QDMR and DKSLC require a shorter computational time. However, the solution quality is of much lower quality than both BSMA and VNDMR1.



(a) tree cost (b) execution time

Figure 4. 7 Tree cost and execution time vs. network size (group size = 10% of network size)

Figure 4.7 presents the results of the VNDMR1 algorithm and other existing algorithms in terms of the tree cost and execution time with different network sizes, where the group sizes is 10% of the overall network sizes. Again, it can be seen in Figure 4.7(a) that VNDMR1 outperforms all the other four algorithms upon the solution quality. Figure 4.7(b) shows that VNDMR1 requires less execution time than BSMA. The time complexity of the VNDMR1 is $O(l^2n\log^2(n))$, while BSMA's time complexity is $O(kn^3\log(n))$ (n is the number of nodes in the network, l is the number of edges, k is the k-th shortest paths that are considered between the source and a destination).

Ghaboosi and Haghighat (2007) develop a path relinking algorithm which outperforms a number of existing algorithms, including KPP, BSMA, GA-based algorithms (Wang *et al*., 2001; Haghighat *et al*., 2004), Tabu search based algorithms (Wang *et al*., 2004; Ghaboosi and Haghighat, 2006; Skorin-Kapov and Kos, 2003; Youssef *et al*., 2001) and another path relinking algorithm for the Steiner tree problem (Bastos and Ribeiro, 2001), where the average tree costs of these algorithms on a set of random networks have been reported in (Ghaboosi and Haghighat, 2007), as shown in Table 4.4. In order to compare the proposed VNDMR algorithms with all these algorithms, the simulations are performed as those designed in (Ghaboosi and Haghighat, 2007) by generating a set of random network graphs which includes three random topologies for each network size (10, 20, 30, 40, 50, 60, 70, 80, 90 and 100 nodes). The link cost depends on the length of the link, and all the link delays are set to 1. The group size is set to 30% of the network size in each graph, the delay bounds are set to different values depending on the network sizes (Δ = 7 for network size 10-30, Δ = 8 for network size 40-60, Δ = 10 for network size 70-80 and Δ = 12 for network size 90-100). This group of random graphs will be used to test the algorithms proposed in the later sections in the thesis. To encourage scientific comparisons, the set of problem instances are publicly available at http://www.cs.nott.ac.uk~rxq/MRPresource.html for the benefits of the research community.

Table 4. 4 Average tree costs of the proposed VNDMR1, VNDMR2 and some existing heuristics and algorithms on the random networks of 10-100 nodes

Algorithms		**Average Tree Cost**
Heuristics	KPP1 (Kompella *et al*., 1993a)	905.581
	KPP2 (Kompella *et al*., 1993b)	911.684
	BSMA (Zhu *et al*., 1995)	872.681
GA-based Algorithms	Wang *et al*. (2001)	815.969
	Haghighat *et al*. (2004)	808.406
TS-based Algorithms	Skorin-Kapov and Kos (2003)	897.875
	Youssef *et al*. (2001)	854.839
	Wang *et al*. (2004)	869.291
	Ghaboosi and Haghighat (2006)	739.095
PR Algorithm	Ghaboosi and Haghighat (2007)	691.434
VNS Algorithms	The proposed VNDMR1	**680.067**
	The proposed VNDMR2 (Qu *et al*., 2009)	**672.470**

VNDMR2 is extended by initializing it from a greedy randomized initial solution. The randomized initial solution is generated by randomly choosing a starting destination node and connecting it with the source by using the Dijkstra's shortest path algorithm, then repeatedly connecting the unvisited destination nodes with the sub-tree until all destination nodes have been mounted to the tree. The simulation results are reported in Table 4.4 and Table 4.5.

Table 4. 5 Average tree cost, standard deviation of the tree cost and execution time of VNDMR1 and VNDMR2 on the random networks

Network Size	**VNDMR1**			**VNDMR2**		
	Tree Cost	σ	**Execution Time(s)**	**Tree Cost**	σ	**Execution Time(s)**
10	**94.67**	0	0.005	**94.67**	0	0.003
20	282.33	0	0.015	**275.33**	0	0.032
30	415.67	0	0.036	**405.93**	17.44	0.17
40	518	0	0.063	**513.87**	0.28	0.362
50	726.67	0	0.151	**713.5**	31.27	0.859
60	812.33	0	0.292	**790.27**	8.97	1.392
70	805.33	0	0.682	**803.97**	7.01	2.571
80	922.33	0	1.286	923.9	9.44	5.127
90	1182.67	0	3.151	**1182.6**	9.6	11.705
100	1040.67	0	4.292	**1020.67**	21.32	15.332

Table 4.4 shows that VNDMR2 performs the best in terms of the average tree cost. Note that in the literature all the other work has reported the average tree cost on all the tested instances and the comparisons of the average tree costs are thus presented in Table 4.4. Details of the average tree cost, standard deviation and execution time of VNDMR1 and VNDMR2 on each network size are given in Table 4.5. It can be seen that VNDMR2 produces better solutions on all the tested network sizes. It can be seen that VNDMR2 spends longer computing time than VNDMR1 to obtain the better results.

Figure 4.8 presents the execution time of the path relinking algorithm, VNDMR1 and VNDMR2. It can be seen that both VNDMR1 and VNDMR2 can obtain better results in a very short time (less then 5 seconds) which is much less than that of the path relinking algorithm. This indicates that Ghaboosi and Haghighat's path relinking algorithm is not practical for solving real-time network routing problems especially of very large and densely connected networks due to its high time complexity. This group of experiments show that VNDMR2 outperforms other existing heuristics and algorithms with regard to both the average tree cost and computational time.

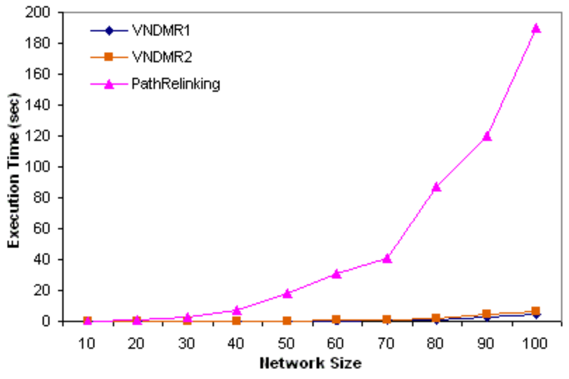


Figure 4. 8 Execution time vs. network size of VNDMR1, VNDMR2 and Ghaboosi and Haghighat's Path Relinking

In summary, through the above simulations on instances of different network sizes and different group sizes, variants of the VNDMR algorithm with different initial solution (VNDMR0, VNDMR1), and with different neighbourhood structures (VNDMR1, VNDMR2), have been evaluated. Experimental results show that both the initial solution and the neighbourhood structures will affect the performance of the VNDMR algorithm. Generally, with the same neighbourhood structures, better initial solution will help VNDMR to obtain better search results and well designed neighbourhood structures will contribute to better performance as well. Experimental results on a group of random graphs demonstrate that effectiveness and efficiency of the VNDMR algorithms in comparison with other existing heuristics and algorithms in terms of both the total tree cost and the execution time.

The research outcomes of the proposed VNDMR algorithms have been reported in (Qu *et al.*, 2009), and published in proceedings of Learning and Intelligent Optimisation (LION3), Lecture Notes in Computer Science, 5851. pp. 15-29.

4.3 A Hybrid GRASP Algorithm

GRASP is an efficient metaheuristic for optimisation problems. However, little attention has been given to GRASP for solving the QoS multicast routing problem. As reviewed in Chapter 3, only one GRASP algorithm (Skorin-Kapov and Kos, 2006) in the literature has been proposed for the DCLC multicast routing problem. Based on the previous variable neighbourhood descent search algorithm VNDMR (Qu *et al.*, 2009), a hybrid GRASP with VNDMR applied in the local search phase is thus investigated. Furthermore, motivated by the successful applications of the simple yet effective pilot method in the literature for the Steiner tree problem in Graphs in (Duin and Voß, 1999) and the network design problem in (Höller *et al.*, 2008), a pilot method is applied as a post optimisation procedure to further enhance the performance of the proposed algorithm. The aim is to investigate advanced hybrid metaheuristics for designing more efficient approaches to the DCLC multicast routing problem.

4.3.1 Algorithm Description

GRASP is a multi-start metaheuristic, where each iteration consists of two phases: a construction phase and a local search phase. The construction phase of GRASP aims to diversify the search by using certain random methods, while the local search phase in the GRASP intensifies the search by exploring the neighbourhoods of the current solution. The motivation is to investigate the hybrid GRASP approach, namely GRASP-VND+pilot, by applying the previous VNDMR algorithm in the local search phase and a pilot method as the intensification method for the DCLC multicast routing problem.

4.3.1.1 The Construction Phase

In the proposed GRASP algorithm, each solution, i.e. a multicast tree, is represented by a binary vector. Given a multicast routing problem in a network with $|V| = n$ nodes, the binary vector with n bits is defined, where each bit (from 0 to $n-1$) in the binary vector represents one node in the network, and takes a value of 1 if the corresponding node is in the multicast tree, 0 otherwise. This simple binary vector has been used for representing a multicast solution in (Skorin-Kapov and Kos, 2003; 2006). In the construction phase of the hybrid GRASP algorithm, the similar greedy strategy in (Skorin-Kapov and Kos, 2006) is used to create the randomized initial solution, a delay-constrained Steiner tree T, in the following three steps. The pseudo-code of the construction phase is given in Figure 4.9.

```
GreedyRandomSolution($G=(V, E), s, R, \Delta, \alpha$)
{   // $s$: source node; $R$: destination set; $\Delta \geq 0$ is the delay bound
    // $\alpha$: parameter to create the RCL (restricted candidate list)
    for all $r_i \in R$
        Calculate $ConnectPath[r_i]$ and $ConnectCost[r_i]$;
    $T = s$;
    while $R \not\subset T$ do {
        $BestConnectCost = \min\{ConnectCost[r_i],\ \forall r_i \in R \backslash T\}$;
        Create RCL of all $r_i \in R \backslash T$ where $ConnectCost[r_i] \leq \alpha \times BestConnectCost$;
        $r$ = Random(RCL, $Randseed$); //Randomly choose $r$ from RCL
        //$Randseed \in (0,1]$
        $T = T \cup ConnectPath[r]$;
        Update $ConnectPath[r_j]$ and $ConnectCost[r_j]$ for all $r_j \in R \backslash T$;
    } //end of while loop
    return $T$;
}
```

Figure 4. 9 The pseudo-code of the construction phase in the hybrid GRASP algorithm

(1) Starting from the source node, i.e. $T = \{s\}$, the shortest path which connects s and each destination is calculated by using the Dijkstra's shortest path algorithm. This path is denoted as $ConnectPath[r_i]$, for each destination node $r_i \in R \backslash T$. $ConnectPath[r_i]$ is set as the least cost path from s to r_i if the delay of the path satisfies the delay bound; otherwise, $ConnectPath[r_i]$ is set as the least delay path from s to r_i. $ConnectCost[r_i]$ denotes the cost of the path.

(2) The restricted candidate list (RCL) is created by choosing those nodes $r_i \in R \backslash T$, for which $ConnectCost[r_i] \leq \alpha \times BestConnectCost$, where $\alpha \geq 1$ is the parameter to control the size of the RCL, $BestConnectCost = \min\ \{ConnectCost[r_j],\ \forall r_j \in R \backslash T\}$. If $\alpha = 1$, then the algorithm is purely greedy. This means the RCL contains only the destination node with the least connect

cost. If $\alpha > 1$, the RCL includes the nodes with the path cost within the range of $\alpha \times$*BestConnectCost*.

(3) A destination node r is randomly chosen from the RCL and *ConnectPath*[r] is added to the tree T. *ConnectPath*[r_i] and *ConnectCost*[r_i] of the remaining unconnected destination nodes in $R \backslash T$ are updated by searching the new shortest path from these destination nodes to the current tree T. After that, the algorithm will return to step (2). The procedure ends until all the destination nodes are included in the tree.

In order to construct a high quality starting solution, a pure greedy method is used in the first iteration i.e. $\alpha = 1$. In the remaining iterations, $\alpha > 1$ is set to give more diversity to the construction phase for exploring the solution space.

4.3.1.2 The Local Search Phase

After a feasible solution is generated in the construction phase, a local search phase is applied based on VNDMR2 developed in (Qu *et al*., 2009) to improve the initial solution. VNDMR2 is based on a variable neighbourhood descent search algorithm and has shown to be effective to solve the DCLC multicast routing problem. VNDMR2 is thus hybridized in the local search phase. In VNDMR2, three neighbourhood structures are designed. The first neighbourhood N_1 is based on the nodes operation which generates a neighbouring solution by flipping a node which is not the source or a destination node in the current solution and then using the Prim's algorithm to create the minimum spanning tree of the given nodes. This process repeats until no better spanning tree can be found for 2 iterations. The other two neighbourhoods N_2 and N_3 operate on the paths by using a path replacement strategy based on the Dijkstra's shortest path algorithm to reduce the tree cost of the current solution and at the same time satisfy the delay bound, more details can be found in Section 4.2.

4.3.1.3 The Application of the Pilot Method

The pilot method may be seen as an intelligent technique to look ahead of possible choices by performing repetition of heuristics in order to record the best result before getting to the final decision of the options concerned. In the proposed GRASP-VND+pilot algorithm, a pilot method is designed to further explore better neighbouring solutions after a local optimum has been found by the local search phase. The framework of the GRASP-VND+pilot algorithm is illustrated in Figure 4.10. The pilot method is used as a post optimisation procedure, where each pilot includes the following two phases:

1) **Shaking process**: During the shaking process, a destination node is randomly selected. A set of back up paths which connects the source and the destination node is then generated for the chosen destination by using the *k*-th shortest path algorithm (Eppstein, 1998). The pilot shakes the current solution by choosing one path at random from the back up path set to replace the original path from the source to the destination node in the current tree.
2) **Pilot search procedure**: Each pilot searches better solutions by looking ahead more neighbouring solutions defined by the pilot heuristic in the pilot search procedure. Different pilot heuristic can be applied. The proposed pilot heuristic in this work uses the same operation as that defined in the neighbourhood structure N_1 in VNDMR2 which is easy to implement and has shown to be effective. The pilot search procedure repeats until no further improvement can be achieved.

```
GRASP-VND+pilot ($G=(V, E)$, $s$, $R$, $\Delta$, $Iter$, $\alpha$)
{ // $s$: source node; $R$: destination set; $\Delta \geq 0$: the delay bound;
  // $Iter$: the number of iterations; $\alpha$: parameter for creating greedy randomized solution;
  // $f$: the objective function; $t$: the pilot depth; $N$: the neighbourhood defined by the pilot method;
  if $Delay(r_i)$ of the Dijkstra's least delay path $p_T(s, r_i) > \Delta$, $\forall r_i \in R$, $p_T(s, r_i) \subseteq T(s, R)$;
  then return FAILED; // no feasible solution exists
  else
  {
   $i = 0$;
   while $i < Iter$ do {
          if $i == 0$ // pure greedy solution ($\alpha$=1) in the first iteration
             $T_{best}$ = GreedyRandomSolution($G$, $s$, $R$, $\Delta$, 1); //the construction phase, see Figure 4.9
          else // the remaining iterations of GRASP-VND ($\alpha$ >1)
             $T_0$= GreedyRandomSolution($G$, $s$, $R$, $\Delta$, $\alpha$); //the construction phase, see Figure 4.9
          $T$ = VNDMR2($G$, $s$, $R$, $\Delta$, $T_0$); // the local search phase
          // the pilot method in GRASP-VND+pilot
          Repeat the following sequence $t$ times:
              (a) Shaking.
                  Generate a neighbouring solution $T'$ from the neighbourhood $N(T)$ of $T$
              (b) The pilot search procedure.
                  Explore a new solution $T''$ of $T'$ defined by the pilot method
              (c) Move or not.
                  if ($f(T'')<f(T')$), set $T' \leftarrow T''$.
          if ($f(T') < f(T_{best})$)    then $T_{best} = T'$;
          $i$++;
   } //end of while loop
  }
  return $T_{best}$;
}
```

Figure 4. 10 The framework of the GRASP-VND+pilot algorithm

In a pilot method, a key factor is the pilot depth, which determines the actual search depth of the method, i.e. the quality of the results and the runtime of the search. In the pilot method, the pilot depth is a given number of iterations of the pilot heuristic. One main shortcoming of the pilot method is that it is associated with a higher time complexity. The pilot method with a larger pilot depth may lead to better solutions; on the other hand, it usually results in longer runtime. As shown in Figure 4.10, when the pilot depth $t = 0$, the pilot method is actually not applied in GRASP-VND+pilot. For this special case, the algorithm is called GRASP-VND, which is the combination of only GRASP and VNDMR2.

4.3.1.4 Time Complexity of the Proposed Algorithms

As described in Section 4.3.1.1, the construction phase in GRASP-VND and GRASP-VND+pilot generates the initial solution by using the Dijkstra's shortest path algorithm for $|R|$ destination nodes, so the time complexity of the construction phase is dominated by the Dijkstra's shortest path algorithm, which is $O(|R|l\log(n))$, where $n = |V|$ is the number of nodes and $l = |E|$ is the number of links in the network. For the proposed GRASP-VND and GRASP-VND+pilot algorithms, the local search phase is based on VNDMR2. In Section 4.2.1.4, the time complexity of VNDMR2 has been proven to be $O(nl\ \log^2(n)(n\log(n)+l))$. The time complexity of GRASP-VND with *Iter* iterations is thus as follows:

$$O(Iter\times(|R|l\log(n)+nl\log^2(n)(n\log(n)+l))) = O(Iter\times l\log(n)(|R|+n^2\log^2(n)+nl\log(n))) \qquad (4.5)$$

The pilot method in GRASP-VND+pilot repeats the pilot search procedure several times to find better neighbouring solutions of the incumbent solution after the shaking process. The search procedure is based on the same operation in the neighbourhood N_1 of VNDMR2, so the time complexity of the pilot method is $O(tnl\log^2(n))$, t is the pilot depth. Finally, the time complexity of the GRASP-VND+pilot algorithm can be drawn as follows:

$$O(Iter\times l\log(n)(|R|+n^2\log^2(n)+nl\log(n)+tn\log(n))) \qquad (4.6)$$

4.3.2 Experiments and Results

The proposed GRASP-VND+pilot, GRASP-VND and the GRASP-CST algorithm in (Skorin-Kapov and Kos, 2006) are implememnted. VNDMR2 is extended to a multi-start algorithm by running it for a fixed number of iterations, named Multi-VND. Each iteration of Multi-VND starts from a greedy random initial solution, which is generated by randomly choosing a starting destination node and connecting it with the source by using the Dijkstra's shortest path algorithm, then repeatedly connecting the unvisited destination nodes with the sub-tree until all destination nodes have been mounted on the tree.

To determine appropriate settings for the parameters in GRASP-VND+pilot and GRASP-VND algorithms, a number of initial tests were carried out. The aim was to obtain good quality solutions by using as less number of iterations as possible to reduce the execution time. Parameter α (see Figure 4.9) should be properly set to find a balance between diversification and intensification of the search over the search space. After a set of initial tests, the number of iterations is set as 4, α is set as 5 in both GRASP algorithms. The pilot depth is a key parameter in GRASP-VND+pilot which affects the quality of solutions and the computational time, here the pilot depth is set as $t = 5$. Parameters for GRASP-CST are set as the same as in (Skorin-Kapov and Kos, 2006), where the number of iterations is 5, α is 5, and the number of iterations without improvement of the local search procedure is 2. The number of iterations is set to 4 in Multi-VND.

4.3.2.1 Experiments on Benchmark Problems (Steinb) in the OR-library

Firstly, experiments are carried out on the benchmark instances Steinb in the OR-library. Characteristics of Steinb instances have been given in Section 3.5.4 (see Table 3.3). For the DCLC multicast routing problem, the delay bound is the key factor which affects the solutions obtained. The smaller the delay bound, the tighter the constraint is. Three types of different delay bounds are set on the Steinb instances to evaluate the performance of the proposed algorithm. The values of these delay bounds are described as follows:

- $\Delta_1 = \infty$ means the delay bound is set as a very large positive number for each instance in the experiments. The DCLC multicast routing problem is actually reduced to the unconstrained Steiner tree problem, since the delays of the links do not introduce any restriction in constructing the Steiner tree. The optimal solutions of this set of unconstrained multicast routing instances, i.e. Steiner tree problems, have already been obtained in the OR-library, shown in Table 3.3.

- $\Delta_2 = 1.1\times Delay(T_{OPT})$, where T_{OPT} denotes the multicast tree of the optimal solution for each unconstrained Steiner tree instance, and $Delay(T_{OPT})$ denotes the delay of the optimal solution. As the delay bound is set as greater than the delay of the optimal solution, the optimal solution to the unconstrained Steiner tree problem is thus the optimal solution to the DCLC multicast routing problem.
- For $\Delta_3 = 0.9\times Delay(T_{OPT})$, the delay bound is set as smaller than the delay of the optimal solution of each unconstrained Steiner tree instance, the optimal solution is thus not known to any of the DCLC multicast routing instance with this tighter delay bound.

Table 4. 6 Performance of GRASP-VND+pilot, GRASP-VND, Multi-VND and GRASP-CST for unconstrained Steiner tree problems ($\Delta_1 = \infty$). (The best values are in bold, the values marked with '*' denote the optimal solution)

No.	**GRASP-VND+pilot**				**GRASP-VND**				**Multi-VND**				**GRASP-CST**			
	Avg.	**Best**	σ	**Time(s)**	**Avg.**	**Best**	σ	**Time(s)**	**Avg.**	**Best**	σ	**Time(s)**	**Avg.**	**Best**	σ	**Time(s)**
B01	**82***	82*	0	0.62	**82***	82*	0	0.73	**82***	82*	0	0.29	**82***	82*	0	0.63
B02	**83***	83*	0	0.80	**83***	83*	0	1.04	**83***	83*	0	1.34	**83***	83*	0	0.80
B03	**138***	138*	0	1.28	**138***	138*	0	2.04	**138***	138*	0	1.96	**138***	138*	0	1.10
B04	**59***	59*	0	0.61	**59***	59*	0	0.88	**59***	59*	0	0.99	**59***	59*	0	0.68
B05	**61***	61*	0	0.85	**61***	61*	0	1.35	**61***	61*	0	1.66	**61***	61*	0	0.86
B06	**122***	122*	0	4.51	123.6	122*	1.17	3.10	124	124	0	3.53	123.1	122*	1.20	1.80
B07	**111***	111*	0	1.41	**111***	111*	0	2.11	**111***	111*	0	2.31	**111***	111*	0	1.86
B08	**104***	104*	0	3.19	**104***	104*	0	3.76	104.6	104*	1.27	4.21	**104***	104*	0	2.80
B09	**220***	220*	0	5.53	**220***	220*	0	9.34	**220***	220*	0	8.61	**220***	220*	0	4.92
B10	**86***	86*	0	2.59	87	86*	2.11	3.36	87	86*	2.11	4.94	86.5	86*	1.58	2.39
B11	**88***	88*	0	4.07	88.7	88*	0.95	3.83	89	88*	1.41	7.03	88.1	88*	0.32	2.93
B12	**174***	174*	0	10.02	**174***	174*	0	15.32	**174***	174*	0	14.13	**174***	174*	0	6.76
B13	**166.4**	165*	2.27	45.00	171.1	165*	2.23	11.10	167.4	165*	2.07	11.21	169.7	165*	2.75	6.98
B14	**235***	235*	0	11.36	235.5	235*	0.53	11.84	235.9	235*	0.32	31.50	235.4	235*	0.52	8.57
B15	**319.6**	318*	0.84	58.66	320	320	0	41.99	320	320	0	39.07	321.2	318*	1.40	16.56
B16	**127***	127*	0	22.31	**127***	127*	0	12.24	**127***	127*	0	14.24	128	127*	2.11	6.41
B17	**131***	131*	0	13.73	133.7	131*	0.95	14.14	131.3	131*	0.95	15.54	**131***	131*	0	10.3
B18	**218***	218*	0	28.38	218.2	218*	0.42	39.70	218.2	218*	0.42	33.47	218.4	218*	0.48	19.28

In the first group of experiments, the delay bound is set as $\Delta_1 = \infty$. The simulation was run 10 times on each instance. The average and best tree cost, the standard deviation of the results and the computing time of GRASP-VNS+pilot, compared with those of GRASP-VND, Multi-VND and GRASP-CST are illustrated in Table 4.6. Itcan be seen that the GRASP-VND+pilot algorithm gives the best solutions (marked in bold) on all the instances except one (B13) with respect to the average tree cost, while GRASP-VND, Multi-VND and GRASP-CST obtain 10 out of 18 best solutions. Both GRASP-VNS+pilot and GRASP-CST find the optimal solution at least once out of 10 runs for all 18 instances, which is better than GRASP-VND and Multi-VNS which find the optimal solutions on 17 and 16 instances in the 10 runs, respectively. The GRASP-VND+pilot algorithm with pilot method improves the solutions of GRASP-VND on 8 out of 18 instances. It should be noticed that the GRASP-VNS+pilot algorithm achieves the best results by consuming longer computing time than all the other algorithms.

In the second group of experiments, the delay bound is set as $\Delta_2 = 1.1 \times Delay(T_{OPT})$. With the tighter bounded end-to-end delay, similar results are obtained, as shown in Table 4.7. The GRASP-VND+pilot algorithm still performs the best among these four algorithms in terms of the average tree cost. It obtains the best solutions for all the 18 instances except one (B13), compared with those of GRASP-VND and Multi-VND (11 best solutions), and GRASP-CST (8 best solutions). The GRASP-VND+pilot algorithm improves the quality of solutions of GRASP-VND on 7 instances and even reduces the runtime on 10 out of 18 instances compared with GRASP-VND. Although GRASP-VND+pilot consumes longer computing time for some instances, it converges to the optimal solutions faster than the other three algorithms on four instances (B1, B2, B4 and B7) and leads to better solutions with lower average tree costs than those found by the other three algorithms in most cases.

Table 4. 7 Performance of GRASP-VND+pilot, GRASP-VND, Multi-VND and GRASP-CST for Steiner tree problems with $\Delta_2 = 1.1 \times Delay(T_{OPT})$

No.	Δ	**GRASP-VND+pilot**				**GRASP-VND**				**Multi-VND**				**GRASP-CST**			
		Avg.	**Best**	σ	**Time(s)**	**Avg.**	**Best**	σ	**Time(s)**	**Avg.**	**Best**	σ	**Time(s)**	**Avg.**	**Best**	σ	**Time(s)**
B01	145	**82***	82*	0	0.37	**82***	82*	0	0.51	**82***	82*	0	0.29	**82***	82*	0	0.54
B02	228	**83***	83*	0	0.79	**83***	83*	0	1.08	**83***	83*	0	1.41	**83***	83*	0	0.8
B03	248	**138***	138*	0	1.20	**138***	138*	0	2.15	**138***	138*	0	1.98	**138***	138*	0	1.09
B04	173	**59***	59*	0	0.59	**59***	59*	0	0.83	**59***	59*	0	1.02	**59***	59*	0	0.64
B05	125	**61***	61*	0	0.84	**61***	61*	0	1.37	**61***	61*	0	1.71	**61***	61*	0	0.81
B06	281	**122***	122*	0	4.58	123	122*	1.05	3.23	124	124	0	3.58	123	122*	1.05	1.73
B07	212	**111***	111*	0	1.43	**111***	111*	0	1.93	**111***	111*	0	2.37	**111***	111*	0	1.83
B08	209	**104***	104*	0	3.19	**104***	104*	0	4.20	104.9	104*	1.45	3.88	**104***	104*	0	2. 8
B09	280	**220***	220*	0	5.53	**220***	220*	0	9.77	**220***	220*	0	8.61	**220***	220*	0	4.87
B10	262	**86***	86*	0	2.65	86.5	86*	1.58	3.22	**86***	86*	0	5.12	87	86*	2.11	2.20
B11	235	**88***	88*	0	3.83	88.5	88*	1.08	4.48	88.2	88*	0.63	6.52	88.3	88*	0.67	2.92
B12	225	**174***	174*	0	18.44	**174***	174*	0	16.75	**174***	174*	0	12.25	175	174*	1.67	6.37
B13	190	170.1	165*	2.08	19.96	170.9	165*	2.23	11.27	**168.2**	165*	1.69	11.98	170.6	165*	2.37	6.67
B14	221	**235***	235*	0	22.33	235.2	235*	0.44	12.43	243.6	236	3.10	9.27	235.2	235*	0.41	8.19
B15	308	**319.7**	318*	0.71	45.98	320	320	0	39.52	320	320	0	38.96	319.8	318*	0.63	17.94
B16	291	**127***	127*	0	22.23	**127***	127*	0	12.79	**127***	127*	0	15.91	130	127*	2.58	6.12
B17	219	**131.5**	131*	0.53	34.84	133.1	132	1.45	15.78	131.7	131*	1.25	14.75	131.9	131*	0.32	9.64
B18	425	**218***	218*	0	57.65	**218***	218*	0	42.67	218.1	218*	0.31	34.07	218.1	218*	0.32	19.54

In the third group of experiments, the delay bound Δ_3 is set to a smaller value $0.9 \times Delay(T_{OPT})$. Table 4.8 shows that GRASP-VND+pilot obtains the best results on 12 instances upon the average tree cost. This is better than those of GRASP-VND (6 best results), Multi-VND (10 best results) and GRASP-CST (7 best results). GRASP-VND+pilot, GRASP-VND and GRASP-CST can not find feasible solutions on two instances (B03 and B07) due to the tighter delay bound. The simple multi-start Multi-VND algorithm has shown to be more robust than all the other algorithms by finding solutions for instances (B03 and B07), although it also fails to find a feasible solution for instance B14. GRASP-VND+pilot achieves better results than GRASP-VND on 7 out of 18 instances, showing again that the pilot method is able to improve the performance of GRASP-VND. With regards to the average computing time, GRASP-VND+pilot requires longer runtime than the other algorithms to get the better average tree costs.

Table 4.9 summarizes the average tree cost, standard deviation, and computational time of the four algorithms on the Steinb instances after the above three groups of experiments. Please note that for the third test, the average value is for the 15 instances that all the algorithms have obtained the feasible solutions. As can be seen in the table that although the four algorithms have competitive average performances in the three tests, GRASP-VND+pilot has the overall best performance among the four algorithms in terms of the average tree cost, although at the expense of a longer computational time. Besides, GRASP-VND+pilot algorithm is more stable than the other three algorithms with respect to the average standard deviation.

Table 4. 8 Performance of GRASP-VND+pilot, GRASP-VND, Multi-VND and GRASP-CST for Steiner tree problems with $\Delta_3 = 0.9 \times Delay(T_{OPT})$. ("\" denotes that no feasible solution was obtained)

No.	Δ	GRASP-VND+pilot				GRASP-VND				Multi-VND				GRASP-CST			
		Avg.	Best	σ	Time(s)	Avg.	Best	σ	Time(s)	Avg.	Best	σ	Time(s)	Avg.	Best	σ	Time(s)
B01	118	**83**	83	0	1.05	**83**	83	0	0.44	**83**	83	0	0.25	**83**	83	0	0.46
B02	187	**84**	84	0	1.60	**84**	84	0	0.68	**84**	84	0	0.69	**84**	84	0	0.63
B03	203	\	\	\	\	\	\	\	\	**141.7**	141	0.48	1.94	\	\	\	\
B04	142	**62**	62	0	1.70	63.9	62	4.38	0.83	64	64	0	1.19	62.6	62	1.26	0.65
B05	102	**62**	62	0	1.58	**62**	62	0	1.23	**62**	62	0	1.04	64.7	62	0.95	0.54
B06	199	125	125	0	6.66	125	125	0	3.63	125	125	0	3.56	**124**	124	0	1.58
B07	173	\	\	\	\	\	\	\	\	**112**	112	0	1.65	\	\	\	\
B08	171	**107**	107	0	8.16	**107**	107	0	4.34	**107**	107	0	3.42	**107**	107	0	2.59
B09	229	**221**	221	0	13.09	**221**	221	0	10.33	**221**	221	0	9.34	**221**	221	0	5.05
B10	215	88.9	88	1.45	6.32	89.8	88	1.55	3.19	**88.3**	88	0.95	4.60	**88.3**	88	0.95	2.12
B11	180	**89**	89	0	9.65	89.5	89	0.85	5.00	89.6	89	1.26	7.06	89.2	89	0.63	3.09
B12	184	179	177	2.83	18.07	178.4	175	2.56	15.38	**177.7**	177	1.16	13.68	178.5	177	4.24	6.37
B13	139	172	169	2	12.38	173	173	0	8.79	**169**	169	0	11.32	172	168	2.44	6.62
B14	180	**237.4**	237	0.89	20.45	237.9	237	1.05	12.41	\	\	\	\	242.1	237	1.85	6.83
B15	194	**322**	322	0	38.53	**322**	322	0	26.17	341	339	3.46	20.62	330.1	328	1.45	15.62
B16	238	**131.7**	129	0.95	15.71	**131.7**	129	0.95	8.51	132	132	0	16.15	**131.7**	129	0.95	5.46
B17	180	**134.2**	134	0.42	27.39	136.3	134	1.64	11.94	135	134	1.55	11.59	134.8	134	0.42	8.34
B18	348	**219**	219	0	47.44	219.2	219	0.42	40.11	219.8	219	0.42	36.55	219.6	219	0.52	18.55

Table 4. 9 Summary of average results for the three experiments on the Steinb in the OR-library

Test	GRASP-VND+pilot			GRASP-VND			Multi-VND			GRASP-CST		
	Cost	σ	Time(s)	Cost	σ	Time(s)	Cost	σ	Time(s)	Cost	σ	Time(s)
1	**140.278**	**0.173**	11.941	140.933	0.465	9.881	140.689	0.474	10.891	140.744	0.576	5.313
2	**140.517**	**0.184**	13.691	140.789	0.436	10.221	141.094	0.469	9.649	140.939	0.674	5.406
3	**138.653**	**0.510**	13.955	139.052	0.823	9.371	139.893	0.587	9.405	139.367	0.921	5.178

4.3.2.2 Experiments on Random Networks

In order to compare the GRASP-VND+pilot and GRASP-VND algorithms with other existing algorithms, the proposed algorithms are tested on a group of random networks which are the same as those generated in Section 4.2.2.3. Experiments were run10 times on each random network.

Table 4. 10 Average tree costs of GRASP-VND+pilot, GRASP-VND, Multi-VND, GRASP-CST and some existing heuristics and algorithms on the random networks with 10-100 nodes

Algorithms		Average Tree Cost
Heuristics	KPP1 (Kompella *et al.*, 1993a)	905.581
	KPP2 (Kompella *et al.*, 1993b)	911.684
	BSMA (Zhu *et al.*, 1995)	872.681
GA-based Algorithms	Wang *et al.* (2001)	815.969
	Haghighat *et al.* (2004)	808.406
TS-based Algorithms	Skorin-Kapov and Kos (2003)	897.875
	Youssef *et al.* (2001)	854.839
	Wang *et al.* (2004)	869.291
	Ghaboosi and Haghighat (2006)	739.095
PR Algorithm	Ghaboosi and Haghighat (2007)	691.434
VNS Algorithms	VNDMR1 (Qu *et al.*, 2009)	680.067
	VNDMR2 (Qu *et al.*, 2009)	672.470
	Multi-VND	656.777
GRASP Algorithms	GRASP-CST (Skorin-Kapov and Kos, 2006)	669.927
	GRASP-VND algorithm	654.520
	GRASP-VND+pilot algorithm	**650.823**

Table 4.10 shows that both GRASP-VND+pilot and GRASP-VND obtain better average tree costs than Multi-VND and GRASP-CST. Details of the average tree cost, standard deviation and execution time of these three algorithms on each network size are given in Table 4.11, showing that GRASP-VNS+pilot finds the best solutions on 8 out of 10 different network size instances, while GRASP-VND, Multi-VND and GRASP-CST obtain the best results on 2, 3 and 2 out of the 10 types of random networks, respectively. With respect to the average standard deviation σ, GRASP-VNS+pilot has lower σ (7.001) than GRASP-VND (8.38) and GRASP-CST (7.238), but higher than Multi-VND (6.566). GRASP-VND+pilot requires the longest average computing time (16.238 seconds) in comparison with GRASP-VND (7.092 seconds), Multi-VNS (7.674 seconds) and GRASP-CST (3.289 seconds) to improve the average tree costs

of GRASP-VND on 8 out of the 10 types of random networks. To summarize, the GRASP-VND+pilot algorithm performs the best compared with other existing algorithms with respect to the average tree cost as shown in Table 4.10 and Table 4.11. However, it consumes longer average computational time than the other algorithms as reported in Table 4.11.

Table 4. 11 Average tree cost, standard deviation of the tree cost and execution time of GRASP-VND+pilot, GRASP-VND, Multi-VND and GRASP-CST on the random networks

Network Size	**GRASP-VND+pilot**			**GRASP-VND**			**Multi-VND**			**GRASP-CST**		
	Cost	σ	**Time(s)**	**Cost**	σ	**Time(s)**	**Cost**	σ	**Time(s)**	**Cost**	σ	**Time(s)**
10	**94.67**	0	0.019	**94.67**	0	0.008	**94.67**	0	0.004	**94.67**	0	0.009
20	**271.13**	1.48	0.154	272.07	2.25	0.051	275.33	0	0.062	**271.13**	1.48	0.048
30	**392.33**	0	0.613	**392.33**	0	0.230	396.97	3.95	0.301	394.67	0	0.156
40	513.6	0.35	1.593	515.23	3.90	0.592	**513.4**	0.21	0.772	526.47	1.79	0.388
50	**663.87**	6.75	3.416	669.87	19.90	1.348	666.8	3.99	1.924	697.07	3.43	0.815
60	**754.81**	7.08	6.933	774.07	15.16	2.351	766.07	14.98	2.997	761.13	17.13	1.625
70	**780.13**	3.94	13.844	782	0	5.603	798.1	4.95	5.764	797.53	1.64	2.648
80	**878.9**	16.55	31.187	883.67	15.66	13.410	896.37	8.76	10.891	902.67	5.49	5.941
90	1163.77	28.67	52.202	1165.07	22.01	21.669	**1154.23**	23.37	22.932	1201.93	18.02	10.27
100	**995.03**	5.22	58.380	996.23	4.91	25.659	1005.83	5.46	31.091	1052	23.4	10.983
Avg.	**650.823**	**7.001**	16.834	654.520	8.380	7.092	656.777	6.566	7.674	669.927	7.238	3.289

The outcomes of this research on the hybrid GRASP algorithm have been published in the proceedings of the 4th Multidisplinary International Scheduling Conference (MISTA2009) (Xu and Qu, 2009).

4.4 A Jumping Particle Swarm Optimisation Algorithm

The population-based PSO (Kennedy and Eberhart, 1995) simulates simplified natural social systems such as flocks of birds or schools of fish. In PSO, a population (swarm) is made up of simple agents (particles) and evolves by following very simple rules in a descentralized way. Particles are typically modelled as entities moving in a multi-dimensional continuous space (search space) and interacting by sharing information about their positions (solutions) both locally and globally. For each particle *i* in the swarm at iteration *j*, a position (solution) $x_{i,j}$ and a velocity (rate of change) $v_{i,j}$ are updated in the evolution using the following two equations:

$$v_{i,j+1} = c_0 v_{i,j} + c_1 r_1 (b_i - x_{i,j}) + c_2 r_2 (g_j - x_{i,j}) + c_3 r_3 (g_{i,j} - x_{i,j}) \quad (4.7)$$

$$x_{i,j+1} = x_{i,j} + v_{i,j+1} \quad (4.8)$$

Eq. (4.7) handles the velocity update by summing up four components:

1) The first component $c_0v_{i,j}$ (inertia) enables the particle to keep the flow of its previous movement, avoiding abrupt moves and precipitate convergence;

2) The second component $c_1r_1(b_i - x_{i,j})$ encourages the particle's self-learning (cognition) ability by using its best position b_i achieved so far as a reference;

3) The third component $c_2r_2(g_j - x_{i,j})$ is the social factor that guides the particle to the best position g_j so far within the swarm, which keeps memory of the best performance so far of the entire swarm;

4) The fourth component $c_3r_3(g_{i,j} - x_{i,j})$ uses the best location in a neighbourhood sub-swarm $g_{i,j}$. This is to enhance the exploration capacity of the particles and also to prevent premature convergence within the swarm.

Each of the four addends in Eq. (4.7) is assigned a weight within $C = (c_0, c_1, c_2, c_3)$ to establish the importance of each component. Each of the coefficients in C is a preset real number in the range [0,1]. Moreover, in order to confer a stochastic behaviour, the latter three components are multiplied by random numbers $R = (r_1, r_2, r_3)$ drawn from a uniform [0, 1] distribution. Once the new velocity $v_{i,j+1}$ is calculated by using Eq. (4.7), Eq. (4.8) adds the new velocity to the current position $x_{i,j}$ so that the particle moves to a new position.

While the original PSO algorithm was designed for continuous optimisation problems, a variant called DPSO (Discrete Particle Swarm Optimisation) has been designed to deal with combinatorial optimisation problems. Since in DPSO, a discrete exploration of search space is more appropriate for combinatorial optimisation problems. The JPSO (Jumping Particle Swarm Optimisation, Moreno-Pérez *et al.*, 2007) algorithm is a recently introduced DPSO. One of the main advantages of JPSO is that it retains the simplicity of the original PSO but works on a discrete search space. This makes it a good option to tackle complex combinatorial optimisation problems. This motivated the work in this thesis to design a JPSO algorithm for solving the DCLC multicast routing problem and also the underlying Steiner tree problem.

Up to now, no previous investigation has been carried out on using JPSO to tackle both problems. Most related papers in the literature have mainly focused on one of them. On the DCLC multicast routing problem, the JPSO algorithm is compared against the best known results reported in the literature, both on solution quality and computational time. On the Steiner tree problem, the efficiency and effectiveness of the JPSO algorithm are evaluated by calculating the exact gap to the global optimal solutions known for the benchmark datasets used. A large number of experiments and simulations demonstrate that the JPSO algorithm obtains highly competitive results for both the DCLC multicast routing problems and the Steiner tree problems on some benchmark instances.

4.4.1 Algorithm Description

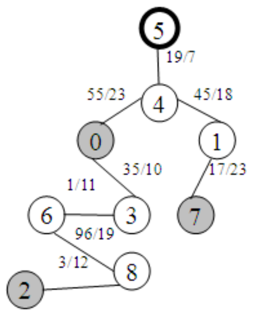
Without using the velocity to define how the swarm of particles move in the search space, JPSO applies a simplified attractor-follower scheme to allow a number of elements (particles) to move (jump) from position to position (solution) in the discrete space. If there is a particle with a good fitness in a certain region of the space, the other particles in the swarm will follow its position in order to improve themselves. For a particle *i*, there are three attractors, the first attractor b_i is the best position that particle *i* has achieved so far, the second attractor g_j is the best position of the swarm so far and the third attractor $g_{i,j}$ is the best position found by the sub-swarm made up by the neighbours of particle *i* at iteration *j*. JPSO defines four types of moves including the inertial move and other three moves towards the attractors, all of them have a likelihood c_x given by the weight vector where $\sum_{x=0}^{3} C_x = 1$. In the case of the inertial move, a mutation or neighbourhood move operator can be employed to explore the current position and to prevent premature convergence. For the three follower-attractor moves, crossover operators are used to just partially imitate the structure of the attractors by the follower.

At each generation, a random number *r* uniformly distributed in [0, 1] is generated. This number *r* along with the weights c_x determines how a particle moves.

- If $r \in [0, c_0)$, the inertial move will be triggered.
- If $r \in [c_0, c_0+c_1)$, the cognitive move, the attractor will be the best positioned particle in its neighbourhood at the current generation.
- If $r \in [c_0+c_1, c_0+c_1+c_2)$, the social move, the attractor will be the global best position in the swarm so far.
- If r ∈ [c0+c1+c2, c0+c1+c2+c3 = 1], the local move, the particle will try to move towards its own best position so far.

Once the particle's position x_{ij} has been updated, the JPSO algorithm conducts a local search to improve the quality of the solution represented by that position.

4.4.1.1 Initialization



predecessor:	4	4	8	0	5	x	3	1	6
node:	0	1	2	3	4	5	6	7	8

Figure 4. 11 An illustrative example of the representation of the multicast tree with 3 *superpath*s: (4-0-3-6-8-2), (5-4) and (4-1-7)

In the implementation, the swarm consists of |*P*| particles. Each particle in the swarm starts with a random initial position, i.e. a randomized multicast tree, which is generated by starting from the source node and randomly selecting the next link which connects any on-tree node in the tree until all destination nodes have been mounted on the tree and non-destination leaves will be pruned from the tree. As defined in the previous VNDMR algorithm (Qu *et al.*, 2009), the proposed JPSO algorithm uses the same predecessor array to represent a multicast tree and operates on the *superpath* in the inertial move to replace it by a new path. Figure 4.11 presents an illustrative example of the representation of the tree and the *superpath*.

4.4.1.2 Moves of Particles in the Swarm

As described above, each particle in the swarm has four types of moves depending on where the random number *r* falls within the interval [0, 1]. In the proposed JPSO algorithm, these moves are defined as follows:

- If $r \in [0, c_0)$, the inertial move, the particle moves around its position. A particle in this scenario will randomly delete a *superpath* in its current tree. The output of this operation is two sub-trees that will be reconnected by using a new random path.
- For the three follower-attractor moves, if $r \in [c_0, c_0+c_1)$, the best located particle in the neighbourhood of the current particle ($g_{i,j}$) acts as the attractor, if $r \in [c_0+c_1, c_0+c_1+c_2)$, the attractor is the best position achieved by the whole swarm so far (g_j), if $r \in [c_0+c_1+c_2, 1]$, the attractor is the best position achieved by the current particle so far (b_i). Once the attractor is selected, a path replacement operator is carried out by copying the cheapest path along the source to one destination from the attractor to the follower (the current particle's position). If the copied path is already in the multicast tree which represents the follower's position, a random move is applied to the follower particle. The movements of the particles are implemented by deleting the predecessor of the nodes in the original path to be replaced in the follower particle and updating the predecessor of the nodes by the new chosen path.

After each move, a local search algorithm is applied to improve the new particle's position. Finally, the particle finishes its jump by updating its best position b_i. A new position replaces b_i and/or g_j if that new position has smaller tree cost within the delay bound or has smaller tree delay with the same tree cost.

In the local search algorithm implemented here, the neighbourhood structures are defined based on the nodes of the multicast tree. One neighbour of the current tree is obtained by deleting or adding a non-destination node and creating a new spanning tree of the remaining nodes by using the Prim's spanning tree algorithm (Betsekas and Gallager, 1992). Two variants of local search approaches have been evaluated: one is a greedy approach which explores all neighbours of the current tree and then selects the best one; the other approach is a first improvement variant in which the first better neighbour found is selected.

4.4.1.3 The Path Replacement Operator

The path replacement operator used to update a particle's position with that of an attractor in the three follower-attractor moves is illustrated as follows. Figure 4.12 shows three trees corresponding to a follower particle T_0, an attractor particle T_a and the new generated position T'. The path replacement is carried out as follows. The path (5-4-0-3-6-8-2) in the current position T_0 is replaced by the path (5-4-1-7-8-2) from the tree corresponding to the attractor T_a. Thus a new tree T' is generated, which becomes the new position of the follower particle. The delay bound in this example is $\Delta = 82$.

Current tree T_0	Attractor tree T_a	New tree T'
$Cost(T_0) = 271$	$Cost(T_a) = 216$	$Cost(T') = 230$
$Delay(T_0) = 82$	$Delay(T_a) = 75$	$Delay(T') = 75$

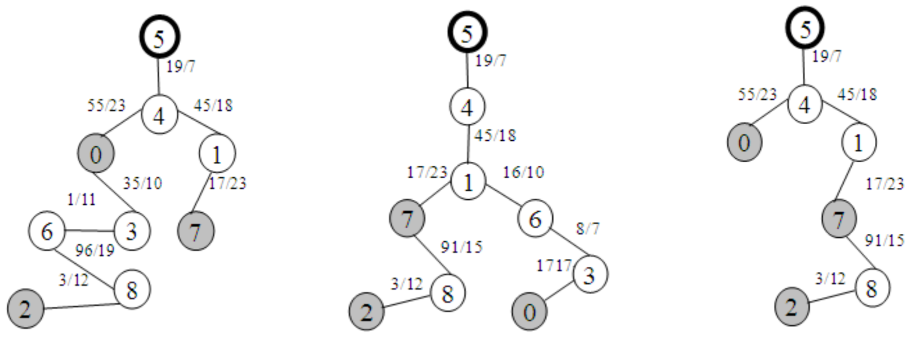


Figure 4. 12 An example of the path replacement operator
("x/y" denotes "cost/delay" of the links in the tree. Shaded nodes are destination nodes)

The pseudo-code below describes the implemented JPSO algorithm. A swarm of |*P*| particles is created with random positions. For a number of iterations, each particle (p.n) makes a move as explained above, either based on its current position (the inertial move) or by copying part of the attractor's position (the cognitive, social or local move). Once the particle has jumped to a new position, the local search (greedy or first improvement) is applied. The particle's best position (p.b) and the swarm's best position (g) are updated. After a predefined number of generations, the best position (g) achieved by the swarm is returned as the final solution.

JPSO Pseudo-code

```
JPSOMR($G = (V, E)$, $s$, $R$, $\Delta$, $|P|$, $c_0$, $c_1$, $c_2$, $c_3$)
{   // $s$: source node; $R$: destination set; $\Delta \geq 0$: the delay bound;
    //$|P|$: number of particles in the swarm.
    // p.n: the new position of the particle
    // p.c: the current position of the particle
    // p.bn: the best position in the neighbour particles of the current particle
    // g: the global best position found by the swarm
    // p.b: the best position found by the own particle

    Create $|P|$ randomized initial feasible solutions for all particles in the swarm;
    for (TotalIterations) do
        for each(p in the swarm) do
            generate a random number $r$ within the range (0, 1]; // $r \in [0, 1]$
            case ($r$) {
                 r is in c_0: p.n = RandomMove(p.c);              // Inertial move
                 r is in c_1: p.bn = GetBestNeighbourhood(p);
                        p.n = PathReplacement(p.bn, p.c);// Cognitive move
                 r is in c_2: p.n = PathReplacement(g, p.c);      // Social move
                 r is in c_3: p.n = PathReplacement(p.b, p.c);    // Local move
            }
            Greedy or first improvement local search on the current particle p.n;
            Calculate the Cost and Delay of the current particle p.n;
            if (((Cost(p.n) < Cost(p.b)) && (Delay(p.n) < Δ)) || (Cost(p.n) == Cost(p.b)) &&
            (Delay(p.n) < Delay(p.b)))
                then p.b = p.n;
                if (((Cost(p.n) < Cost(g)) && (Delay(p.n) < Δ)) || (Cost(p.n) == Cost(g)) &&
                (Delay(p.n) < Delay(g)))
                    then g = p.n;
                end if
            end if
            p.c = p.n;
        end for
    end for
    return g;
}
```

RandomMove(): a procedure that moves the particle to a new position by randomly choosing a *superpath* in the current position and replace it by a random new path;

GetBestNeighbourhood(): a procedure that picks the best position among the neighbourhood particles of the current particle;

PathReplacement(*attractor, follower*): a function that replaces the path from source to a destination in the follower by choosing the best path in the attractor.

4.4.2 Experiments and Results

The simulator MRSIM is used to implement and evaluate the proposed JPSO algorithm. JPSO is then compared with some related algorithms in the literature. Firstly, to determine the proper parameter settings within JPSO, a number of initial tests have been carried out. Then, the JPSO algorithm on two sets of benchmark Steiner tree problems (Steinb and Steinc) selected from the OR-library is tested. Details of Steinb and Steinc instances have been given in Table 3.3 and Table 3.4, respectively, in Section 3.4.4. Finally, the proposed JPSO is compared with some existing algorithms in the literature on a set of random networks. These random networks are the same as those generated in the previous experiments for testing the VNDMR algorithm and the GRASP algorithm. All simulations were run 30 times on a Windows XP PC with an Intel Core 2 Duo E8500 3.16GHZ processor and 8GB RAM for the small Steinb instances and 20 times on the large Steinc instances.

4.4.2.1 Parameter Tuning within the JPSO

First, a series of extensive experiments is conducted to tune the parameters and understand the behaviour of the JPSO algorithm with respect to the size of the swarm, the cooperation between particles, the effect of the local search, and the evolution of the swarm.

1) Effect of the Swarm's Size

In the first group of experiments, the JPSO is evaluated with 6 different swarm sizes (|*P*| = 1, 2, 5, 10, 20, and 30) with $c_0 = c_1 = c_2 = c_3 = 0.25$ and a maximum number of iterations equals to 100. No local search is integrated to have a clear view of an appropriate swarm size. The average tree costs and computational time are presented in Table 4.12 and Figure 4.13. The performance of JPSO with small particle numbers (1, 2, and 5) is not stable with large standard deviations. Better and stable solutions are obtained by JPSO with swarm size of 20 and 30, but with an increased computational time.

Table 4. 12 JPSO with different swarm sizes and no local search ('OPT' denotes the optimal solution and values in bold are the best results obtained)

No.	**OPT**	**|*P*| = 1**		**|*P*| = 2**		**|*P*| = 5**		**|*P*| = 10**		**|*P*| = 20**		**|*P*| = 30**	
		Cost	σ	**Cost**	σ	**Cost**	σ	**Cost**	σ	**Cost**	σ	**Cost**	σ
B01	**82**	96.1	18.633	86.3	6.473	**82**	**0**	**82**	**0**	**82**	**0**	**82**	**0**
B02	**83**	117.9	20.331	97.6	8.829	92.4	5.696	91.2	3.751	**90**	**0**	**90**	**0**
B03	**138**	174.6	11.972	164.5	9.192	161.6	6.465	158.5	0.707	**153.2**	**6.017**	154.3	4.041
B04	**59**	99.9	18.291	88.8	6.161	84. 9	3.621	81.6	4.453	77	3.625	**74.7**	**4.31**
B05	**61**	86.3	17.296	76.2	10.028	67.1	3.086	66.5	3	64.8	1.251	**64**	**0.816**
B06	**122**	183.1	27.416	157.4	6.841	151.8	5.973	146.8	3.488	145.3	4.619	**143.3**	**4.424**
B07	**111**	144.6	24.859	136.9	14.855	115.3	4.031	119.4	3.745	113.8	1.5	**112.8**	**1.5**
B08	**104**	147.5	26.818	124.2	12.667	123.6	11.908	113.4	3.779	**112.8**	**2.152**	113.5	2.258
B09	**220**	253.6	18.155	243.2	11.586	232.2	2.049	233.5	1.975	231	1.732	**230**	**0**
B10	**86**	162.8	32.034	135.3	22.949	138	16.294	123.6	4.333	**122.5**	**0.837**	**122.5**	**1.225**
B11	**88**	177.1	36.022	154	17.247	147.9	10.548	140	3.286	137.5	1.732	**133. 9**	**3.887**
B12	**174**	280.3	35.972	251.7	28.7	233.6	12.986	229.2	7.374	**224.2**	**3.189**	226.3	2.363
B13	**165**	223.7	25.027	206.8	20.359	183.4	8.385	179.6	5.505	176.4	7.537	**173.4**	**2.608**
B14	**235**	286.4	9.064	281	8.897	275.1	5.216	271.8	5.268	268.3	4.414	**265.4**	**7.468**
B15	**318**	372.9	16.72	364.9	14.024	353.6	3.13	352.2	2.041	349.8	2.95	**348. 7**	**3.786**
B16	**127**	225.5	45.604	185.6	26.957	180.6	19.034	162	0	**160.8**	**2.858**	162	0
B17	**131**	243.4	67.341	192	65.776	152.4	14.775	143	2.236	**143**	**0.707**	**143**	**0**
B18	**218**	344.9	57.356	308.6	42.863	272. 3	3.055	272.6	3.503	269	0	**268.3**	**1.155**

Statistically, the paired t-test value of the difference between JPSO with |*P*| = 10 and |*P*| = 20 is 6.02 which is greater than 3.97 for p= 0.001 in the T-table, showing JPSO with |*P*| = 20 is significantly better than that of with |*P*| = 10. The same is done for JPSO with |*P*| = 20 and |*P*| = 30, and a value of 2 is obtained, which is smaller than 2.11 for *p* = 0.05 in the T-table, showing JPSO with |*P*| = 20 and |*P*| = 30 has no significant difference. While the computational time of JPSO with |*P*| = 20 is much less than that of |*P*| = 30, |*P*| = 20 is therefore a good swarm size here for JPSO. More particles may be helpful for obtaining good results; however, a too large swarm actually hinders the algorithm's performance sometimes.

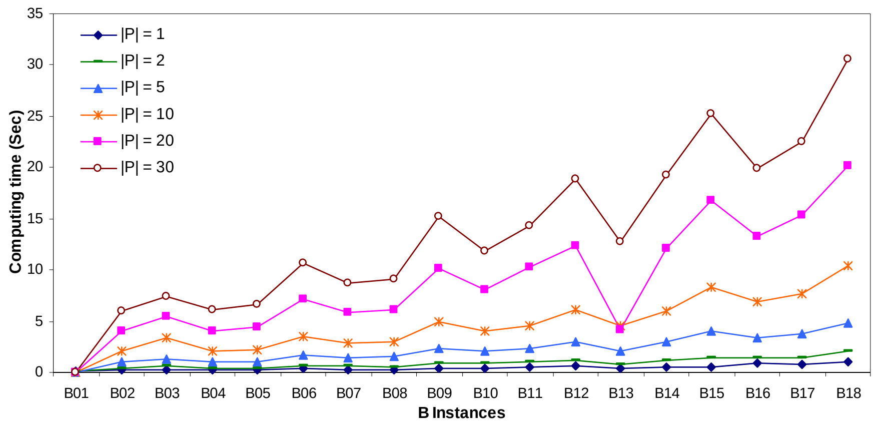


Figure 4. 13 Computational time of JPSO with different swarm sizes without local search on small Steinb instances

2) Effect of Cooperation between Particles

A key feature of JPSO is that during the search, particles in the swarm cooperate by sharing information about better positions in order to explore the search space effectively. The second set of experiments was conducted to find out if this is actually happening in JPSO. Recall that the values in the weight vector *C* = (c_0, c_1, c_2, c_3) determine the extent of a particle's movement towards the position based on its own behaviour or that of the other particles in the swarm. For example, setting *C* = (1, 0, 0, 0) means that the particles do not cooperate as they make their next move based only on their own current positions. This is equivalent to having a multi-start approach because each particle carries out its very own search by inertial movements followed by the local search. When c_0, c_1, c_2, and c_3 take values different from zero, particles cooperate by using both inertial and attractor movements in a true JPSO process.

The number of particles is set as |*P*| = 20, with two different weight vectors ($c_0 = 1$, $c_1 = c_2 = c_3 = 0$) and ($c_0 = c_1 = c_2 = c_3 = 0.25$), and compared JPSO with or without local search strategies (greedy and first improvement) described in Section 4.4.1.2. The average tree costs from the computational experiments on the 18 instances in the OR-library are given in Table 4.13.

Table 4. 13 JPSO using different settings of cooperation between particles and local search ('OPT' denotes the optimal solution and values in bold report the best results for each instance, each of the four columns reports results from a combination of cooperation ($c_x = 0.25$) or no cooperation ($c_0 = 1$) with the greedy or first improvement (non-greedy) local search)

$\|P\| = 20$		Average Tree Cost				$\|P\| = 20$		Average Tree Cost			
		$c_0 = 1$		$c_x = 0.25$				$c_0 = 1$		$c_x = 0.25$	
No.	OPT	Non-Greedy	Greedy	Non-Greedy	Greedy	No.	OPT	Non-Greedy	Greedy	Non-Greedy	Greedy
B01	82	**82**	**82**	**82**	**82**	B10	86	**86**	**86**	**86**	**86**
B02	83	87.5	**85**	88	85.5	B11	88	88.2	**88**	**88**	**88**
B03	138	**138**	**138**	**138**	**138**	B12	174	**174**	**174**	**174**	**174**
B04	59	**59**	**59**	**59**	**59**	B13	165	169	**165**	174	**165**
B05	61	**61**	**61**	61.4	**61**	B14	235	236	235.9	235.8	**235.2**
B06	122	**122**	122.4	**122**	**122**	B15	318	321.8	320.8	321	**319.7**
B07	111	**111**	**111**	**111**	**111**	B16	127	135.1	**127**	129	**127**
B08	104	104.8	**104**	**104**	**104**	B17	131	134	**131.1**	131.2	131.2
B09	220	**220**	**220**	**220**	**220**	B18	218	**218**	**218**	**218**	**218**

It is obvious that JPSO with the two local search strategies achieves much better results than that of JPSO without local search procedure (These results thus are not presented in Table 4.13). We can see that the JPSO with cooperation and the greedy local search has the overall best performance compared with the JPSO with other settings. For the small instances (B01 to B12), there is practically no significant difference between the performances of the JPSO with different settings. However, for the large instances (B13 to B17), cooperation helps the JPSO to achieve better results than its multi-start variant ($c_0 = 1$, $c_1 = c_2 = c_3 = 0$). Within the results of the JPSO ($c_x = 0.25$), there is a subtle difference in the performance on instances B14 and B15 depending on the type of local search. The results demonstrate that both the local search and cooperation between particles play important roles within JPSO, i.e. multi-start of the local search does not perform as well as JPSO with the same local search, especially for larger instances; JPSO with local search outperforms that of without local search.

Figures 4.14 and 4.15 show the computational time spent by variants of JPSO algorithms using different parameter settings of cooperation and local search on the small and large problem instances, respectively. In both figures, variances between different runs are small. Figure 4.14 shows that with the exception of instance B02, the four settings spend comparable computational time. However, it is intriguing that on instance B02, only cooperation between particles and greedy local search helps the algorithm to reduce the excessive execution time. A detailed examination into the JPSO performance with other settings shows the reason is the search was trapped in good local optima, so JPSO had to continue to evolve until the global optimal solution was obtained.

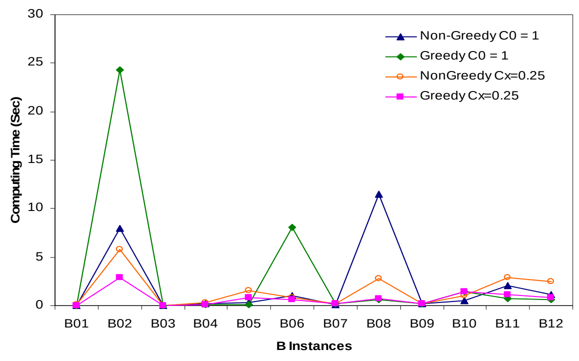


Figure 4. 14 Computational time of variants of JPSO on small instances

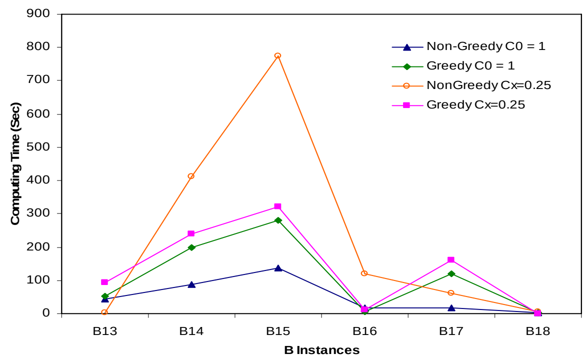


Figure 4. 15 Computational time of variants of JPSO on large instances

Results with respect to computational time are less clear on the large instances in Figure 4.15. It can be seen that the best results achieved with respect to the average tree cost (reported in Table 4.13 on the large instances) do no imply that more computational effort is required by the cooperation between particles, as two exceptions can be found on instances B13 and B17. Note however, that when no cooperation is allowed, JPSO with the greedy local search seems to be slower than that of JPSO with the first improvement local search for the large instances.

3) Evolution of the Swarm

In this set of experiments, the evolution of the swarm across iterations on problem instances of different sizes is investigated. The evolution of the best particle in the swarm is compared with respect to the different settings of the JPSO on three instances: one of network size 75 (instance steinb12, Figure 4.16), one of network size 100 (instance steinb16, Figure 4.17) and one of network size 500 (instance steinc01, Figure 4.18).

Figure 4.16 shows that instance steinb12 is relatively easy no matter the settings in the algorithm, although the optimum is found faster with greedy local search. On the medium size instance (steinb16), Figure 4.17 shows that although the JPSO reaches the optimum with different settings, the variant with cooperation and greedy local search is the fastest while the variant with no cooperation and greedy local search is the slowest. For the large instance (steinc01), Figure 4.18 shows that JPSO with cooperation and greedy local search is the first one that converges to the optimum, in this case the slowest is JPSO with cooperation and non-greedy local search. We can see that particularly in the large instance, greedy local search is beneficial both for the variants with and without cooperation.

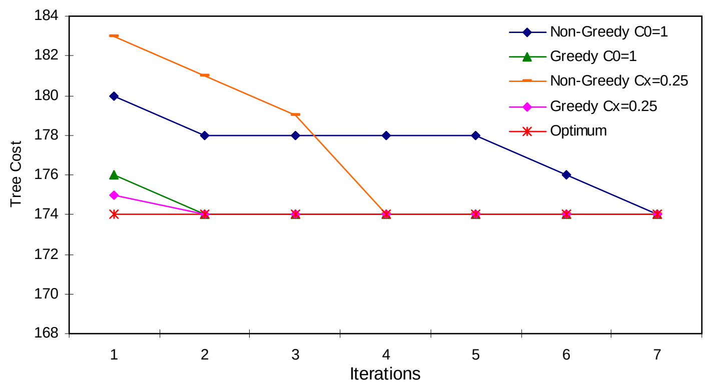


Figure 4. 16 Evolution across iterations of JPSO with $|P| = 5$, different settings of cooperation and local search on a Steiner tree problem of small size (network size = 75, instance steinb12)

Instance steinb12 (Figure 4.16) seems to be an easy problem so the local search does much of the work and the cooperative nature of the JPSO does less. However, when the first improvement local search is used, there is a slower convergence without the cooperation. Instance steinb16 (Figure 4.17) is harder to solve and it is clear that cooperation within the swarm helps to converge to the optimum. Instance steinc01 (Figure 4.18) is even harder and then greedy local search is clearly beneficial. From the three figures, we can see that the local search will greatly affect the performance of the JPSO algorithm. No matter what cooperation between particles in the swarm is applied, the JPSO algorithm with greedy local search will converge faster than the JPSO with non-greedy local search. On the other hand, the JPSO algorithm with c_x = 0.25 obtains the optimal solution faster than the JPSO algorithm with $c_0 = 1$ and $c_1 = c_2 = c_3 = 0$ in Figure 4.17. It means that if given more possibilities of moving towards the better positions (c_x = 0.25), the JPSO algorithm performs better than that with just random moves ($c_0 = 1$, $c_1 = c_2 = c_3 = 0$) in the swarm. It means the particles cooperation is also important to the performance of the JPSO algorithm.

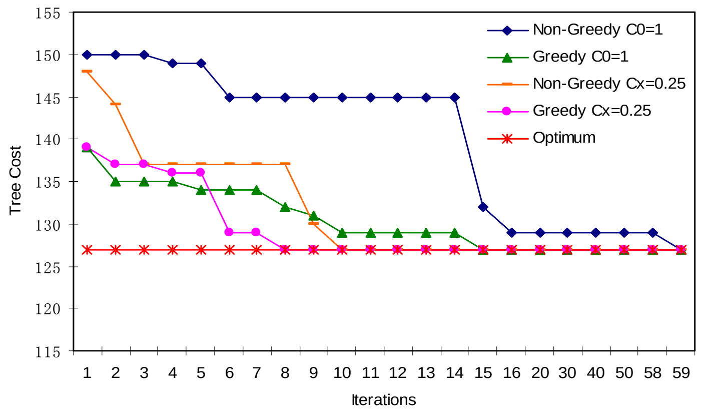


Figure 4. 17 Evolution across iterations of JPSO with |*P*| = 20, different settings of cooperation and local search on a Steiner tree problem of medium size (network size = 100, instance steinb16)

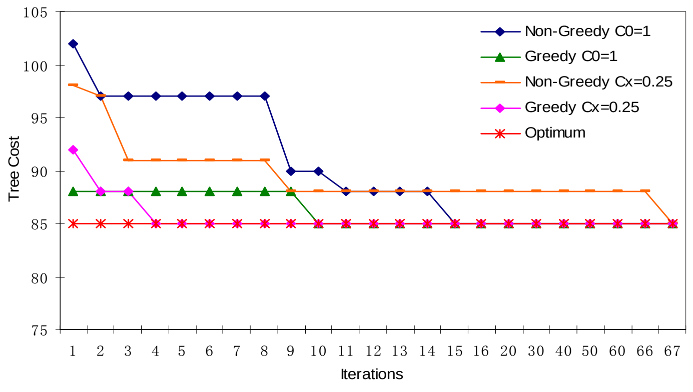


Figure 4. 18 Evolution across iterations of JPSO with |P| = 20, different settings of cooperation and local search on a Steiner tree problem of large size (network size = 500, instance steinc01)

4.4.2.2 Experimental Results on the Steiner Tree Problems

After tuning the parameters and studying the behaviour of the proposed JPSO in the previous subsections, extensive experiments have been conducted to compare the overall performance of the proposed JPSO to other recent algorithms in the literature on the Steiner tree problems. The parameters are set as ($c_0 = c_1 = c_2 = c_3 = 0.25$), $|P| = 20$ and the greedy local search is used. The number of iterations is 100 for Steinb instances and 10 for larger and harder Steinc instances. 10 independent runs have been executed for each algorithm on each instance.The JPSO is firstly compared with the GRASP-CST algorithm by Skorin-Kapov and Kos (2006) on the Steinb instances. To obtain comparable results,GRASP-CST has been re-implemented with the same parameters as that in Skorin-Kapov and Kos (2006), i.e. the number of iterations is 5, α is 5, and the number of iterations without improvement of the local search procedure is 2.

Table 4.14 shows the average and best tree costs, and the average computational time on the Steinb instances. The proposed JPSO algorithm performs better than GRASP-CST on 7 out of the 18 instances, while GRASP-CST obtains slightly better average tree costs than those obtained by JPSO on 2 instances (B02 and B17). It should be noticed that the JPSO achieves better results by consuming more computing time in most cases.

Table 4. 14 Comparison between JPSO and GRASP-CST approaches on Steinb problems (Values in the columns ('OPT') and in bold report the optimal solutions for each instance)

No.	OPT	JPSO			GRASP-CST			No.	OPT	JPSO			GRASP-CST		
		Mean	Best	Time(s)	Mean	Best	Time(s)			Mean	Best	Time(s)	Mean	Best	Time(s)
B01	**82**	**82**	82	0.002	**82**	82	0.086	**B10**	**86**	**86**	86	1.469	86	86	2.394
B02	**83**	86.5	83	1.802	**83**	83	0.797	**B11**	**88**	**88**	88	1.141	88.2	88	0.646
B03	**138**	**138**	138	0.048	**138**	138	0.136	**B12**	**174**	**174**	174	0.8	**174**	174	1.255
B04	**59**	**59**	59	0.063	**59**	59	0.067	**B13**	**165**	**165**	165	93.421	171.4	165	2.099
B05	**61**	**61**	61	0.815	**61**	61	0.108	**B14**	**235**	**235.2**	235	239.922	235.4	235	1.816
B06	**122**	**122**	122	0.618	124.2	122	0.496	**B15**	**318**	**319.7**	318	320.513	321.7	320	5.178
B07	**111**	**111**	111	0.213	**111**	111	0.183	**B16**	**127**	**127**	127	10.503	129.3	127	1.611
B08	**104**	**104**	104	0.697	**104**	104	0.424	**B17**	**131**	131.2	131	159.33	**131**	131	1.799
B09	**220**	**220**	220	0.199	**220**	220	0.641	**B18**	**218**	**218**	218	1.136	218.3	218	4.502

The performance of JPSO and GRASP-CST are then tested within a fixed maximum computing time (60 seconds), the stopping criterion is either the algorithm finds the optimal solution or within the given 60 seconds. Table 4.15 shows the average tree cost and the average running time when the algorithms found the solutions. In the fixed maximum 60 seconds, GRASP-CST performs slightly better than the JPSO algorithm on the small Steinb instances. However, the JPSO algorithm spends shorter computing time to find the optimal solutions on 8 out of 18 instances compared with GRASP-CST.

Table 4. 15 Comparison between JPSO and GRASP-CST within the fixed computing time on Steinb problems ('OPT' denotes the optimal solution for each instance, bold number means better tree cost or shorter computing time)

No.	**OPT**	**JPSO**		**GRASP-CST**		**No.**	**OPT**	**JPSO**		**GRASP-CST**	
		Cost	**Time(s)**	**Cost**	**Time(s)**			**Cost**	**Time(s)**	**Cost**	**Time(s)**
B01	**82**	**82**	**0.029**	**82**	0.086	**B10**	**86**	**86**	1.771	**86**	**0.442**
B02	**83**	85.5	2.935	**83**	**0.101**	**B11**	**88**	**88**	0.964	**88**	**0.811**
B03	**138**	**138**	**0.045**	**138**	0.136	**B12**	**174**	**174**	**1.073**	**174**	1.257
B04	**59**	**59**	**0.066**	**59**	0.067	**B13**	**165**	167.4	43.432	**165**	**15.163**
B05	**61**	**61**	**0.086**	**61**	0.108	**B14**	**235**	235.6	41.803	**235**	**2.557**
B06	**122**	**122**	**0.824**	**122**	1.201	**B15**	**318**	319.8	43.946	**318.4**	**27.484**
B07	**111**	**111**	0.31	**111**	**0.183**	**B16**	**127**	**127**	11.525	**127**	**2.557**
B08	**104**	**104**	0.538	**104**	**0.431**	**B17**	**131**	131.7	49.862	**131**	**1.81**
B09	**220**	**220**	**0.214**	**220**	0.644	**B18**	**218**	**218**	**1.056**	**218**	7.945

Table 4. 16 Comparison between JPSO, GRASP-CST and DPSO-STP approaches on Steinc instances ('OPT' denotes the optimal solution for each instance, bold number means better tree cost)

No.	**OPT**	**JPSO**			**GRASP-CST**			**DPSO-STP**		
		Mean	**Best**	**Worst**	**Mean**	**Best**	**Worst**	**Mean**	**Best**	**Worst**
C01	**85**	86	85	88	**85**	85	85	**85**	85	85
C02	**144**	149.1	146	152	**144**	144	144	**144**	144	144
C03	**754**	760.5	759	762	760.9	760	761	**754.4**	754	758
C04	**1079**	1083.8	1080	1085	1098.5	1087	1102	**1079.2**	1079	1080
C05	**1579**	**1579**	1579	1579	1587	1587	1587	**1579**	1579	1579
C06	**55**	**55**	55	55	**55**	55	55	**55**	55	55
C07	**102**	102.7	102	103	**102**	102	102	**102**	102	102
C08	**509**	**509.7**	509	510	517.4	517	518	509.9	509	512
C09	**707**	710.7	709	712	714.4	713	715	**709.1**	708	711
C10	**1093**	**1093.8**	1093	1094	1095.8	1093	1097	1094.5	1093	1098
C11	**32**	32.3	32	33	32.6	32	33	**32.1**	32	33
C12	**46**	**46**	46	46	46.4	46	47	**46**	46	46
C13	**258**	**258.5**	258	259	262.4	260	264	260.5	259	263
C14	**323**	**323.9**	323	324	327.8	325	331	324.5	324	326
C15	**556**	**556**	556	556	556.3	556	559	556.7	556	558
C16	**11**	12	12	12	**11.4**	11	12	**11.4**	11	12
C17	**18**	**18.2**	18	19	18.8	18	19	18.4	18	19
C18	**113**	**115.7**	115	116	118	116	119	116	115	119
C19	**146**	148	148	148	151.4	151	152	**147.4**	146	149
C20	**267**	267.1	267	268	269.4	268	271	**267**	267	267

The JPSO algorithm is further compared against GRASP-CST and another recent discrete particle swarm optimisation algorithm, DPSO-STP by Zhong *et al*. (2008), on the larger and harder Steinc instances. The mean, best and worst tree costs found by the three algorithms are presented in Table 4.16. Results show that on 12 of the 20 Steinc instances, JPSO is capable of finding the optimal solutions at least once. Although the DPSO-STP algorithm obtains the best average tree cost on 13 out of 20 problems, the JPSO approach produces smaller tree cost than DPSO-STP on 7 out of 20 problems (C08, C10, C13, C14, C15, C17, and C18). Both JPSO and DPSO-STP outperformed GRASP-CST on obtaining lower tree costs, since GRASP-CST only finds the best result on 5 out of 20 problems.

4.4.2.3 Experiments for the DCLC Multicast Problem on Random Networks

In this group of experiments, the proposed JPSO algorithm is compared with several other algorithms from the literature on a set of random graphs for the DCLC multicast routing problem. This set of random network graphs is the same as that designed in Ghaboosi and Haghighat (2007); Qu *et al*. (2009) and Xu and Qu (2009). The simulations were run 10 times on each random network. The average tree cost on all the random networks are shown in Table 4.17 and the average tree cost and standard deviation of the JPSO algorithm on each network size are presented in Table 4.18.

Table 4. 17 Average tree costs of JPSO and some existing heuristics and algorithms on the random networks of 10-100 nodes

Algorithms		Average Tree Cost
Heuristics	KPP1 (Kompella *et al*., 1993a)	905.581
	KPP2 (Kompella *et al*., 1993b)	911.684
	BSMA (Zhu *et al*., 1995)	872.681
GA-based Algorithms	Wang *et al*. (2001)	815.969
	Haghighat *et al*. (2004)	808.406
TS-based Algorithms	Skorin-Kapov and Kos (2003)	897.875
	Youssef *et al*. (2001)	854.839
	Wang *et al*. (2004)	869.291
	Ghaboosi and Haghighat (2006)	739.095
PR Algorithm	Ghaboosi and Haghighat (2007)	691.434
VNS Algorithms	VNDMR1 (Qu *et al*. 2009)	680.067
	VNDMR2 (Qu *et al*. 2009)	672.470
	Multi-VND	656.777
GRASP Algorithms	GRASP-CST (Skorin-Kapov and Kos 2006)	669.927
	GRASP-VND algorithm	654.520
	GRASP-VND+pilot algorithm	**650.823**
JPSO Algorithm	The proposed JPSO algorithm	**662.100**

It can be seen from Table 4.17 that, except the hybrid GRASP algorithms (GRASP-VND and GRASP-VND+pilot) proposed in the previous work, JPSO produces the better average tree cost than other algorithms tested in this experiment. The GRASP-CST and VNDMR2 approaches are the closest competitors. To make a closer comparison between the three best performing approaches, Table 4.18 gives details of the average tree cost and standard deviation obtained for each of the different network size instances. The tree costs highlighted in bold clearly show that JPSO has a better overall performance. JPSO achieves the smaller average tree costs of the instances on 4 out of 10 network sizes compared with the other two algorithms. JPSO finds the same average tree cost as that of the other two algorithms on the instances of 2 different network sizes. VNDMR2 still shows good performance as it achieves the best average tree cost on instances of 4 network sizes.

Table 4. 18 Average tree cost and standard deviation of JPSO on 10 network sizes compared with VNDMR and GRASP-CST

Network Size	**VNDMR2**			**GRASP-CST**			**JPSO**		
	Tree Cost	**Time (sec)**	σ	**Tree Cost**	**Time (sec)**	σ	**Tree Cost**	**Time (sec)**	σ
10	**94.67**	0.003	0	**94.67**	0.009	0	**94.67**	0.244	0
20	275.33	0.032	0	**270.67**	0.032	0	**270.67**	0.125	0
30	405.93	0.17	17.44	**394.67**	0.17	0	395.67	0.64	4.041
40	**513.87**	0.362	0.28	526.47	0.362	1.789	526	5.052	3.834
50	713.5	0.859	31.27	697.07	0.859	3.429	**687.33**	18.817	1.095
60	790.27	1.392	8.97	761.13	1.392	17.129	**748.67**	6.464	0
70	803.97	2.571	7.01	797.53	2.571	1.644	**785.67**	11.11	0
80	923.9	5.127	9.44	902.67	5.127	5.489	**889.67**	19.068	0.577
90	**1182.6**	11.705	9.6	1201.93	11.705	18.019	1194	30.049	1.378
100	**1020.67**	15.332	21.32	1052	15.332	23.404	1028.67	71.798	2.466

As noted earlier in this section, to the best of my knowledge, this is the first study of a JPSO algorithm on both the Steiner tree and DCLC multicast routing problems. A set of extensive experimental comparisons have been investigated on a large number of benchmark Steiner tree problems from the OR-library and a range of multicast routing problems with different characteristics. In both problem scenarios, experimental results demonstrate the superior performance of the JPSO algorithm over a number of state-of-the-art approaches. It shows that swarm optimisation is a successful approach to tackle the DCLC multicast routing problem and the underlying Steiner tree problem.

The research outcomes of the JPSO algorithm have been reported in a paper which is now under review at the Journal of Heuristics.

4.5 A Scatter Search with Path-relinking Algorithm

This section investigates the first hybrid Scatter Search and Path Relinking metaheuristic, hereafter named SSPR, for the DCLC multicast routing problem. Although the SS metaheuristic has been successfully applied to provide promising solutions for combinatorial optimisation problems in the literature, as far as I am concerned, no research has been carried out by applying it for solving the QoS multicast routing problem. SS is a very flexible metaheuristic, where each component can be designed in alternative ways with regard to the problems being concerned. The basic SS consists of 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 *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×*b*. The solutions in *RefSet* are ordered according to their objective function value, where the best solution is the first one in the set.

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. the size of each subset is 2. The cardinality of *SubSets* is therefore 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.

d) **An Improvement Method** is applied to enhance a solution by exploring neighbourhoods of the current solution in order to generate new better solutions.

e) **A Reference Set Update Method** aims to maintain the reference set consisting of the *b* best solutions obtained in the previous search procedure. Different criteria may be defined to add solutions to *RefSet* and delete solutions from *RefSet*.

As suggested in (Glover *et al*., 2000), a path relinking procedure is applied as the combination method in the SSPR algorithm. PR is an evolutionary method which combines elements of pairs of solutions by starting from one of these solutions, called an initiating solution, and generating a trajectory in the neighbourhood space which connects to another solution, called the guiding solution. The main goal of PR is to incorporate attributes of the pairs of solutions and record a series of moves leading from the initiating solution to the guiding solution. To further intensify the search towards better solutions, two improvement strategies: tabu search and variable neighbourhood search, are explored in the hybrid SSPR algorithm. The details of the proposed SSPR algorithm are described in the following subsection.

4.5.1 Algorithm Description

In the SSPR algorithm, the Diversification Generator is a pure random generator to create a set of initial solutions (*Pop*), 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. As in the previous GRASP algorithm, the binary array with $|V| = n$ bits is used to represent a solution. 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. A reference set (*RefSet*) with diverse solutions is selected from the initial solution set *Pop*. The solutions in *RefSet* are ordered according to their objective function value, where the best solution is the first one in the set. The Subset Generation Method in SSPR generate a set of subset (*SubSets*) selected from *RefSet*, where the *SubSets* include all pairs of solutions in *RefSet*. A path relinking procedure is then designed as the solution combination method to operate on each pair of solutions in *SubSets*. After a new solution is generated by the path relinking, two local search heuristics, a tabu search and a variable neighbourhood search algorithm, are applied as the improvement methods to further explore better neighbouring solutions of the newly generated solution. Finally, the *RefSet* is updated by adding new better solutions in it and deleting worse solutions from it. The stopping criterion in SSPR is either within a number of iterations or a given amount of time. The pseudo-code of the proposed SSPR is presented in Figure 4.19.

```
SSPR(G =(V, E), s, R, Δ, Psize, b)
 {  // s: source node; R: destination set; Δ ≥ 0 is the delay bound; Psize: the size of the initial population (Pop);
    // b: the size of the reference set (RefSet);
   if for the Dijkstra's least delay multicast tree T, $d(p_T(s, r_d)) > \Delta, \forall r_d \in R$;
   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  $Pop = Pop \cup x$ ;  else  Discard x ;
      until (|Pop| = Psize);
      repeat
         Build RefSet = { $x^1$, …, $x^b$} with b diverse solutions in Pop.
         Order the solutions in RefSet according to their objective function value in 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
                     Apply the path relinking method to combine the solutions of the selected subset to obtain a
                     new solution x. //See Section 4.5.1.1.
                     // d) Improvement Method
                     Improve the generated new solution x. //Apply the local search heuristic, see Section 4.5.1.2.
                     // 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.
}
```

Figure 4. 19 The pseudo-code of the SSPR algorithm

4.5.1.1 The Path Relinking Method

The path relinking method in the proposed SSPR algorithm is described as follows. During the PR procedure, the difference bits between the initiating solution and the guiding solution will be recorded by comparing the correspoding bits of the two solutions's multicast tree binary arrays. At each step, starting from the initiating solution, the PR 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 by the given nodes in the array while concerning the end-to-end delay bound from the source node to every 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 PR procedure.

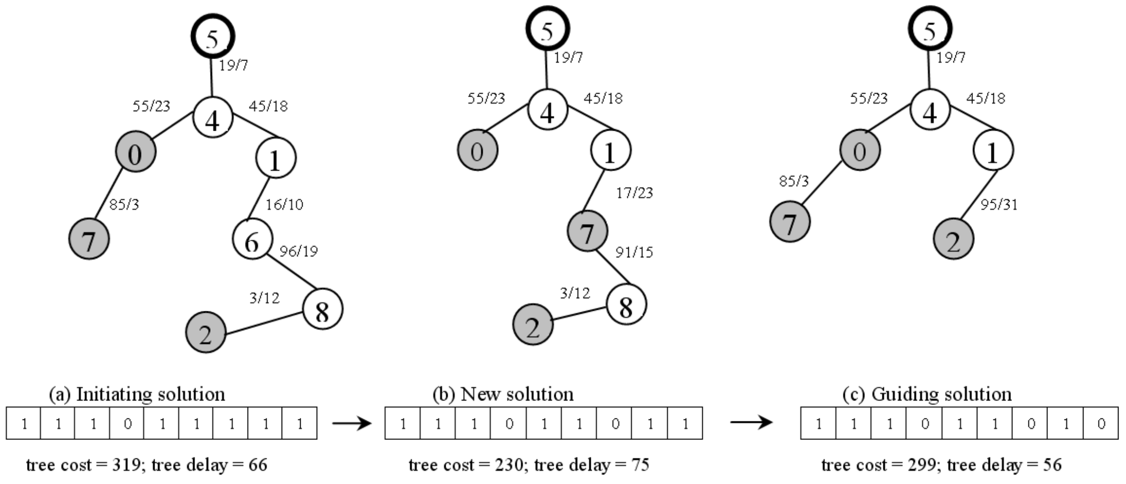


Figure 4. 20 An example of the PR process on the random network shown in Figure 3.1

An illustrative example of the PR method in the proposed SSPR method is shown in Figure 4.20. As described above, the binary array is used to present a solution. It can be seen 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 PR process. Here the delay bound $\Delta = 100$.

4.5.1.2 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, two variants of SSPR algorithms are designed, namely SSPR-TS and SSPR-VND, where two improvement methods, a TS heuristic (Skorin-Kapov and Kos, 2003) and the previous VND multicast routing algorithm (Qu *et al*., 2009), respectively, are applied within the SSPR algorithm.

The TS heuristic uses the same solution representation as that in the path relinking method described above. The initial solution of the TS heuristic is the best solution generated by the PR procedure. Neighbouring solutions include all the solutions whose binary arrays are exactly one bit different from the current solution. In other words, the neighbouring 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 neighbouring solution becomes the current solution in the next iteration. The process stops after a desired number of iterations without improvements; here the number is set to 2. In the TS 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 VND algorithm in SSPR-VND is based on the previous VNDMR2 employing three neighbourhood structures, one is a node-based neighbourhood by flipping nodes in the network to generate new neighbouring 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. Detailed description about the VNDMR2 can be found in Section 4.2.1.2.

4.5.2 Experiments and Results

In order to evaluate the performance of the proposed SSPR algorithm, a large number of simulations have been conducted on the benchmark Steinb instances and a set of random networks, which are the same simulation data used in the previous work for testing algorithms. Details of Steinb instances can be found in Section 3.4.4, and the characteristics of the random networks have been described in Section 4.2.2.3. All simulations were run on a Windows XP computer with PIV 3.4GHZ, 1G RAM.

4.5.2.1 Experiments on Steinb Instances in the OR-library

1) 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, the performance of the SSPR algorithms with different population sizes (*Psize*) and reference set sizes (*b* = *Psize*/10) is evaluated. The iteration number is set as 4 in all the SSPR algorithms. 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 10 times for each variant of the algorithm. Table

4.19 and Figure 4.21 present the average tree cost, standard deviation (σ) and computational time of the SSPR algorithms on the 18 Steinb instances with different population sizes. Here the delay bound $\Delta_1 = \infty$.

Table 4. 19 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**	0.00	**82**	0	**82**	0.00
B02	89.3	0.95	89	0	88.7	0.95	**87.8**	2.53	88.5	0.97
B03	138.4	1.26	**138**	0	**138**	0.00	**138**	0	**138**	0.00
B04	67.6	3.37	67.8	3.82	64.3	1.95	**64.2**	2.30	64.9	1.85
B05	62.7	0.03	61.9	0.88	61.7	0.67	61.5	0.71	**61**	0.00
B06	133.4	6.64	131.4	4.62	127.4	2.50	**125.7**	1.42	**125.7**	1.34
B07	113.4	2.84	113.8	3.16	112	0.47	**112.1**	0.32	112.2	0.42
B08	107.7	3.09	105.7	1.25	106	2.05	**105.4**	0.97	106	1.33
B09	221	0	220.5	0.53	220.6	0.52	220.3	0.48	**220.2**	0.42
B10	100.4	4.40	98.7	5.68	95.7	1.42	97	3.40	**95.7**	0.67
B11	108.6	3.06	103.1	3.38	105.5	1.84	99.8	6.16	**97**	6.53
B12	183.2	1.48	180.9	2.56	180.2	1.87	**178.8**	1.81	179.2	3.01
B13	184	4.69	179.7	5.83	175.4	4.03	174.9	4.15	174.3	4.27
B14	254.2	8.07	250.8	5.07	249.3	6.27	**245.9**	2.64	246.7	3.43
B15	328.3	2.0	326.6	3.31	325.9	2.18	**324.2**	2.53	324.9	1.37
B16	150	0	148.7	4.11	146.7	4.79	**140.5**	5.17	140.6	6.96
B17	139.1	0.32	139.6	0.97	137.5	2.68	137.5	2.32	**135.5**	1.96
B18	222.8	1.14	222.4	0.97	222.4	1.35	**221.2**	1.03	**221.2**	0.92

Results in Table 4.19 indicate that SSPR with *Psize* = 40, 50 and 60 outperforms other variants of the algorithm. The paired t-test value between the average tree costs on the 18 instances with *Psize* = 40 and 50 is then calculated. The result 2.56 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 0.772, difference between these two variants can be seen as not significant.

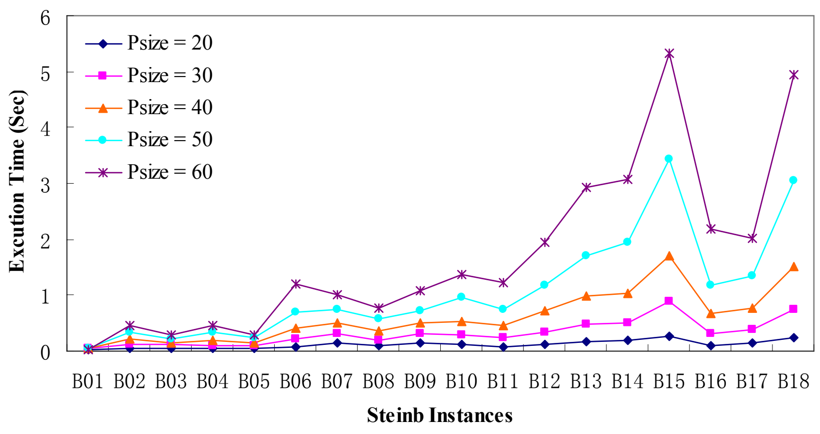


Figure 4. 21 The computational time of the SSPR algorithm with different population size (*Psize*) on Steinb instances

From Figure 4.21, we can see that with the increasing population size, the execution time of the SSPR algorithm increases. In particular, the computing time of the SSPR algorithm with *Psize*= 50 is less than that of the SSPR algorithm with *Psize* = 60, their performance is similar with respect to the average tree cost on each instance. Thus the most appropriate population size *Psize* of the proposed SSPR algorithm is 50 for the DCLC multicast routing problem. The same population size is set as 50 in the later experiments.

2) Comparisons on Steinb Instances with Different Delay Bounds

In the second group of experiments, the performance of SSPR-VND and SSPR-TS are compared with other two algorithms GRASP-VND and Multi-VND in the previous work (Xu and Qu, 2009) on the Steinb instances with three different delay bounds, $\Delta_1 = \infty$, $\Delta_2 = 1.1\times Delay(T_{OPT})$ and $\Delta_3 = 0.9\times Delay(T_{OPT})$ (see Section 4.3.2.1), respectively. Firstly, the delay bound is set as $\Delta_1 = \infty$, where ∞ denotes a large enough value so that the solutions obtained are actually the solutions to the minimum Steiner tree problem without the delay constraint. For a fair comparison, the same running time (60 seconds) is set for the four algorithms in each run and all algorithms were run 10 times on each instance.

Table 4. 20 Experimental results for minimum Steiner tree problem ($\Delta_1 = \infty$) (the best results are in bold, the values marked with '*' denote the optimal solution)

No.	SSPR-VND			SSPR-TS			GRASP-VND			Multi-VND		
	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	85	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	123.6	122	0.84
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	167.4	165	2.07	167.6	165	2.67	166.3	165	2.11	**165***	165	0
B14	235.5	235	0.53	235.9	235	0.32	**235***	235	0	235.6	235	0.52
B15	**318***	318	0	319.4	318	0.97	319.4	318	0.97	320	320	0
B16	**127***	127	0	131.8	127	2.3	**127***	127	0	**127***	127	0
B17	**131***	131	0	**131***	131	0	**131***	131	0	**131***	131	0
B18	**218***	218	0	218.1	218	0.32	**218***	218	0	**218***	218	0

The average, best and standard deviation of the two variants of SSPR algorithms, along with GRASP-VND and Multi-VND on the Steinb instances are illustrated in Table 4.20. The values marked with '*' indicate the cost of the optimal solution. In Table 4.20, we can see that both SSPR-VND and

GRASP-VND have similar performance, obtaining 16 best solutions out of 18 instances in terms of the average tree cost and both are better than SSPR-TS which finds 13 best solutions and Multi-VND which finds 15 best solutions. In addition, the results obtained by SSPR-VND are more stable than those of SSPR-TS since the SSPR-VND has smaller average standard deviation (0.144) compared with that of SSPR-TS (0.509). The experiment results also show that SSPR-VND outperforms SSPR-TS, mainly due to the better improvement method VND within the same SSPR metaheuristic.

In the second set of experiments, the delay bound is set as $\Delta_2 = 1.1 \times Delay(T_{OPT})$. With the slightly tighter bounded end-to-end delay compared with the delay bound $\Delta_1 = \infty$ 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 in Table 4.21. SSPR-VND has similar overall performance to those of GRASP-VND and Multi-VND, which find 15, 16 and 14 out of 18 best results in terms of average tree cost, respectively.

Table 4. 21 Experimental results for the DCLC multicast routing algorithm with $\Delta_2 = 1.1 \times Delay(T_{OPT})$ (Values in the second column ('Δ') are delay bounds, the best results are in bold, the values marked with '*' denote the optimal solution)

No.	Δ	**SSPR-VND**			**SSPR-TS**			**GRASP-VND**			**Multi-VND**		
		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	85.7	83	2.55	**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***	122	0	**122***	122	0	123.8	122	0.63
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	167.4	165	2.07	169.3	165	1.77	166.8	165	2.35	**166.6**	165	2.07
B14	221	236.2	235	3.31	257.8	244	11.92	**235***	235	0	243.2	236	3.79
B15	308	**318.2**	318	0.63	318.9	318	1.05	319.2	318	1.03	319.8	318	0.63
B16	291	**127***	127	0	134	130	1.76	**127***	127	0	**127***	127	0
B17	219	131.3	131	0.95	**131***	131	0	**131***	131	0	**131***	131	0
B18	425	**218***	218	0	218.1	218	0.33	**218***	218	0	218.1	218	0.32

In the third group of experiments, the delay bound Δ_3 is set to a smaller value $0.9 \times Delay(T_{OPT})$. The optimal solutions are thus not known to any of the cases. Due to the more strict delay constraints, we can see that no feasible solutions are obtained for some instances as presented in Table 4.22. The table again shows that SSPR-VND outperforms SSPR-TS with respect to the average tree costs on 11 instances. It is also observed 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 Multi-VND when comparing the number of best average tree costs found by each algorithm which are 14, 11 and 10 out of 18 instances, respectively. It demonstrates that the SSPR-VND is more flexible in solving the problems of different delay bounds.

Table 4. 22 Experimental results for the DCLC multicast routing algorithm with $\Delta_3 = 0.9\times$ *Delay*(T_{OPT}) (Values in the second column ('Δ') are delay bounds, the best results are in bold)

No.	Δ	**SSPR-VND**			**SSPR-TS**			**GRASP-VND**			**Multi-VND**		
		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	89.2	86	1.69	**84**	84	0	**84**	84	0
B03	203	**141**	141	0	142	142	0	/	/	/	**141**	141	0
B04	142	**62**	62	0	64.5	63	0.71	**62**	62	0	64	64	0
B05	102	**62**	62	0	62.3	62	0.48	**62**	62	0	**62**	62	0
B06	199	125	125	0	125	125	0	**124.9**	124	0.33	125	125	0
B07	173	**112**	112	0	113	112	2.83	/	/	/	**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	0.32	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	177.5	177	1.58	**176.6**	175	0.89	177	177	0
B13	139	**169**	169	0	169.8	169	1.03	171.8	168	2.5	**169**	169	0
B14	180	/	/	/	244	237	6.38	237	237	0	/	/	/
B15	194	326.1	320	4.86	338.6	329	4.85	**327.1**	326	1.55	328.7	326	0.95
B16	238	**130.5**	129	1.58	133.8	129	1.99	**130.5**	129	1.58	132	132	0
B17	180	**134**	134	0	/	/	/	134.3	134	0.48	**134**	134	0
B18	348	**219**	219	0	219.1	219	0.32	219.2	219	0.42	219.4	219	0.52

4.5.2.2 Experiments for the DCLC Multicast Routing Problem on Random Networks

Furthermore, in order to compare the SSPR algorithms with other existing heuristics and algorithms in the literature, a group of experiments were carried out on a set of random networks which are the same as the simulation data in the previous work. The simulation results are reported in Table 4.23 and Table 4.24.

In Table 4.23, we can see that SSPR-VND has the best overall performance on the random networks with respect to the average tree cost in comparison with other heuristics and algorithms shown in the table. SSPR-VND outperforms SSPR-TS in terms of the average tree cost, showing that the better improvement method improves the performance of the SSPR metaheuristic.

Table 4.24 presents more details of the average tree cost and standard deviation of these four algorithms on each network size. It can be seen that SSPR-VND outperforms other three algorithms by obtaining 7 best results out of the 10 different network size problems. GRASP-VND only finds 4 best average tree costs, showing that the SSPR metaheuristic is more effective than the GRASP metaheuristic for the DCLC multicast routing problem when the same local search method is applied. The average standard deviation of SSPR-VND for the 10 network sizes is 4.8, which is better than 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 networks.

Table 4. 23 Average tree costs of SSPR-TS, SSPR-VND and some existing heuristics and algorithms on the random networks of 10-100 nodes

Algorithms		**Average Tree Cost**
Heuristics	KPP1 (Kompella *et al*., 1993a)	905.581
	KPP2 (Kompella *et al*., 1993b)	911.684
	BSMA (Zhu *et al*., 1995)	872.681
GA-based Algorithms	Wang *et al*. (2001)	815.969
	Haghighat *et al*. (2004)	808.406
TS-based Algorithms	Skorin-Kapov and Kos (2003)	897.875
	Youssef *et al*. (2001)	854.839
	Wang *et al*. (2004)	869.291
	Ghaboosi and Haghighat (2006)	739.095
PR Algorithm	Ghaboosi and Haghighat (2007)	691.434
VNS Algorithms	VNDMR1 (Qu *et al*., 2009)	680.067
	VNDMR2 (Qu *et al*., 2009)	672.470
	Multi-VND	656.777
GRASP Algorithms	GRASP-CST (Skorin-Kapov and Kos, 2006)	669.927
	GRASP-VND algorithm	654.520
	GRASP-VND+pilot algorithm	650.823
PSO Algorithm	JPSO algorithm	662.100
SS Algorithms	The proposed SSPR-TS	679.690
	The proposed SSPR-VND	**644.840**

Table 4. 24 Average tree cost and standard deviation of the SSPR algorithms compared with GRASP-VND and Multi-VND on the random networks

Network Size	**SSPR-VND**			**SSPR-TS**			**GRASP-VND**			**Multi-VND**		
	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	0.008
20	272.53	2.41	0.314	274.4	1.97	0.141	**272.07**	2.25	0.085	275.33	0	0.089
30	393.5	3.57	1.453	414.1	14.96	0.577	**392.33**	0	0.353	395.27	3.95	0.461
40	513.33	0.00	3.522	533.43	3.22	1.291	**512.8**	1.55	0.857	513.33	0	1.238
50	**660.83**	0.53	9.575	707.43	8.27	2.727	662.33	1.94	2.109	665.07	3.92	3.027
60	**748.07**	7.03	13.637	759.07	17.92	5.271	757.33	13.48	3.894	757.37	14.01	4.894
70	**779.5**	5.97	29.608	811.2	5.57	9.08	780.83	2.96	9.029	796.2	4.48	9.268
80	**863.33**	5.93	66.356	935.33	22.65	15.791	868.87	7.73	19.421	896.2	4.48	16.365
90	**1132.77**	19.35	116.419	1215.17	21.07	31.595	1155.57	19.02	32.621	1136.23	11.17	38.76
100	**989.87**	3.19	177.454	1048.83	16.03	47.218	995.23	4.23	41.681	1002.9	4.4	53.256
Avg.	**644.84**	4.8	41.837	679.690	11.39	11.371	649.203	5.316	11.006	653.257	4.641	12.737

The above simulations on small and medium sized problems from SteinLib in OR-library and a group of random networks demonstrate that SSPR-VND with variable neighbourhood search outperforms SSPR-TS with tabu search in terms of average tree costs. This indicates that the improvement method will greatly affect the performance of the proposed SSPR metaheuristic, and better local search can contribute to better results. The SSPR-VND algorithm has the best performance in terms of the average tree cost for the DCLC multicast routing problem in comparison with some existing algorithms and heuristics.

The results of the SSPR-VND algorithm have been pulished by the journal of Applied Intelligence on line (DOI:10.1007/s10489-101-0256-x).

4.6 Summary

This chapter investigates four new metaheuristic approaches for the DCLC multicast routing problem, including the VNS algorithms, a JPSO algorithm, a hybrid GRASP algorithm and a combination of SS with PR algorithm. Extensive experiments on a set of benchmark instances in the OR-library and a group of random networks have been carried out to evaluate the performance of each approach and compare the proposed algorithms with other existing heuristics and algorithms. All of these proposed metaheuristic algorithms outperform other existing algorithms in the literature on a number of benchmark instances and have shown to be competitive for tackling the DCLC multicast routing problem. The main characteristic of the proposed VNS approach is that of designing three simple yet effective neighbourhood structures, which enables a flexible exploration over the search space of the complex DCLC multicast routing problem. A large amount of experimental results demonstrate that the proposed VNS approach is effective and efficient for solving the problem. The cooperative nature of JPSO is an important factor to its success for both the Steiner tree problem and the DCLC multicast routing problem. Although the proposed JPSO offers a very robust approach, one shortcoming of the JPSO is its relatively high computing time among the four new approaches especially for large networks. The hybrid GRASP approach combined with VNS local search and the post-processing pilot method has shown to produce high quality solutions for the DCLC multicast routing problem efficiently. The combination of SS with PR by applying a variable neighbourhood search algorithm as the improvement method achieves the best performance in terms of average tree cost on a set of random networks for the DCLC multicast routing problem. The research in this chapter demonstrates that these four new metaheuristic approaches are competitive and effective solutions for solving the DCLC multicast routing problem.

CHAPTER 5. Fitness Landscape Analysis for the DCLC Multicast Routing Problem

5.1 Introduction

This chapter presents the first fitness landscape analysis on the DCLC multicast routing problem. Although QoS multicast routing problems have attracted many research attentions over the past two decades, few researches have been conducted to analyse the underlying landscape features. Before properly choosing a specific heuristic or metaheuristic for a particular optimisation problem, the theoretical landscape analysis of the problem is very useful for observing the behaviour of search algorithms and thus can help predicting their performance. Due to the lack of the theoretical foundations of the landscape analysis on the DCLC multicast routing problem, the first analysis of the fitness landscapes on the problem is performed in this thesis. As reviewed in Section 3.4.3.4, to the best of my knowledge, there are only three related work (Zahrani *et al*., 2006; 2007; and 2008) on the landscape analysis for the QoS multicast routing problem. In these papers, the same authors analysed the landscapes on a different type of multicast routing problem, namely the group multicast routing problem, by using a simulated annealing procedure with a logarithmic cooling schedule to estimate the depth of the deepest local minima in the landscape. The aim in this chapter is to give a better understanding of landscape properties for the DCLC multicast routing problem within the single multicast request scenario. This may provide some theoretical basis for more advanced search algorithms on various related applications of multicast routing problems in the future by predicting the performance and improving the design of metaheuristics.

5.2 Fitness Landscape Analysis Methods

The concept of fitness landscapes (Wright, 1932) introduced to investigate the behaviour of biological evolutionary optimisation algorithms has shown to be very useful in the evolutionary theory. The fitness landscape analysis is an ideal tool for predicting the performance of combinatorial optimisation algorithms by measuring the problem's landscape and understanding the behaviour of algorithms. Based on a solution representation, and an objective function, the landscape of the problem can be defined where the set of solutions in the search space are represented by points of different height to reflect their fitness (objective function value) in the multi-dimensional hyper-space. A heuristic algorithm can be seen as a search process to navigate through the landscape in order to find the highest peak (or the lowest point) for the maximisation (or minimisation) problem.

A fitness landscape of a problem instance for a given combinatorial optimisation problem can be defined as $L(X, f, d)$, where X represents a set of points (solutions) in the landscape, $f: X \rightarrow \Re$ is an objective function which assigns a real-valued fitness to each of the points in X, and d is a distance metric which defines the spatial structure of the landscape. A fitness landscape can thus be interpreted as a graph $G_L = (V, E)$ with a set of vertices $V = X$ and a set of links $E = \{(x, y) \in X \times X \mid d(x, y) = d_{min}\}$, where d_{min} denotes the minimum distance between two points in the search space. Another property of the landscape is its diameter $diamG_L$ which is defined as the maximum distance between two points in the search space.

For a search problem, the set of possible solutions may be encoded by using binary strings $X = \{0, 1\}^n$ of a fixed length n. This encoding generates a hyper-space G_L of dimension n, where n is normally the size of the problem. For a binary coded search problem, the distance measure d can be defined by using the Hamming distance between bit strings. The minimum distance d_{min} between two solutions is 1 (one bit with a different value), and the maximum distance $diamG_L$, i.e. the diameter of the landscape, is n. Statistic methods have been employed to measure the fitness landscape properties such as the fitness distance correlation and auto-correlation analysis. These two methods are commonly used as the landscape analysis techniques in the literature, and will be used in this work to conduct the fitness landscape analysis of the DCLC multicast routing problems in the following subsections.

5.1.1 Fitness Distance Correlation (FDC) Analysis

The fitness distance correlation (FDC) coefficient proposed by Jones and Forrest (1995) is the most commonly used method to estimate the global feature of the fitness landscape. It has been widely used as a measure to predict the problem difficulty for genetic algorithms. Given a set of points $\{x_1, x_2, \ldots, x_m\}$ and their fitness values, the FDC coefficient ρ is defined as follows:

$$\rho(f, d_{opt}) = \frac{Cov(f, d_{opt})}{\sigma(f)\sigma(d_{opt})} \tag{5.1}$$

where $Cov(\cdot, \cdot)$ denotes the covariance of two random variables, $\sigma(\cdot)$ denotes the standard deviation and d_{opt} represents the distance between a sample point and the global optimum or best-known solution in the search space. The sample points can be arbitrary points or local optimal points within the search space. The FDC coefficient determines how close the fitness values of a set of sample points and their d_{opt} to the optimum are correlated by examining the statistic correlation between the fitness and the distance. For a maximisation problem, if the fitness of sample solutions in a search increases when the search moves towards the optimum (the distance becomes smaller), then the search is expected to be easy, and properly guided by the correlation along a "path" to the optimum via solutions with increasing fitness. A value of $\rho = -1.0$ (or $\rho = 1.0$) for a maximisation (or minimisation) problem indicates a perfect correlation between the fitness and the distance to the optimum, and thus predicts an easy problem. On the contrary, a value of $\rho = 0$ means that no correlation exist between the fitness and the distance to the optimum, and thus the underlying problem is hard to solve. As classified in (Jones and Forrest, 1995), a value of $\rho \leq -0.15$ (or $\rho \geq 0.15$) for maximisation (or minimisation) problems indicates an easy problem.

Fitness distance analysis has been conducted for various combinatorial optimisation problems, for example the TSP problem (Boese, 1995), the flow-shop scheduling problem (Reeves, 1998), the graph matching problem (Stadler, 1992), the graph

bipartitioning problem (Merz and Freisleben, 1998), and the timetabling problem (Ochoa *et al.*, 2009). A summary of landscape metrics and related issues on evolutionary algorithms can be found in (Kallel *et al.*, 2001).

5.1.2 Auto-correlation Analysis

An important property of a landscape is its ruggedness, which significantly influences the performance of search heuristics. Generally, the more rugged a landscape is, the harder the problem can be solved by heuristic search algorithms. To measure the ruggedness of a fitness landscape, Weinberger (1990) suggests a statistical method to analyse the correlation structure of a fitness landscape based on the auto-correlation method. The idea is to generate a series of random walks of sufficiently large number of steps and calculate the correlation of neighbouring points in the random walk in the search space. The fitness of each solution encountered during the random walk is recorded to obtain a time series of fitness values. The auto-correlation of the time series obtained from the random walk of W steps in the search space, denoted by ρ_i, can be empirically estimated by $r(i)$:

$$r(i) = \frac{\sum_{t=1}^{W-i}(f_t - \bar{f})(f_{t+i} - \bar{f})}{\sum_{t=1}^{W}(f_t - \bar{f})} \tag{5.2}$$

where $\bar{f}$ is the mean fitness of the W points visited, and i is the time lag or distance between two points along the random walk. Based on this auto-correlation function, the correlation length l can be defined as follows (Stadler, 1995):

$$l = -\frac{1}{\ln(|r(1)|)} \tag{5.3}$$

where $r(1) \neq 0$ is defined according to Eq. (5.2). The correlation length l directly reflects the ruggedness of a landscape, the smaller value for l indicates a more rugged landscape. A landscape is said to be smooth if there is high correlation between neighbouring points. The correlation length l is typically related to the size of the problem instance (Stadler, 1995), so the relative correlation length in relation to the landscape diameter $diamG_L = n$ (here n is the size of the problem instance) is often reported by calculating n / l. The auto-correlation analysis has been carried out in the literature for measuring the landscapes of different problems (Weinberger, 1990; Manderick *et al.*, 1991; Hordijk, 1996; and Ochoa *et al.*, 2009).

5.3 Experimental Study

The landscape analysis is performed on three heuristic methods and two neighbourhood operators based on the fitness distance correlation analysis and the auto-correlation analysis for the DCLC multicast routing problem. The testing problems have been generated based on a set of small and medium sized Steiner tree problems Steinb from SteinLib in the OR-library. The characteristics and optimal solutions of the 18 Steinb instances have been presented in Table 3.3 in Chapter 3.

5.3.1 Experimental Results of FDC Analysis

To analyse the landscape of the DCLC multicast routing problem, the fitness distance correlation analysis on the Steinb instances is firstly conducted. For FDC analysis, the optimal solution of the given instance needs to be known in advance. For problems where the optimal solution is not known, in the literature the best-known solution is usually used as an estimation of the optimum in many studies. x^* in this study denotes the optimal solution or the best-known solution on the search space.

In this study, a binary vector has been used to present the solution (the multicast tree) for a multicast routing problem. Given a multicast routing problem in a network $G = (V, E)$ with $|V|$ nodes, the binary vector with the ordered $|V|$ bits is defined to represent a multicast tree, where each bit (from 0 to $|V| - 1$) corresponds to one node in the network. Each bit in the vector takes a value of 1 if the corresponding node is in the multicast tree, 0 otherwise. This simple binary vector has been widely used and shown to be easy yet effective for representing a multicast solution in the literature (Skorin-Kapov and Kos, 2003; Skorin-Kapov and Kos, 2006; and Xu and Qu, 2009). Based on the solution representation of binary vector, the Hamming distance has been used to calculate the distance between solutions. The minimum distance d_{min} between two solutions is 1. As the source and all destination nodes should be included in a multicast tree, the value of their corresponding bits in the binary vector (of a feasible solution) should always be 1. The maximum distance $diamG_L$ between two solutions is therefore $|V| - |R| - 1$, where the $|R|$ is the number of destination nodes in the multicast group.

5.3.1.1 FDC Analysis on Steinb Instances with Different Delay Bounds

In order to have a set of solutions with even distribution of distances to the optimal solution, a fixed number (10 in the current experiments) of feasible solutions have been randomly generated for each distance i away from x^*, $i = 1, \ldots, |V| - |R| - 1$. This set of random solutions is thereafter used as the starting points of the local search heuristic method to generate a set of local optima. As the local search starting from some random solutions may lead to the same local optimum, up to $10\times(|V| - |R| - 1)$ number of local optima may be generated by the local search heuristic.

Different local search heuristic methods may be applied to generate different sets of local optima. In the FDC analysis, a simple local search heuristic, namely LS, is implemented to produce the local optima in the search space. Based on the binary vector solution representation defined above, the LS heuristic searches for the local optima in the search space by using a node-based neighbourhood in a non-deterministic procedure. At each step of the search, the node-based neighbourhood operator flips a randomly selected bit (excluding the bits of the source node and the destination nodes) in the binary vector of the current solution. The modified Prim's spanning tree algorithm is then used to create a new minimum spanning tree based on the newly produced set of nodes in the binary vector, i.e. after a node has been removed or a new node has been added to the binary vector. The new solution that is better than the best solution found so far with respect to the tree cost will be accepted as the new current best solution during the search. This procedure is repeated until all possible bits have been flipped and the best solution found during the procedure is accepted as the local optimum. Within the $|V|$ bit representation of a multicast tree in the network $G = (V, E)$, the bits corresponding to the source node and $|R|$ destination nodes can not be flipped, so the number of possible neighbouring solutions of the current multicast tree is bounded by $|V| - |R| - 1$. The similar node-based neighbourhood operator has also been applied in (Skorin-Kapov and Kos, 2003; Skorin-Kapov and Kos, 2006; Xu and Qu, 2009). Details of the LS heuristic are shown in Figure 5.1.

```
**LS**($G = (V, E), s, R, \Delta, T_0$)
{ // $s$: the source node; $R$: the destination nodes set, i.e. the multicast group; $\Delta \geq 0$: the delay bound;
  // $T_0$: the random initial solution, represented by a binary vector corresponding to $n$ nodes, $n = |V|$;
  $T_{best} = T_0$;
  $k = 1$; $k_{max} = n - |R| - 1$;
  Mark all bits as unvisited in $T_0$;
  while $k \leq k_{max}$ do {
        Randomly select an unvisited bit $x$ in $T_0$; // $s$ and nodes in $R$ are excluded
        Flip the value of bit $x$ to generate a new binary vector $T_x$;
        Generate a minimum spanning tree $T'$ of the given nodes in $T_x$ by using the Prim's algorithm;
        if((($Cost(T') < Cost(T_{best})$) and ($Delay(T') \leq \Delta$)) or
              (($Cost(T') == Cost(T_{best})$) and ($Delay(T') < Delay(T_{best})$)))
        then $T_{best} = T'$;
        mark bit $x$ as visited;
        $k$++;
  end of while loop }
  return $T_{best}$;
}
```

Figure 5. 1 Pseudo-code of the local search heuristic LS for producing the local optima

For the DCLC multicast routing problem, the delay bound plays a key role in obtaining feasible solutions in search algorithms. The smaller the delay bound is, the tighter the problem is constrained. To observe the effect of delay bounds on the fitness landscape of the DCLC multicast routing problem, the fitness distance analysis on the Steinb instances with three different delay bounds have been carried out, namely $\Delta_1 = \infty$, $\Delta_2 = 1.1 \times Delay(T_{OPT})$, and $\Delta_3 = 0.9 \times Delay(T_{OPT})$, as described in Section 4.3.2.1.

For FDC analysis, it is known that higher values of correlation ρ indicate that the fitness and the distance to the optimum are correlated and thus the problem presents to be easier for search algorithms. As suggested in (Jones and Forrest, 1995), problem difficulty has been classified based on FDC coefficients, where a value of $\rho \geq 0.15$ for minimisation problems indicates a straightforward problem and should be easy for search algorithms. For a value of $\rho \leq -0.15$, the distance to the optimum increases with decreasing fitness, indicating that the problem being concerned is deceptive and misleading. For problems with $-0.15 < \rho < 0.15$, the landscape presents no correlation, thus indicates that the problem being concerned can be classified as difficult to solve.

Table 5. 1 Fitness distance correlation on the Steinb instances with different delay bounds ('OPT' denotes the cost of the optimal solution for each unconstrained Steiner tree instance, Δ: the value of delay bound for the corresponding problem instance)

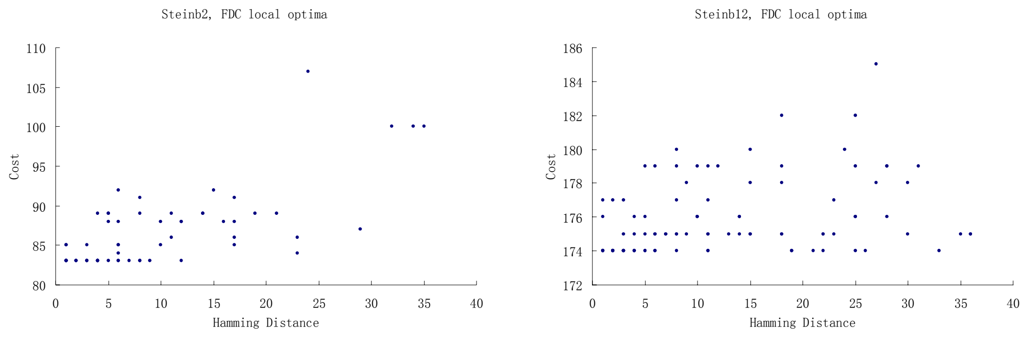
No.	**OPT**	$\Delta_1 = \infty$			$\Delta_2 = 1.1\times Delay(T_{OPT})$				$\Delta_3 = 0.9\times Delay(T_{OPT})$			
		cost	σ	ρ	Δ	**cost**	σ	ρ	Δ	**cost**	σ	ρ
B01	82	82.25	0.44	-0.23	145	82.1	0.4	0.89	118	84	3	0.6
B02	83	86.6	4.89	0.71	228	85.23	3.34	0.51	187	88.45	3.54	0.4
B03	138	139.48	2.54	-0.05	248	139.56	2.62	-0.03	203	144.54	3.54	-0.2
B04	59	65.20	6.17	0.54	173	70.67	11.04	0.56	142	72.19	10.17	0.52
B05	61	62.52	2.39	0.27	125	65.86	5.73	0.54	102	67.57	7.92	0.3
B06	122	125.03	2.47	0.19	281	125.86	3.35	0.4	199	133.62	10.81	0.23
B07	111	112.03	2.89	0.73	212	112.03	3.09	0.79	173	113.28	2.73	0.47
B08	104	104.86	2.04	0.56	209	105.68	6.16	0.69	171	113.28	10.14	0.27
B09	220	220.66	1.80	0.14	280	220.18	0.39	0	229	225	3.87	-0.14
B10	86	91.08	5.63	0.44	262	91.08	5.65	0.46	215	92.32	6.41	0.24
B11	88	96.5	3.61	0.48	235	96.5	9.74	0.43	180	104.52	15.36	0.48
B12	174	190.21	2.25	0.32	225	190.21	27.13	0.3	184	205.87	37.74	0.43
B13	165	165.91	3.99	0.83	190	165.91	2.16	0.6	139	169.11	0.46	-0.21
B14	235	236.08	2.21	0.17	221	236.08	3.19	0.81	180	241.43	1.81	0.09
B15	318	318.63	1.06	-0.06	308	318.63	1.35	0.72	194	339.33	9.87	-0.32
B16	127	132.53	7.46	0.44	291	132.53	6.61	0.44	238	137	7.88	0.32
B17	131	137.44	4.96	0.5	219	137.44	6.69	0.35	180	145.25	16.94	0.2
B18	218	219.57	2.53	0.12	425	219.57	1.88	0.21	348	221.67	3.28	0.25
Avg	140.11	143.70	3.30	0.34	237.61	144.17	5.58	0.48	187.89	149.91	8.64	0.22

Table 5.1 presents the average, the standard deviation, denoted by σ**,** of the multicast tree cost, and the fitness distance coefficient ρ for each instance. Based on the classification described above, in the table, for

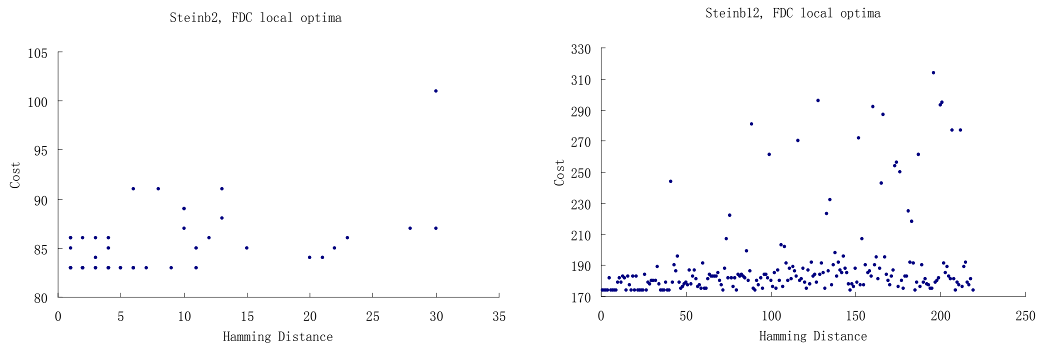
the DCLC multicast routing problem with $\Delta_1 = \infty$, four instances (B03, B09, B15 and B18) show to be difficult. One instance B01 presents to be a misleading problem, and others are relatively easy to solve, i.e. the fitness and distance to the optimum are correlated. For the DCLC multicast problem with a slightly tighter delay bound $\Delta_2=1.1\times Delay(T_{OPT})$, two instances B03 and B09 remain to be difficult problems, while other instances turn to be relatively easier (where $\rho \geq 0.15$). For the DCLC multicast problem with the tightest delay bound $\Delta_3=0.9\times Delay(T_{OPT})$, two instances B09 and B14 are difficult problems, three instances B03, B13 and B15 are misleading, while the others are easy instances.

From the experimental results, we can see that the fitness landscape for the DCLC multicast routing problem is instance dependant with different level of difficulties for search algorithms. In addition, different delay bounds will affect the landscapes of the same network topology. Note that even only two instances remain to be difficult for problem with delay bound Δ_2, it might not be easy for search algorithms to find better solutions for the other instances with $\rho \geq 0.15$, as the delay bound restricts the search within the search space with disconnected areas. It means that the introduction of the delay bound Δ_2 results in different distribution of solutions in the search space. This can be seen in Table 5.1, that for some instances, the cost of the best multicast tree obtained by LS for the DCLC multicast routing problem with tighter delay bounds is higher than that of the problem of the same topology with no delay bound. Generally, the smaller the delay bounds, the tighter the constraints to the problem. This is shown by the worst average tree cost 149.91 over the 18 instances obtained for problem with delay bound Δ_3, compared 144.17 for the delay bound Δ_2 and 143.7 for the delay bound Δ_1.

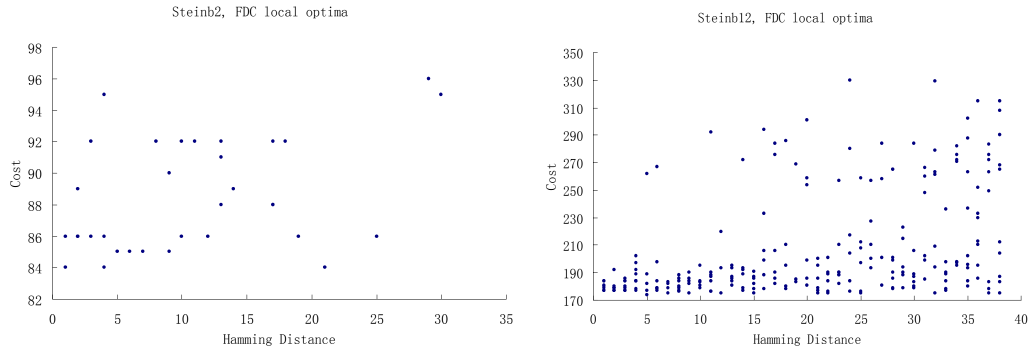
In addition to the values of the FDC coefficient, a fitness distance scatter plot can provide more insightful information of the landscape. Figure 5.2 presents some example scatter plots of fitness and distance for two instances with different delay bounds. The fitness distance scatter plots for the two instances clearly demonstrate the changes of the fitness distance landscape on problem instances with different delay bounds. For instance B02, the correlation between the cost and the distance decreases from 0.71 for Δ_1, to 0.51 for Δ_2 and 0.4 for Δ_3. It means for instance B02, the tighter the delay bound, the lower the ρ value, thus the higher level of difficulty. For instance B12, although positive correlations with a ρ value larger than 0.15 have been obtained, all scatter plots show that a large number of local optima exist in the landscapes. Especially, for B12 with delay bounds Δ_2 and Δ_3, we can see many shallow valleys near the optimum, showing that many local optima are crowed near the optimum or best-known solution. This makes the search to be easily trapped in local optima.



(a) fitness-distance scatter plots of local optima on two instances with $\Delta_1 = \infty$.



(b) fitness-distance scatter plots of local optima on two instances with $\Delta_2 = 1.1\times Delay(T_{OPT})$.



(c) fitness-distance scatter plots of local optima on two instances with $\Delta_3 = 0.9\times Delay(T_{OPT})$.

Figure 5. 2 Fitness-distance scatter plots on selected instances with different delay bounds

5.3.1.2 Impact of Different Local Search Methods on the FDC analysis

Based on the above basic analysis on fitness distance correlation, the fitness landscape of different local search methods on the DCLC multicast routing problem is then analysed. To observe the impact of different local search methods on the FDC landscape, three other local search heuristics, namely BSMA (Zhu *et al*., 1995), TS-CST (Skorin-Kapov and Kos, 2006) and VNDMR2 (Qu *et al*., 2009) in the literature, are applied for the DCLC multicast routing problem. The brief description of the three local search algorithms is given as follows:

1) The BSMA heuristic

As reviewed in Section 3.4.3, BSMA is a well known deterministic algorithm for the DCLC multicast routing problem developed in the mid 1990s. In BSMA, a delay-bounded path switching operator has been devised to replace a *superedge* (see the definition in Section 3.4.3) in the multicast tree by a new alternative path, hopefully resulting in a new delay-bounded tree with lower tree cost. At each step, one chosen *superedge* is removed, leading to two sub-trees. The Dijkstra's shortest path algorithm is then consecutively called to find a new delay-bounded shortest path that connects the two sub-trees and reduce the tree cost. The BSMA heuristic refines the tree iteratively to reduce costs until the tree cost cannot be further reduced.

2) The TS-CST algorithm

The TS-CST algorithm is developed based on a tabu search metaheuristic, where a binary vector solution representation has been used (and adapted in this study here, see Section 5.3.1 above). Similar to the presentation in this study, a solution (a multicast tree) in TS-CST is represented by a binary string consisting of $|V \setminus (\{s\} \cup R)|$ bits. Each bit corresponds to a node in $V \setminus (\{s\} \cup R)$. Nodes whose corresponding bits have a value of zero in the binary set represent the Steiner nodes in the given multicast solution tree. Nodes with their corresponding bits value set to 1 mean that they are not included in the multicast solution. Neighbourhood of a solution includes all the solutions which are exactly one bit different in the binary set from the chosen solution. In other words, the neighbouring solutions are all those multicast trees generated by adding or removing exactly one node in $V \setminus (\{s\} \cup R)$. The Prim's algorithm is applied to generate a new delay-constrained minimum spanning tree of the given set of nodes. The best new neighbouring solution is chosen as the current solution in the next iteration, i.e. a move in the TS-CST algorithm. In order to prevent the algorithm from oscillating between neighbouring solutions in TS-CST, a tabu list of length one is repeatedly updated to remember the corresponding bit of the last performed move. The process stops after a pre-defined number of iterations without improvements, set to 2 in TS-CST.

3) The VNDMR2 algorithm

VNDMR2 is a variable neighbourhood descent search algorithm developed in the previous work (see Section 4.2.1.2) for the DCLC multicast routing problem. Three neighbourhoods, one is node-based and the other two are link-based neighbourhoods, have been designed concerning the structure of the multicast tree. The node-based neighbourhood is similar to the neighbourhood designed in TS-CST. A neighbouring tree defined in the node-based neighbourhood is obtained by removing or adding a Steiner node from or to the current multicast tree, and creating a minimum spanning tree which spans the nodes by using the Prim's algorithm. Once a better multicast tree is found, the current solution is updated. This procedure is repeated until no improvement can be achieved for more than 3 iterations. The two link-based neighbourhoods are designed based on a path replacement operator which is similar to the path switching operator in BSMA. The path replacing operator replaces a *superpath* (also called *superedge* in BSMA) by a new delay-bounded path with a lower cost by using the *k*-shortest path algorithm. Both of the two link-based neighbourhoods iteratively refine the multicast tree to lower costs until no better tree can be found. The three neighbourhoods have been designed to reduce the tree cost while satisfying the delay bound constraint in the problem.

The FDC analysis of the landscape from different local search methods for four instances (B02, B10, B12, and B18) of different sizes and delay bounds is presented in Table 5.2. The average tree cost, the standard deviation σ of the tree costs and the FDC coefficient ρ obtained by the four local search methods on these four instances are given in the table.

Table 5. 2 Fitness distance correlation analysis of the landscape from four local search methods on four instances of different sizes and delay bounds (The bold number denotes the best result)

Δ	No.	LS			BSMA			TS-CST			VNDMR2		
		cost	σ	ρ	**cost**	σ	ρ	**cost**	σ	ρ	**cost**	σ	ρ
Δ_1	**B02**	86.6	4.89	0.71	85.86	4.23	0.69	87.32	6.6	0.69	**83.14**	**0.4**	0.71
	B10	91.13	5.63	0.44	93.83	6.39	0.65	88.65	4.25	0.47	**86.89**	**1.73**	0.71
	B12	175.91	2.25	0.32	186.22	5.67	0.09	174.13	0.73	0.2	**174.01**	**0.1**	0.09
	B18	219.77	2.53	0.12	244.31	8.17	0.17	**218.26**	0.57	0.52	218.53	**0.71**	0.49
Δ_2	**B02**	85.23	3.34	0.51	85.4	3.39	0.54	85.73	5.23	0.54	**83.19**	**0.85**	0.54
	B10	91.08	5.65	0.46	94.16	6.57	0.64	88.52	4.13	0.61	**86.49**	**1.28**	0.36
	B12	190.21	27.13	0.3	187.44	6.18	0.17	**174.02**	**0.15**	0.28	175.83	2.3	0.27
	B18	219.57	1.88	0.21	242.38	7.67	0.04	**218.29**	0.67	0.54	218.3	**0.66**	0.52
Δ_3	**B02**	88.45	3.54	0.4	86.12	3.62	0.54	86.52	4.28	0.66	**83.52**	**1.86**	0.61
	B10	92.32	6.41	0.24	92.99	4.76	0.32	88.77	4.17	0.65	**86.74**	**1.83**	0.5
	B12	205.87	37.74	0.43	186.93	5.35	0.08	**174.24**	**0.94**	0.37	177.67	**0.94**	-0.38
	B18	221.67	3.28	0.25	243.7	7.27	0.18	**218.18**	**0.42**	0.48	218.27	0.56	0.47

Different local search methods applied in the fitness landscape analysis result in different ρ values as shown in Table 5.2. VNDMR2 has the best overall performance by finding the best average tree costs on 7 out of 12 cases. It means VNDMR2 is more effective to guide the search to better solutions compared with the other three local search methods. VNDMR2 is also highly stable with respect to the standard deviation σ on 10 out of 12 instances. The average σ over the 12 instances of VNDMR2 is the smallest (1.10) among the four local search algorithms, compared with 8.69 for LS, 5.77 for BSMA, and 2.68 for TS-CST.

In most of the cases, local search with higher ρ values indicates to be suitable for solving the problem, because a good correlation between fitness and distance can be found. However, some exceptions can be seen such as for instance B12, landscape of VNDMR2 has a lower ρ value than that of TS-CST, probably due to that the landscape near the global optimum is more rugged with many local optima, as shown in Figure 5.2. For the simple local search method LS, although it can obtain a higher ρ on some instances, most of the solutions found by LS have lower quality with higher fitness values compared with the other three algorithms.

5.3.2 Experimental Results of Auto-correlation Analysis

Another important measure of landscape characteristic is its ruggedness. To analyse the ruggedness of the landscape of the DCLC multicast routing problem, the auto-correlation analysis is performed on the same benchmark Steinb instances. A random walk of $W = 1000$ steps has been conducted on each instance. The random walk starts from a random multicast tree. The random multicast tree is generated by starting from the source node and randomly selecting the next connected node until a destination node is added. This procedure repeats until all destination nodes have been added in the tree. Different neighbourhood operators may be applied to perform the auto-correlation analysis for the DCLC multicast routing problem. Firstly, a pure random one-flip neighbourhood operator is designed.

5.3.2.1 Auto-correlation Analysis based on the Random One-flip Neighbourhood

Based on the binary vector representation, the one-flip neighbourhood is defined as randomly flipping a Steiner node in the binary vector of the current solution and then generating a random tree with the given nodes set, i.e. nodes represented by a value of 1 at the corresponding bit in the binary vector. The random tree is generated by using the similar procedure described above to generate the starting point of the random walk. The difference is that only the nodes included in the binary vector can be selected. This procedure repeats until a feasible tree that satisfies the delay constraint is generated. The random walk is conducted by using the one-flip neighbourhood.

Table 5.3 presents the experimental results of the auto-correlation analysis by using the pure random one-flip neighbourhood to generate random walks on the Steinb instances with the same three different delay bounds as used in the FDC analysis. Both the correlation length l, calculated according to Eq. 5.3, and the correlation length in relation to the diameter of the landscape n / l, here $n = diamG_L$, have been calculated.

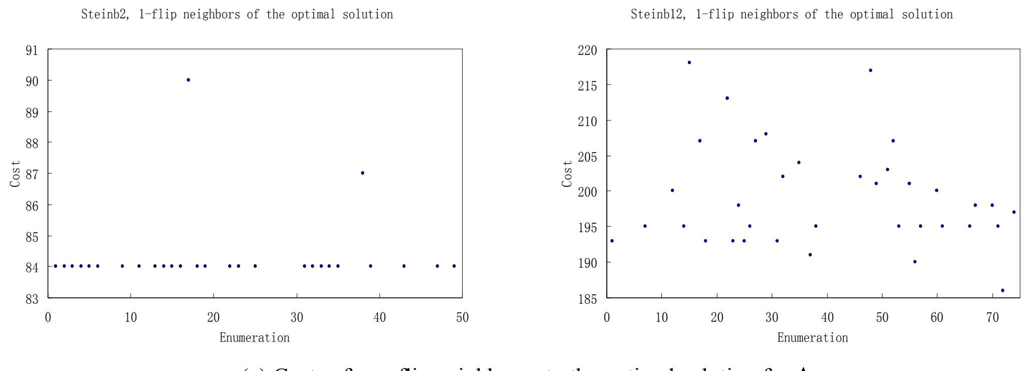
Table 5. 3 The auto-correlation analysis on the Steinb instances with different delay bounds ($|V|$ is the network size; $|R|$ is the multicast group size; the landscape diameter $n = diamG_L = |V| - |R| - 1$)

No.	$\|V\|$	$\|R\|$	n	Δ_1		Δ_2		Δ_3	
				l	n / l	l	n / l	l	n / l
B01	50	9	40	0.61	16.39	0.61	16.39	3.58	11.17
B02	50	13	36	0.35	102.86	0.19	189.47	0.63	57.14
B03	50	25	24	0.26	92.31	0.21	95.24	0.18	133.33
B04	50	9	40	0.54	74.07	0.57	70.18	0.57	70.18
B05	50	13	36	0.26	138.46	0.43	83.72	0.88	40.91
B06	50	25	24	0.4	60	0.46	52.17	0.4	60
B07	75	13	61	0.22	277.27	0.22	277.27	0.32	190.63
B08	75	19	55	0.34	161.76	0.36	152.78	0.56	98.21
B09	75	38	36	0.31	116.13	0.21	171.43	0.33	109.09
B10	75	13	61	0.44	138.64	0.41	148.78	0.42	145.24
B11	75	19	55	0.37	148.65	0.28	196.43	0.35	157.14
B12	75	38	36	0.34	105.88	0.33	109.09	0.28	128.57
B13	100	17	82	0.41	200	1.46	56.16	0.33	248.48
B14	100	25	74	0.36	205.56	0.41	180.49	0.46	160.87
B15	100	50	49	0.27	181.48	0.4	122.5	0.45	108.89
B16	100	17	82	0.46	178.26	0.27	303.7	0.37	221.62
B17	100	25	74	0.41	180.49	0.29	255.17	0.31	238.71
B18	100	50	49	0.26	188.46	0.29	168.97	0.27	181.48

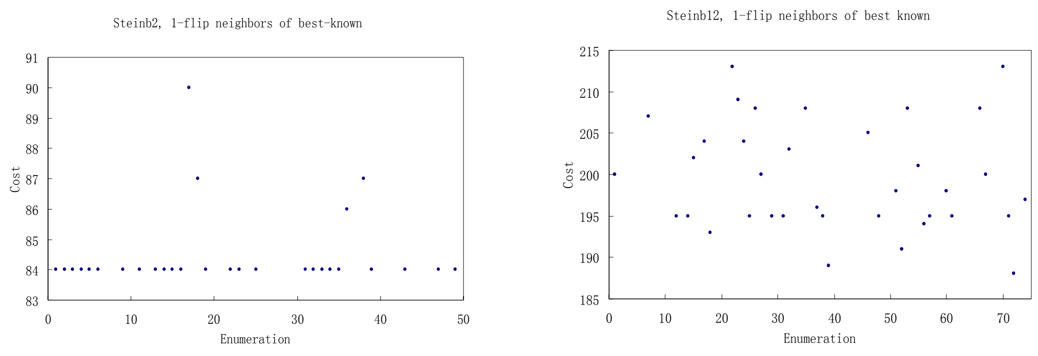
The ruggedness of a fitness landscape is determined by the correlation of two neighbouring points in the search space. The higher the correlation length l (or the lower the correlation length to the diameter of the landscape n / l), the smoother the landscape and thus the easier the search of an algorithm based on the underlying neighbourhood. From Table 5.3, we can see that although the correlation length is also problem instance dependent, the landscape of all the instances with different delay bounds are highly rugged with small l (most values of l are less than 1). For $\Delta_1 = \infty$, the problem is reduced to the unconstrained Steiner tree problem, the smoothest landscape corresponds to instance B01 which has a highest correlation length l

as expected by a lowest *n* / *l*. Some instances such as B03, B05, B07, B15 and B18 have relatively lower correlation lengths which indicate rugged landscapes for these instances. For $\Delta_2 = 1.1\times Delay(T_{OPT})$, instances B01, B06, B13 have smooth landscapes as reflected by relatively high correlation lengths. The landscapes of instances B02, B03, B07, B09, B11, B16, B17 and B18 are more rugged due to the relatively lower correlation lengths with higher *n* / *l*. The correlation lengths obtained for $\Delta_3 = 0.9\times Delay(T_{OPT})$ show different landscapes of the instances. Instance B01 is the smoothest landscape which has the highest correlation length, while instance B03 is the instance with the lowest correlation length which shows the ruggedness of the landscape for B03. Experimental results in the table show that the delay bounds will affect the ruggedness of landscapes on problem instances as varied correlation lengths are obtained from the same neighbourhood operator.

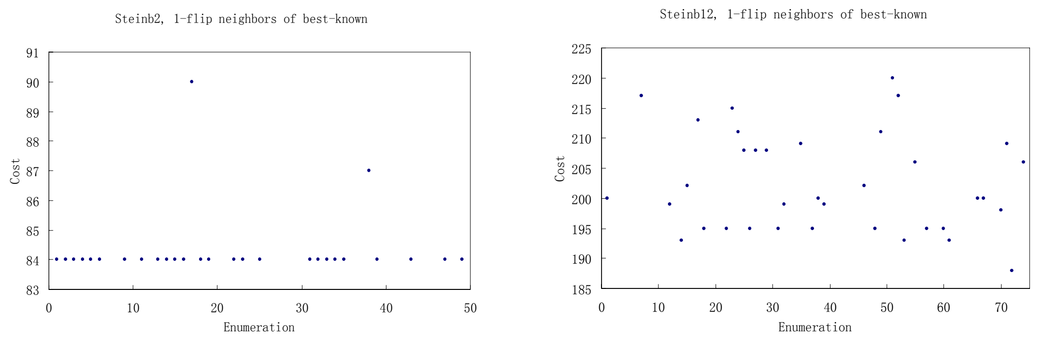
To visually observe the landscape structure of the DCLC multicast routing problem, the neighbours to the optimum or best-known solution of the Steinb instances are ploted by using the same one-flip operator in the random walk. Figure 5.3 shows the example plots of the cost of all these one-flip neighbours to the optimal solution or the best-known solution in the search space. Infeasible neighbours are not shown and an empty space appears at the corresponding positions. The plots in Figure 5.3 again illustrate that the ruggedness of landscapes is highly dependent on the specific instances. It is obvious that the landscape of B12 with different delay bounds is more rugged than that of B02, as small changes in the solution make a huge difference in the cost of the solutions for B12. The landscape of B02 is smoother than B12 as most of the one-flip neighbours have the same cost located in a plateau.



(a) Costs of one-flip neighbours to the optimal solution for $\Delta_1 = \infty$.



(b) Costs of one-flip neighbours to the optimal solution for $\Delta_2 = 1.1\times Delay(T_{OPT})$.



(c) Costs of one-flip neighbours to the best-known solution for $\Delta_3 = 0.9\times Delay(T_{OPT})$.

Figure 5. 3 Scatter plots of costs of the one-flip neighbours to the optimum or best solution on two instances

5.3.2.2 Impact of Different Neighbourhood Operators on the Auto-correlation Analysis

The correlation length as a measure of landscape ruggedness can be used to compare different neighbourhoods employed in local search algorithms designed for the DCLC multicast routing problem. The higher the correlation, the smoother the landscape defined, and the more effective the local search. In addition to the pure random neighbourhood designed above, the auto-correlation analysis is carried out to evaluate two more neighbourhood operators, a node-based neighbourhood and a link-based neighbourhood for the DCLC multicast routing problem, both neighbourhood operators have been used in the previous VNDMR2 algorithm.

1) The Link-based Neighbourhood Operator

The link-based neighbourhood operates on links in the network. It replaces a randomly selected *superpath* in the current tree with a feasible alternative path (which satisfies the delay bound), generated by using the *k*-shortest algorithm.

2) The Node-based Neighbourhood Operator

The node-based neighbourhood operates on nodes in the network. At each step, it randomly flips a Steiner node in the binary vector of the current tree and uses the Prim's algorithm to generate a minimum spanning tree on the given nodes in the binary vector. This operator repeats until a new feasible tree is generated.

Table 5. 4 The auto-correlation analysis of three neighbourhood operators on four instances in Steinb with different delay bounds ($|V|$ is the network size; $|R|$ is the multicast group size; the landscape diameter $n = diamG_L = |V| - |R| - 1$. *pure*: the pure random one-flip operator; *link*: the link based operator; *node*: the node based operator)

No.	$\|V\|$	$\|R\|$	n	Metrics	Δ_1			Δ_2			Δ_3		
					pure	*link*	*node*	*pure*	*link*	*node*	*pure*	*link*	*node*
B02	50	13	36	l	0.35	1.19	**2.02**	0.19	1.06	**1.65**	0.63	1.31	**3.7**
				n / l	102.86	30.25	**17.82**	189.47	33.96	**21.82**	57.14	27.48	**9.73**
B10	75	13	61	l	0.44	1.66	**2.53**	0.41	1.54	**2.16**	0.42	1.71	**1.57**
				n / l	138.64	36.75	**24.11**	148.78	39.61	**28.24**	145.24	35.67	**38.85**
B12	75	38	36	l	0.34	**1.98**	0.52	0.33	**2.04**	0.54	0.28	**1.88**	0.81
				n / l	105.88	**18.18**	69.23	109.09	**17.65**	66.67	128.57	**19.15**	44.44
B18	100	50	49	l	0.26	**2.91**	0.62	0.29	**2.73**	0.6	0.27	**2.74**	1.1
				n / l	188.46	**16.84**	79.03	168.97	**17.95**	81.67	181.48	**17.88**	44.55

For a fair comparison, the same maximum number of iterations *maxiter* (here *maxiter* = 5) has been set for the three different neighbourhood operators when searching the neighbouring solutions. The autocorrelation analysis has been carried out based on these neighbourhoods for solving four instances (B02, B10, B12, and B18) of different sizes from Steinb. Table 5.4 presents the results of the correlation length l and the correlation length in relation to the diameter of the landscape n / l obtained by the three neighbourhood operators for the four instances with different delay bounds.

As shown in Table 5.4, the link-based neighbourhood and node-based neighbourhood increase the smoothness of the landscape. Longer correlation lengths have been obtained by using these two neighbourhood operators designed with regard to network structures compared with the pure random neighbourhood. This means that both the link-based and node-based neighbourhoods are more effective than the pure random neighbourhood in the local search algorithms for the DCLC multicast routing problem. For two instances (B02 and B10) with three different delay bounds, the node-based neighbourhood found the smoothest landscapes as reflected by the longest l and lowest n / l in the table.

While for the other two instances (B12 and B18), the landscapes defined by the link-based neighbourhood operator are smoother than that of node-based neighbourhood operator. This demonstrates that effective local search algorithms can be defined based on different neighbourhoods for solving DCLC multicast routing problems with different characteristics. This is consistent with the better performance observed in the previous work on the variable neighbourhood descent search algorithm VNDMR2 for the DCLC multicast routing problem (Qu *et al*., 2009). VNDMR2 obtained better solutions compared with existing heuristics and algorithms employing single neighbourhood operator.

5.4 Summary

In this chapter, to measure the landscape features of the DCLC multicast routing problem, two of the most commonly used landscape analysis techniques, the fitness distance correlation analysis and the auto-correlation analysis, have been applied to the problem for the first time. Extensive experiments on a set of problem instances generated based on the benchmark Steiner tree problems in the OR-library reveal that the landscape of the DCLC multicast routing problem is highly instance dependent with different landscape characteristics can be observed. The delay constraint has shown a great influence on the underlying landscape features of the problem. In addition, the impact of different local search methods on the analysis of fitness distance landscape is investigated and the effectiveness of different neighbourhood operators in the auto-correlation analysis is compared for solving the problem of different sizes and delay bounds. According to the analysis, both the node-based neighbourhood and the link-based neighbourhood operators are effective and suitable for solving the DCLC multicast routing problem. The fitness landscape analysis techniques discussed in this paper have shown to be useful for analyzing the underlying properties of the DCLC multicast routing problem and predicting the effectiveness of local search algorithms and neighbourhood operators on the problem.

CHAPTER 6. Metaheuristic Approaches for the Multi-objective Multicast Routing Problem

6.1 Introduction

A range of inter-dependent and different conflicting QoS objectives and constraints (e.g. the cost, delay, bandwidth, link utilisation, delay variation, packet lost ratio and hop count) exist in real world applications. As a result, the QoS multicast routing problem can be more appropriately considered within the multi-objective optimisation context with multiple objectives to be optimised simultaneously. In recent research, some multi-objective multicast routing algorithms have emerged to tackle these complex QoS multicast routing problems with more realistic features. With the recent development in metaheuristics, some advanced algorithms have been proposed for solving a variety of multi-objective optimisation problems in the literature. In this chapter, two novel metaheuristic approaches are investigated for solving the multi-objective multicast routing problem defined in Chapter 3.

6.2 An Evolutionary Multi-objective Simulated Annealing Algorithm with Variable Neighbourhoods

This section presents the investigation of an evolutionary multi-objective simulated annealing algorithm with variable neighbourhoods to solve the multi-objective multicast routing problems in telecommunications. The hybrid algorithm aims at a more flexible and adaptive exploration over the search space by using features of the variable neighbourhood search to find more non-dominated solutions in the Pareto front.

SA, one of the mostly studied metaheuristics, has the ability of escaping from local optima, which makes it a very effective metaheuristic when exploring the search space of complex combinatorial optimisation problems. As reviewed in Chaper 2, various multi-objective algorithms based on simulated annealing have been investigated by researchers for combinatorial optimisation problems. Although several

SA-based algorithms have been proposed for the single objective QoS multicast routing problems in the literature (e.g. Wang and Jiang, 2004; Zhang *et al*., 2005; and Zhang *et al*., 2009), to my knowledge, no research has been carried out by using multi-objective simulated annealing approach for solving the multi-objective multicast routing problem. In a recent work (Li and Landa-Silva, 2008), a variant of SA-based algorithm named Evolutionary Multi-Objective Simulated Annealing (EMOSA) has been designed and tested on a set of benchmark TSP instances. A two-phase strategy in the EMOSA algorithm which tunes search directions and the competition between similar solutions have shown to be effective in comparison with other multi-objective simulated annealing algorithms. Based on their work, in this section, the first EMOSA algorithm is developed to solve the multi-objective multicast routing problem. In addition, due to the specific features of the multicast tree, which are quite different from those of TSP problems, different neighbourhood structures have been designed with respect to different objectives to improve the proposed algorithm. Thus, EMOSA is further extended by employing variable neighbourhoods with respect to different objectives, namely VEMOSA (variable neighbourhood based EMOSA), where four objectives are optimised simultaneously without a priori restriction, aiming to minimise (1) the cost of the multicast tree (Eq. (3.3)), (2) the maximal end-to-end delay (Eq. (3.4)), (3) the maximal link utilisation (Eq. (3.5)), (4) the average delay (Eq. (3.6)), and subject to a link capacity constraint (Eq. (3.8)). Compared with the traditional metaheuristics with single neighbourhood structure, the proposed VEMOSA algorithm is able to optimise different objectives simultaneously, contributing to advanced multi-objective search algorithms for a wider range of optimisation problems with different features in the future.

6.2.1 Algorithm Description

The EMOSA algorithm developed by Li and Landa-Silva (2008) has shown to be effective for solving the TSP problem. In the EMOSA algorithm, a population of solutions evolves by using strategies of the cooling schedule in the simulated annealing algorithm. The main features of EMOSA are described as follows:

1) The Weighted Scalarising Function in the EMOSA Algorithm

As mentioned in Chapter 2, one general technique in multi-objective optimisation is to combine different objectives into a single composite function. In EMOSA, a weighted scalarising function is used to evaluate the fitness of solutions. A Pareto-optimal solution x^* is obtained if it is the unique global minimum of the following scalar optimisation problem:

$$\text{Minimise } g^{(ws)}(x, \lambda) = \sum_{i=1}^{m} \lambda_i f_i(x) \text{ s.t. } x \in X \tag{6.1}$$

where $g^{(ws)}(x, \lambda)$ is called the weighted scalarising function of solution x, λ is a weight vector, $\lambda_i \in [0, 1]$, $i = 1, \ldots, m$ and $\sum_{i=1}^{m} \lambda_i = 1$, i.e. the weights are uniformly assigned to m objectives, each λ_i is associated with one objective function $f_i(x)$, and X denotes the decision space of the set of feasible regions of the solution space.

2) The Acceptance Criterion in the EMOSA Algorithm

In EMOSA, the acceptance decision of a new solution has to take into account all objectives in the problem. The acceptance probability in the standard SA has been modified by using the weighted scalarising function $g^{(ws)}(x, \lambda)$ defined in (Eq. (6.1)). Given an incumbent solution x and its neighbourhood $N(x)$, the probability of x moving to its neighbour $x' \in N(x)$ is thus defined as follows:

$$P(x, x', \lambda, T) = \begin{cases} 1 & \text{if } \Delta g(x, x', \lambda) < 0 \\ e^{\frac{-\Delta g(x, x', \lambda)}{T}} & \text{otherwise} \end{cases} \tag{6.2}$$

where $T > 0$ is the current temperature, $\Delta g(x, x', \lambda) = g^{(ws)}(x', \lambda) - g^{(ws)}(x, \lambda)$ is the difference of the weighted scalarising function values between x and x' defined by Eq. (6.1).

3) The Competition between Similar Members in the Current Population

The competition between similar members of the current population in EMOSA is managed by comparing their weighted scalarising function values (see Eq. (6.1)). During the evolution of EMOSA, a neighbouring solution x' replaces similar solutions in the current population if it is better than its similar solutions with respect to the weighted scalarising function. The similarity between solutions is measured by the Euclidean distance between the weight vectors of two solutions.

4) Two-phase Strategy in the EMOSA Algorithm

In addition to defining a starting temperature, a final temperature, and a temperature decrement in the cooling schedule, in EMOSA, a threshold temperature TP_c is also defined within a two-phase strategy for tuning the weight vectors. The two-phase strategy uses both fixed (the first phase, $T \geq TP_c$) and adaptive (the second phase, $T < TP_c$) search directions during the search in order to explore the multi-objective space more effectively. In the first phase, all weight vectors remain unchanged. In the second phase, the weight vectors are adaptively changed according to the closeness of solutions to their non-dominated neighbour in the population.

Based on the EMOSA algorithm described above, the EMOSA algorithm is firstly applied for solving the multi-objective multicast routing problem. Furthermore, a new algorithm called VEMOSA is proposed, which is an extension of the EMOSA algorithm by designing variable neighbourhoods for the multiple objectives in the problem.

6.2.1.1 The Overall Procedure of VEMOSA

The VEMOSA algorithm starts with an initial population of random solutions T_1, …, T_{pop}. Each solution T_i is a multicast tree which is generated by starting from the source node and randomly connecting the next node until all destination nodes are mounted on the tree. A random weight vector λ^i is assigned to each solution T_i. An external non-dominated solution set is maintained to store non-dominated solutions during the evolution.

During the evolution, for each multicast tree T_i in the population, a neighbouring tree T' is generated by using a randomly chosen neighbourhood structure N_ℓ, where $\ell \in \{1, ..., \ell_{max}\}$ is a set of pre-defined neighbourhood structures and each neighbourhood concerns one objective. The non-dominated solution set is updated if T' is not dominated by the current tree T_i. The neighbouring tree is accepted as the current tree with a probability $P(T_i, T', \lambda^i, TP)$ based on the current temperature TP and the weight vector λ^i, defined by the adopted acceptance criterion in SA, see Eq. (6.2). Note that in VEMOSA, if the number of neighbourhood structures $\ell_{max} = 1$ then it becomes the EMOSA algorithm with one neighbourhood structure.

```
VEMOSA ($m$, $\ell_{max}$, $TP_{max}$, $TP_{min}$, $TP_c$)
 { // $m$: number of objectives; $\ell_{max}$: the number of neighbourhood structures;
  // $TP_{max}$ / $TP_{min}$: starting / final temperatures; $TP_c$: threshold temperature for tuning weight vectors;
  // $CS$: current solution set; $NDS$: the set of non-dominated solutions
  Initialization:
  Create the initial population $CS$ by randomly generating $T_1, \ldots, T_{pop}$ multicast trees; // $pop$: population size
  Produce $pop$ distinct random weight vectors $\lambda^1, \ldots, \lambda^{pop}$, each vector with uniform spread;
  Form the external non-dominated solution set $NDS$ by non-dominated solutions in $CS$;
  Set the current temperature $TP = TP_{max}$;
  while $TP \geq TP_{min}$ do {
      foreach $T^i \in CS$ do {
          $lm = 0$;
          while $lm < LM$ do { // $LM$: number of local search moves between two consecutive temperatures
              Generate a neighbour $T' \in N_\ell(T_i)$ by a random $\ell \in \{1, ..., \ell_{max}\}$, set $lm = lm + 1$;
              if $T^i$ does not dominate $T'$ then
                 Update the external $NDS$;
              Replace the current solution $T_i$ by $T'$ with the probability $P(T_i, T', \lambda^i, TP)$; // see formula 6.2
              // competition between similar members, see section 6.2.1
              Find the closest solution $T^j \in CS$ ($T_j \neq T_i$) of $T_i$;
              if $g(T', \lambda^j) < g(T_j, \lambda^j)$ then
                  $T_j = T'$; //replace the closest solution in $CS$
          end of while loop }
      }
      $TP = TP - \alpha$; // $\alpha$: temperature decrement
      // search direction adaptation: tune the search direction by adjusting the weight vectors, see section 6.2.1
      if $TP < TP_c$ then
      foreach $T^i \in CS$ do {
          Find the closest non-dominated solution $T$ of $T_i$ from $CS$
          foreach $obj \in \{1, \ldots, m\}$ do {
              if $f_{obj}(T) < f_{obj}(T^i)$ then $\lambda^i_{obj} = \mu\, \lambda^i_{obj}$; // $\mu$: the constant for tuning the search direction
              else $\lambda^i_{obj} = \lambda^i_{obj} / \mu$; }
          }
          Normalise $\lambda^i$ by setting $\lambda^i_{obj} = \lambda^i_{obj} / \sum_{p=1}^{m} \lambda^i_p$ , $obj = 1, \ldots, m$;
  end of while loop }
  return the non-dominated solution set $NDS$;
}
```

Figure 6. 1 The framework of the VEMOSA algorithm

In VEMOSA, the competition between the similar members of current population is adopted, where each newly generated neighbouring tree replaces its closest member in the current population if it is better with respect to the weighted scalarising function in Eq. (6.1). To maintain a population of diverse solutions, the two-phase strategy is used to adaptively tune the search direction at the later stage of the evolution. When the current temperature is decreased to below a given threshold, the weight vectors of each solution are adaptively adjusted to tune the search directions near the Pareto front. The VEMOSA algorithm stops either when the temperature drops to the final temperature or after a given period of time. The framework of the proposed VEMOSA algorithm is presented in Figure 6.1.

6.2.1.2 The Variable Neighbourhood Structures in VEMOSA

Compared with other single neighbourhood based metaheuristics, VNS is more flexible and effective due to its ability of escaping from local optima by traversing among different areas of the search space defined by different neighbourhood structures. In multi-objective optimisation, little research attention has been given on applying VNS to solve multi-objective optimisation problems (e.g. Liu *et al.*, 2006; Adibi and Zandieh, 2010).

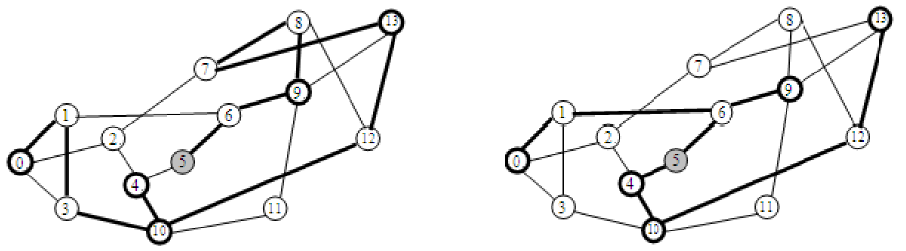
For solving multi-objective problems, two important research issues concerned are the diversification and intensification of the search for Pareto optimal solutions. In the current literature, almost all MOSA algorithms use only one neighbourhood structure. In this research, the effectiveness of the appropriately defined neighbourhood structures for different objectives is investigated on the multi-objective multicast routing problem. The motivation is that traditional algorithms with a single neighbourhood structure are not able to equally intensify different objectives simultaneously. Some of the Pareto optimal solutions for a particular objective thus may be missed in the search space of complex multi-objective problems. The EMOSA algorithm is integrated with a set of neighbourhood structures during the search process. Two variants of EMOSA algorithms have been developed based on Li and Landa-Silva's EMOSA algorithm, namely 1) EMOSA with a single neighbourhood structure; and 2) VEMOSA with a set of variable neighbourhood structures. Both algorithms represent new approaches for solving the multi-objective multicast routing problem in the literature.

In VEMOSA, two types of perturbation operators have been devised as the transition mechanism to generate neighbouring solutions.

- **The Path-switching Operator**. This type of neighbourhood operator operates on paths in the multicast tree. A backup-path-set of *k*-best paths for each objective is constructed from the source node *s* to each destination with respect to each objective by using the *k*-shortest path algorithm (Eppstein, 1998). For *m* objectives, there are maximum $m \times k$ paths in the backup-path-set for each destination (where $m = 4$ as mentioned at the beginning of Section 6.2. and formulated in chapter 3, $k = 25$), see Figure 6.3 for an example. Note that *k* is the maximum number of shortest paths from the source to each destination since some destinations may have less than 25 shortest paths. In the algorithms, the same *k*-best paths in terms of delay are kept for two delay related objectives (2) and (4), i.e. objectives defined by (Eq. (3.4)) and (Eq. (3.6)). To generate a neighbouring tree, a destination node is randomly chosen and the path from the source to the destination in the current tree is replaced by a random path in the backup-path-set of the chosen destination.

- **The Node-switching Operator**. This neighbourhood operator operates on nodes in the multicast tree. A neighbouring tree is generated by deleting a random node (excluding the source and destination nodes) in the current tree. A minimum spanning tree is then generated by using the Prim's algorithm (Betsekas and Gallager, 1992) with the remaining nodes in the tree. The Prim's algorithm is a well known algorithm in graph theory that finds a tree which spans a subset of links and all the nodes in a graph, where the total weight of all the links in the tree is minimised.

In EMOSA with a single neighbourhood, the neighbourhood structure is based on the path-switching operator. In VEMOSA, besides the neighbourhood structure defined in EMOSA, four additional neighbourhood structures based on both the path-switching and the node-switching operators are devised for each objective. By employing different neighbourhood structures specifically target at different objectives, VEMOSA is expected to make intensified search while optimising each objective simultaneously.



(a) An example of random initial solution
C(T) = 10.8, DM(T) = 87, α(T) = 0.6, DA(T) = 48.8

(b) An example of an optimal solution
C(T) = 7.4, DM(T) = 36, α(T) = 0.6, DA(T) = 22.2

Figure 6. 2 Two example solutions (multicast trees) for the NSF network

ID	Paths
1	5-6-1-0
2	5-4-2-0
3	5-6-9-11-10-3-0
4	5-6-1-3-0
5	5-4-10-3-0
(a) least cost paths	

ID	Paths
1	5-4-2-0
2	5-6-1-0
3	5-4-10-3-0
4	5-4-2-7-8-9-6-1-0
5	5-6-9-8-7-2-0
(b) least delay paths	

ID	Paths
1	5-4-10-3-0
2	5-6-1-0
3	5-4-2-0
4	5-4-10-11-9-6-1-0
5	5-4-10-3-1-0
(c) least utilised paths	

Figure 6. 3 An example of the backup-path-set by using the path-switching operator on the NSF network

For the ease of understanding, a random multicast tree and an example of optimal solution are presented in Figure 6.2(a) and Figure 6.2(b) of the benchmark NSF network as shown in Figure 3.2 in Section 3.5.4. To generate a neighbouring tree, a set of backup-paths of destination $r_i = 0$ is shown in Figure 6.3, i.e. the k least cost paths for objective (1), the k least delay paths for objectives (2) and (4), and the k least utilised paths for objective (3). Only 5 paths are listed for each objective in Figure 6.3.

6.2.1.3 Adaptation Strategies in VEMOSA

To diversify the search and avoid getting trapped in local optima, two adaptation strategies are adopted to tune the search directions in VEMOSA. One strategy is the competition between similar members in the current population, another one is the two-phase strategy for tuning weight vectors, both strategies have shown to help improving the performance of the EMOSA algorithm in (Li and Landa-Silva, 2008) for the TSP problem.

- The first strategy concerns the competition between the closest solutions in the current population. After a neighbouring solution T' of T_i is generated, the closest solution T_j measured by the Euclidean distance between two weight vectors in the current population is selected. VEMOSA will replace T_j by T' if T_j is worse than T' by comparing their weighted scalarising function values, i.e. $g(T', \lambda^j) < g(T_j, \lambda^j)$. The competition is an intensification strategy which improves the quality of solutions in the current population.
- The second strategy adaptively tunes weight vectors according to the closest non-dominated neighbouring solution in the current population after the current temperature is below a given threshold. This is similar to that is used in EMOSA (Li and Landa-Silva, 2008), aiming to tune the directions of the search when it is getting closer to the Pareto front at the later stage of the evolution. The second strategy is designed to diversify the search directions along the Pareto front.

6.2.2 Experiments and Results

In order to visually observe the impact of neighbourhood structures and adaptation strategies within the EMOSA and VEMOSA algorithms, firstly, the proposed algorithms are evaluated upon two random networks with two objectives. Secondly, based on the observations obtained, the algorithms are compared with the existing approaches in the literature on two multi-objective multicast routing benchmark problems, namely the NSF network in Figure 3.2 and the NTT network in Figure 3.3. Finally, the effectiveness and efficiency of the proposed algorithms are compared upon a set of random networks of different sizes with respect to four objectives. The random networks in the simulation of this thesis are generated by the multicast routing simulator MRSIM. To generate random network topologies, the parameters are set as α = 0.25 and β = 0.40 in MRSIM. In the random networks, the average node degree is set as 4. For each link, the link capacity z_{ij} is set as 1.5Mbps, the delay d_{ij} is generated as the propagation delay depending on the length of the link, the cost c_{ij} is assigned a random value between (0,100] which represents the bandwidth consumption of the link, and the current traffic t_{ij} is randomly loaded with around 50% of its total link capacity. To encourage scientific comparisons, the detailed information of all the problem instances and the experimental results has been provided at http://www.cs.nott.ac.uk~rxq/MRPresource.html.

The parameter settings for all the algorithms are given in Table 6.1. They were determined after a set of extensive tests. It is well known that simulated annealing with a higher starting temperature may lead to better results; however, it normally consumes much longer computational time. The purpose here is to find the balance between the quality of solutions and the running time. For fair comparisons, the same parameter settings are kept for both the VEMOSA and EMOSA algorithms on all the instances tested in this chapter. All simulations were run on a Windows XP computer with PIV 3.4GHZ, 1G RAM.

Table 6. 1 Parameter settings in EMOSA and VMOSA algorithms

Algorithm parameters	Settings
population size	50
starting, final temperature and temperature decrement	100, 5 and 5
number of iterations between two consecutive temperatures	25
temperature threshold for tuning weight vectors	50
constant μ in the adaptation strategy for tuning the search direction	1.05

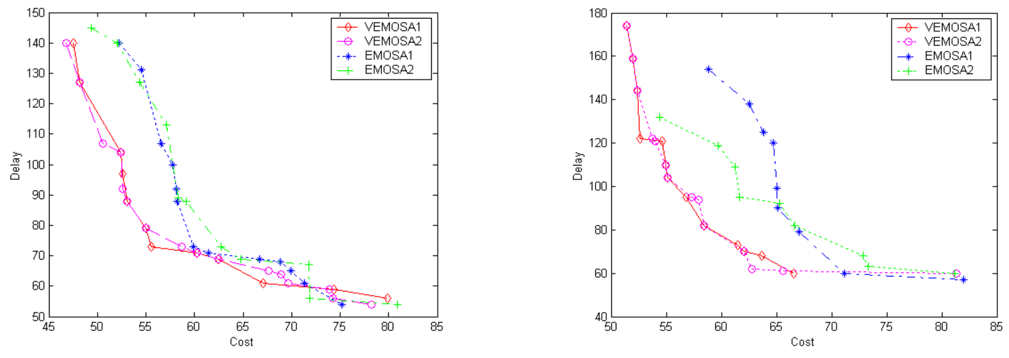
6.2.2.1 Variable Neighbourhoods and Adaptation Strategies in VEMOSA

In this set of experiments, VEMOSA and EMOSA are compared to identify the impact of different neighbourhood structures on the performance of the proposed multi-objective algorithms. In addition, the effects of adaptation strategies are tested in variants of these algorithms. To visually compare the Pareto optimal front generated by these algorithms, only two objectives, namely the cost and delay as defined in

Section 6.2, are therefore considered on two random networks of $|V|$ =50 and $|V|$ =100 with different group sizes, i.e. the number of destination nodes $|R|$ = 20%×$|V|$ and $|R|$ = 30%×$|V|$. The computational time of each algorithm for each run is set to 160 seconds. If within the given computational time the temperature drops to the final temperature, the algorithms will reheat the current temperature to the initial temperature and start again. Four variants of the algorithms using the following notations have been implemented:

- VEMOSA1 and VEMOSA2: VEMOSA with and without adaptation
- EMOSA1 and EMOSA2: EMOSA with and without adaptation

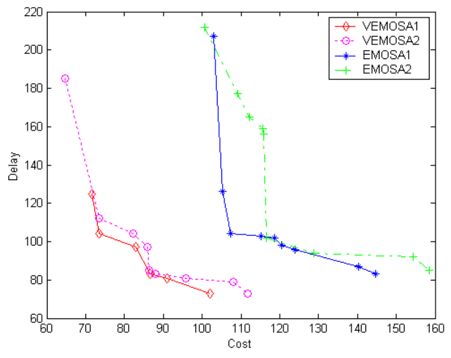
The union of non-dominated solutions found by each algorithm after 20 runs for the random network of size $|V|$ =50 with different group sizes are shown in Figure 6.4. It can be clearly seen that both VEMOSA algorithms with five neighbourhood structures find much better non-dominated solutions in almost all cases than the two EMOSA algorithms with single neighbourhood structure. This is due to that the variable neighbourhood structures in VEMOSA effectively deal with different objectives simultaneously. Within both VEMOSA and EMOSA, the adaptation strategies do not seem to make significant contributions, shown by the similar performance with interweaving Pareto fronts for this small network of 50 nodes.



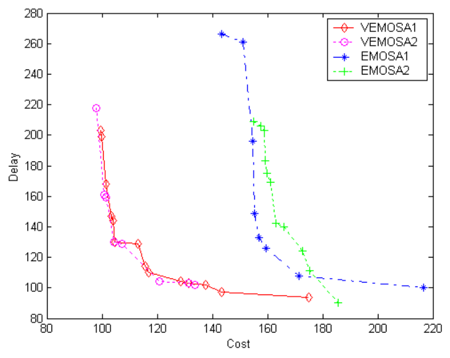
(a) Cost vs. Delay ($|V|$ = 50, $|R|$ = 10) (b) Cost vs. Delay ($|V|$ = 50, $|R|$ = 15)

Figure 6. 4 The non-dominated solutions found by variants of algorithms for the problems of $|V|$ =50 with different group sizes in 160 seconds

Similarly, in Figure 6.5, the union of non-dominated solutions found by 20 runs of the four algorithms shows that VEMOSA performs better than EMOSA with a much larger scale on the larger network of 100 nodes and different group sizes. It becomes clearer that variable neighbourhoods can really contribute to a better performance of VEMOSA on larger networks. It is interesting to see that with single neighbourhood structure, the adaptation strategies in EMOSA1 significantly improve the algorithm performance compared against EMOSA2. However, both VEMOSA algorithms with variable neighbourhood structures have similar performances. This shows that the adaptation strategies have less impact on VEMOSA, where variable neighbourhoods can effectively tune search directions towards diverse Pareto optimal solutions of different objectives. The proposed VEMOSA algorithm outperforms EMOSA even without adaptation strategies which carefully concern the diversities and directions of the search.



(a) Cost vs. Delay ($|V|$ = 100, $|R|$ = 20)



(b) Cost vs. Delay ($|V|$ = 100, $|R|$ = 30)

Figure 6. 5 The non-dominated solutions found by variants of algorithms for problems of $|V|$ =100 with different group sizes in 160 seconds

6.2.2.2 The VEMOSA and EMOSA Algorithms on Benchmark Instances

1) Comparison on the NSF Network

For the NSF benchmark problem in Figure 3.2, an optimal Pareto Front (*PF*) that consists of 16 solutions for the problem with four objectives defined in Section 6.2 have been found by an exhaustive search in (Crichigno and Baran, 2004b). The same four variants of VEMOSA and EMOSA algorithms in Section 6.2.2.1 are tested on the NSF network with the four objectives. Table 6.2 presents the results of each algorithm within 100 runs, compared against those of MOEA (Crichigno and Baran, 2004a) and the improved MOEA (Crichigno and Baran, 2004b) in the literature.

For this small NSF problem, the proposed VEMOSA and EMOSA algorithms perform better than the two MOEA approaches in the literature. Within all algorithms, both VEMOSA algorithms perform the best, i.e. always find all 16 Pareto optimal solutions in all 100 runs. This, together with the above observations in Section 6.2.2.1, supports the conclusion that VEMOSA are more effective than EMOSA and MOEA algorithms with only one neighbourhood structure. It is not surprising to see that the VEMOSA algorithms spend longer computational time to obtain better results compared with the EMOSA algorithms when using the same parameter settings.

Table 6. 2 The maximum, minimum and average number of non-dominated solutions found by different algorithms for the NSF network in Figure 6.2(a) and the computational time of each algorithm (|*NDS*|: the number of non-dominated solutions; % in *PF*: the lowest percentage of non-dominated solutions belonging to *PF*, i.e. Min{|*NDS*|}/|*PF*|, here |*PF*| = 16)

Algorithms	**Max |*NDS*|**	**Min |*NDS*|**	**Average |*NDS*|**	**% in *PF***	**Computational time (sec)**
VEMOSA1	16	16	16	**100%**	12.719
VEMOSA2	16	16	16	**100%**	12.528
EMOSA1	16	15	15.98	93.8%	5.346
EMOSA2	16	14	15	87.5%	5.426
MOEA (Crichigno and Baran, 2004a)	16	10	12.72	62.5%	/
MOEA (Crichigno and Baran, 2004b)	16	12	13.54	75%	/

2) Comparison on the NTT network

The proposed four algorithms are tested on the NTT network in Figure 3.3 with different multicast groups, shown in Table 6.3. Similar to (Diego and Baran, 2005), the traffic t_{ij} for each link in the NTT network is randomly loaded with around 50% of its total link capacity in the experiments. However, due to the lack of the same experiment data as that in (Diego and Baran, 2005), this thesis only reports and analyses the experimental results of the proposed four algorithms in this work.

Table 6. 3 Two different multicast groups within the NTT network in Figure 3.3

Group	**Source**	**Destination Set *R***	|*R*|
group1 (small)	5	{0,1,8,10,22,32,38,43,53}	9
group2 (large)	4	{0,1,3,5,6,9,10,11,12,17,19,21, 22,23,25,33,34,37,41,44,46,47,52,54}	24

To obtain an approximation of the Pareto optimal solution set, denoted by Y_{PF}, the following six-step procedure in (Diego and Baran, 2005) is also used here to generate Y_{PF}:

1) Each algorithm is run 10 times.
2) For each algorithm, 10 sets of non-dominated solutions Y_1, Y_2, …, Y_{10} are obtained, one from each run.
3) For each algorithm, the aggregate population Y_T is obtained, where $Y_T = \bigcup_{i=1}^{10} Y_i$.

4) Dominated solutions are deleted from Y_T to obtain the non-dominated solution set for each algorithm, denoted by Y_{alg}. In the present work, *alg* represents an algorithm, i.e. $Y_{VEMOSA1}$, $Y_{VEMOSA2}$, Y_{EMOSA1}, Y_{EMOSA2}.

5) A set of solutions Y' from all algorithms is obtained, i.e. $Y' = \bigcup_{i=1}^{4} Y_{\mathrm{alg}\,i}$

6) Non-dominated solutions in Y' are selected as an approximation of the true Pareto optimal solution set, named Y_{PF}.

Table 6. 4 The total number of non-dominated solutions for each multicast group of the NTT network with two different multicast groups shown in Table 6.3

Computational Time	160 seconds		320 seconds	
Multicast Group	group1	group2	group1	group2
$\mid Y_{PF} \mid$	6	9	6	11

Table 6.4 presents the total number non-dominated solutions in Y_{PF} obtained by the four variants of EMOSA and VMOSA algorithms for the NTT network by using the above six-step procedure. Results are obtained for the NTT network with two different multicast groups by using 160 seconds and 320 seconds, respectively. The proposed four algorithms are compared with regard to the obtained Y_{PF} on the NTT network. Results are presented in Table 6.5 and Table 6.6 for NTT with small and large multicast group, respectively.

Table 6. 5 Four variants of algorithms for solving the NTT network with small multicast group 1 (The best results are in bold)

Algorithms	**Time = 160 seconds**				**Time = 320 seconds**			
	Average Solutions in Y_{PF}	**Average Dominated Solutions**	**Average Total Solutions**	**% Solutions in Y_{PF}**	**Average Solutions in Y_{PF}**	**Average Dominated Solutions**	**Average Total Solutions**	**% Solutions in Y_{PF}**
VEMOSA1	**3.8**	8.2	12	**63.3%**	**4**	6	10	**66.7%**
VEMOSA2	**3.8**	8.6	12.4	**63.3%**	**4**	5	9	**66.7%**
EMOSA1	2.2	12.8	15	36.7%	2.8	13.2	16	46.7%
EMOSA2	3.4	10	14.4	56.7%	3	9	12	50%

It can be seen in Table 6.5 that the VEMOSA algorithms find more non-dominated solutions in Y_{PF} than that of EMOSA for each run. It is interesting to see that EMOSA2 outperforms EMOSA1, meaning that the adaptation strategies of tuning search directions does not help to improve the algorithm performance for the small multicast group instance by diversifying the search directions. VEMOSA1 and VEMOSA2 have similar performance, again showing that the adaptation of search directions does not make significant contribution to the VEMOSA algorithms.

Table 6. 6 Four variants of algorithms for solving the NTT network in Figure 6.4 with large multicast group 2 (The best results are in bold)

Algorithms	**Time = 160 seconds**				**Time = 320 seconds**			
	Average Solutions in Y_{PF}	**Average Dominated Solutions**	**Average Total Solutions**	**% Solutions in Y_{PF}**	**Average Solutions in Y_{PF}**	**Average Dominated Solutions**	**Average Total Solutions**	**% Solutions in Y_{PF}**
VEMOSA1	**2**	8	10	**22.2%**	1	10.2	11.2	9.1%
VEMOSA2	1.8	8.2	10	20%	**1.8**	7.4	9.2	**16.4%**
EMOSA1	0.6	15.4	16	6.7%	1	21.8	22.8	9.1%
EMOSA2	0	27.8	27.8	0	0	17	17	0

Results in Table 6.6 show that for the NTT network with large multicast group, VEMOSA algorithms outperform EMOSA algorithms, demonstrating that variable neighbourhoods help the algorithms to find more non-dominated solutions. EMOSA1 with the adaptation strategy obtains more solutions in Y_{PF} than that of EMOSA2, which does not find a single solution in Y_{PF}. Compared with the results in Table 6.5, this indicates that for the large multicast group instance, the adaptation strategies improve the algorithm performance. However, if given longer computational time, even without adaptation strategies, VEMOSA2 finds more non-dominated solutions than that of VEMOSA1, showing that the adaptation strategies have less impact to the performance of VEMOSA algorithms.

3) Comparison on Random Networks

Finally, the four algorithms are tested on random networks of network size $|V| = 50$ and $|V| = 100$ with different group sizes of $|R| = 20\%\times|V|$ and $|R| = 30\%\times|V|$ and concerning all four objectives defined in section 6.2. Table 6.7 presents the total number of non-dominated solutions obtained by the four algorithms on the random networks by using the six-step procedure defined above.

Table 6. 7 The total number non-dominated solutions of Y_{PF} on the random networks of four objectives

Network Size	$\|V\| = 50$		$\|V\| = 100$	
Group Size	$\|R\| = 10$	$\|R\| = 15$	$\|R\| = 20$	$\|R\| = 30$
$\| Y_{PF} \|$	96	164	46	46

Table 6.8 and Table 6.9 present the results of the four variant algorithms on the two random networks with different group sizes in the given 360 seconds. Average results from 10 runs for each problem show that both VEMOSA algorithms significantly outperform the EMOSA algorithms in finding more non-dominated solutions in Y_{PF} obtained in Table 6.7. The scope of differences again is much larger for larger problems, demonstrating the effectiveness of the VEMOSA algorithms for larger instances. It can be

seen that the two VEMOSA algorithms have similar performances with regard to the average number of non-dominated solutions found, i.e. % solutions in Y_{PF} for problems of different group sizes. The adaptation strategies have small impact on the performance of both VEMOSA and EMOSA algorithms.

Table 6. 8 Comparison of different algorithms on random network of $|V| = 50$ with different group sizes (Computational time = 360 seconds, the best results are in bold)

Algorithms	**$\|R\| = 10$**				**$\|R\| = 15$**			
	Average Solutions in Y_{PF}	**Average Dominated Solutions**	**Average Total Solutions**	**% Solutions in Y_{PF}**	**Average Solutions in Y_{PF}**	**Average Dominated Solutions**	**Average Total Solutions**	**% Solutions in Y_{PF}**
VEMOSA1	17.4	19.8	37.2	18.1%	**28**	94.2	122.2	**17.1%**
VEMOSA2	**17.8**	21.8	39.6	**18.5%**	23	92.2	115.2	14%
EMOSA1	8.4	97.8	106.2	8.8%	0.8	96.6	97.4	0.5%
EMOSA2	4	79.6	83.6	4.2%	8.6	37.8	46.4	5.2%

Table 6. 9 Comparison of different algorithms on random network of $|V| = 100$ with different group sizes (Computational time = 360 seconds, the best results are in bold)

Algorithms	**$\|R\| = 20$**				**$\|R\| = 30$**			
	Average Solutions in Y_{PF}	**Average Dominated Solutions**	**Average Total Solutions**	**% Solutions in Y_{PF}**	**Average Solutions in Y_{PF}**	**Average Dominated Solutions**	**Average Total Solutions**	**% Solutions in Y_{PF}**
VEMOSA1	**4.8**	26.4	31.2	**10.4%**	3.4	15.6	17	7.4%
VEMOSA2	4.6	26.4	31	10%	**6.8**	11	34	**14.8%**
EMOSA1	0.4	40.2	40.6	0.9%	0	19	19	0
EMOSA2	0	51.4	51.4	0	0	27	27	0

To summarize, the extensive experimental results on a range of different multi-objective multicast routing instances with different features demonstrate the efficiency and effectiveness of the proposed evolutionary simulated annealing algorithms based on variable neighbourhoods. Variable neighbourhood structures which are specifically designed for different objectives significantly improve the performance of the proposed algorithms, especially for problems of larger network size. The VEMOSA algorithms work very well even without the adaptation strategies, which have shown to be important in the EMOSA algorithms for both the multi-objective multicast routing problem in the experiments and the TSP instances in (Li and Landa-Silva, 2008).

The research outcomes of these variants of VEMOSA algorithms have been published in the special issue on Heuristic Optimisation in the Journal of the Operational Research Society (Xu and Qu, 2011).

6.3 A Simulated Annealing based Genetic Local Search Algorithm

As reviewed in Chapter 2, in recent multi-objective optimisation research, some SA algorithms (Czyzak and Jaszkiewicz, 1998; Li and Landa-Silva, 2008) and genetic local search (GLS) algorithms (Ishibuchi and Murata, 1998; Jaszkiewicz, 2002) have been investigated for different multi-objective optimisation problems. These algorithms have shown to be very effective when exploring the search space of complex multi-objective optimisation problems, especially when equipped with mechanisms designed with regard to problem features.

To the best of my knowledge, there is no investigation on applying GLS algorithms to multi-objective multicast routing problems. The only two recent relevant algorithmsare applied to single objective multicast routing problems in the literature. In (Zahrani *et al.*, 2008), a GLS utilises a logarithmic simulated annealing in a pre-processing step to analyse the landscape of a single objective multicast routing problem subject to multiple constraints in a group multicast scenario. Another hybrid genetic simulated annealing algorithm is proposed in (Zhang *et al.*, 2009) for the least-cost multicast routing problem with the bandwidth, delay and delay jitter constraints. The hybrid algorithm combines the strengths of both SA and GA. On the one hand, SA has the character of escaping from local optimum by intelligently accepting worse solutions so that it can avoid the premature convergence of GA. On the other hand, the population based GA can help the algorithm to converge fast. Their simulation results demonstrate the efficiency of the hybrid algorithm. In the previous section, the VEMOSA algorithm, an evolutionary multi-objective simulated annealing algorithm with variable neighbourhoods, is proposed to solve the multi-objective multicast routing problem. The strategies based on SA have shown to be effective in driving a population of solutions towards the Pareto front of the problem with four objectives. However, together with the variable neighbourhood structures specially designed for the multiple objectives, the SA strategies have less impact on the performance of the algorithm.

In this work, motivated by the efficiency of both the SA strategies and GLS, the first multi-objective simulated annealing based genetic local search (MOSAGLS) algorithm is developed to solve the multi-objective multicast routing problem. MOSAGLS evolves by using SA-based strategies within the genetic evolutionary process to generate non-dominated solutions while simultaneously minimising five objectives, including (1) the cost, (2) the maximal end-to-end delay, (3) the maximal link utilisation, (4) the average delay, and (5) the delay variation of the multicast tree, see the definition of these five objectives in Section 3.5.2. A new SA-based adaptive mutation probability is used to improve the efficiency of the evolutionary process. The similar SA-based adaptation strategies of competition between the closest members and adaptively tuning search directions in (Li and Landa-Silva, 2008) have been adopted in MOSAGLS. The impact of the SA strategies and the local search heuristic in the genetic evolution has also been investigated within this new hybrid algorithm.

6.3.1 Algorithm Description

In MOSAGLS, the initial population consists of randomly generated multicast trees. During the evolution process, parent solutions are chosen to produce child trees by using the defined crossover and mutation operators. A local search heuristic is then applied to the generated child tree to produce a new improved tree. An external non-dominated solution set *NDS* is maintained to record the non-dominated solutions obtained during the evolution. MOSAGLS stops after a certain computational time, or the temperature of the SA drops to the final temperature. The *NDS* is output as the final results at the end of the algorithm. The pseudo-code of the hybrid MOSAGLS is presented in Figure 6.6.

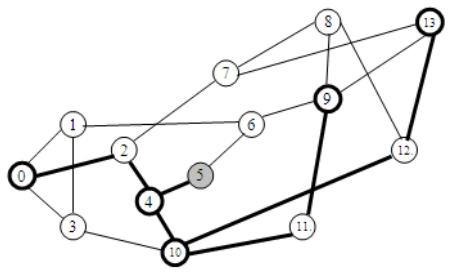
The SA-based strategies in the proposed MOSAGLS are used to adaptively set the mutation rate, to guide and tune the search directions and to make decision on solution acceptance. The temperature of SA is controlled by the cooling schedule throughout generations. After the crossover operation, the mutation is carried out based on an adaptive probability according to both the current temperature and the fitness of the offspring and current population. Each solution in the population is associated with a random weight vector. This vector, together with the temperature, takes part in the solution acceptance and the tuning of search directions. The newly generated trees replace the selected parent based on a probability calculated using the weight vector and the current temperature. When the temperature is decreased to below a threshold, the weight vector is modified to tune the search directions.

```
MOSAGLS ($(G = (V, E), s, R, m, t_{max}, t_{min}, t_c, t_{step})$)
{ // $G$: network topology; $s$: source node; $R$: destination set; $m$: number of objectives;
  // $t_{max}/t_{min}$: starting/final temperatures; $t_c$: temperature threshold for tuning the weight vector;
  // $t_{step}$: temperature decrement;
  Generate the random initial population Pop with $T_1, \ldots, T_{|Pop|}$ multicast trees;
  For $T_1, \ldots, T_{|Pop|}$, produce $|Pop|$ distinct random weight vectors $\lambda^1,\ldots,\lambda^{|Pop|}$, each weight vector with uniform spread;
  Form non-dominated solution set NDS from Pop;
  Calculate the fitness by using formula (6.5) for all the solutions in NDS and Pop;
  do {
      Current temperature $t = t_{max}$;
      while $t \geq t_{min}$ do {
          Randomly select a pair of parent solutions $T_i$ and $T_j$ from Pop;
          Crossover operation over parents $T_i$ and $T_j$;
          Mutation with adaptive probability $p_m$ to the solution from crossover; // SA strategy
          Generate $T'$ by applying the local search to the solution after mutation; // See Section 6.3.1.3
          if $T'$ is not equal to parents $T_i$ or $T_j$ then
              if $T'$ is not in the Pop then
                  Select a parent $T_c$ from $T_i$ or $T_j$ with higher probability by comparing $P(T_i, T', \lambda^i, t)$ and $P(T_j, T', \lambda^j, t)$;
                  Replace the selected parent $T_c$ by $T'$ with the probability $P(T_c, T', \lambda^c, t)$; // SA strategy
          //compete with the closest member in Pop, SA strategy
          Find the closest solution $T_k \in Pop$ of $T'$;
          if $(g^{(ws)}(T', \lambda^k) < g^{(ws)}(T_k, \lambda^k))$ then
              $T_k = T'$ // replace the closest member $T_k$ in Pop;
          Update NDS;
          Re-calculate the fitness for all solutions in NDS and Pop;
          $t = t - t_{step}$ // temperature decrement
          //Adaptively tune the weight vectors, SA strategy
          if $(t < t_c)$ then TuningSearchDirections($t$) // See Figure 6.11
      } end of while loop
  } while stopping condition is not met
  return NDS;
}
```

Figure 6. 6 The pseudo-code of the MOSAGLS algorithm

6.3.1.1 The Representation of the Multicast Tree

In the proposed MOSAGLS algorithm, the same encoding method in Zhang *et al.* (2009) is used, where each chromosome adopts the multicast tree structure. A multicast tree is represented by an ordered set of $|R|$ paths from the source node s to each destination $r_d \in R$, $|R|$ is the group size. That is, each chromosome contains $|R|$ genes $\{g_1, g_2, \ldots, g_{|R|}\}$, where each g_d represents a path between the source node s and the d-th destination node r_d, $d = 1, \ldots |R|$. This representation can be easily adapted in genetic operations and avoid illegal chromosome in the initialization. Given the benchmark NSF network in Figure 3.2, an example multicast tree and its representation in MOSAGLS are shown in Figure 6.7.



The Solution Encoding	
Gene	Path
g_1	5-4-2-0
g_2	5-4
g_3	5-4-10-11-9
g_4	5-4-10
g_5	5-4-10-12-13

(a) A multicast tree, $s = 5$, $R=\{0, 4, 9,10,13\}$ (b) The solution representation

Figure 6. 7 An example multicast tree and its representation on the NSF network

6.3.1.2 The Genetic Algorithm in MOSAGLS

In the evolutionary process of the proposed MOSAGLS, the initial population consists of a fixed number of multicast trees. They are generated by starting from the source node and randomly selecting the next connected node until all destination nodes have been mounted on the tree. In each generation, crossover and mutation operations are carried out on two randomly selected parents from the current population. A local search is used to further explore better neighbouring solutions of the generated tree. A non-dominated set (*NDS*) of selected multicast trees are stored during the evolution. It is maintained by storing the newly generated tree if it is not dominated by any tree in the current *NDS*, and removing the trees which are dominated by the generated tree.

1) The Strength Pareto based Evaluation

The SPEA (Strength Pareto Evaluation Algorithm) in (Zitzler and Thiele, 1999) is adopted in the genetic algorithm in MOSAGLS to calculate the fitness of individuals. It is used to maintain and update *NDS* as well as used in the selection, crossover and mutation operations. The value of the five objectives (1)-(5) mentioned in Section 6.3 is calculated for each individual. Based on the values of the five objectives, the fitness of each individual is evaluated by using the SPEA scheme as follows:

- For each non-dominated solution $T_i \in NDS$, a strength $q_i \in [0,1]$ is calculated. It is the proportion of the number of T_j to the number of population, where T_i dominates or is equal to T_j in the population *Pop*, i.e. denoted by $T_i \triangleright T_j$:

$$q_i = \left| T_j \mid T_j \in Pop \wedge T_i \triangleright T_j \right| \cdot |Pop|^{-1} \tag{6.3}$$

- For each individual T_j in the population, the strength $q_j \in [1, 1+|NDS|]$ is computed by summing the strength of all non-dominated solutions $T_i \in NDS$, where $T_i \rhd T_j$, plus one:

$$q_j = 1 + \sum_{T_i \in NDS, T_i \rhd T_j} q_i \tag{6.4}$$

- Finally, the fitness of each individual T_j in the population is calculated as the inverse of its strength q_j:

$$F(T_j) = (1 + \sum_{T_i \in NDS, T_i \rhd T_j} q_i)^{-1} \tag{6.5}$$

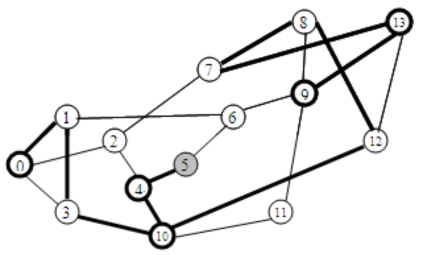
The strength Pareto based evaluation in the proposed algorithm is similar to that is used in MOEA algorithms in (Crichigno and Baran, 2004a; 2004b). This work focuses on the investigation of genetic local search with simulated annealing strategies, so the simple and effective SPEA method in the literature is chosen. In addition, for fair comparisons, the same SPEA method is applied to evaluate the performance of the proposed hybrid algorithm against the MOEA algorithms in the experiments in the late section.

2) The Selection Method

A binary tournament selection method is employed, which is also used in the MOEA algorithm in (Crichigno and Baran, 2004a), to select parents. Each time two individuals from the population are randomly selected. The individual with a higher fitness value defined by the strength Pareto based evaluation in Eq. (6.5) wins the tournament and is selected as a parent. Two parents are chosen by applying the tournament selection twice.

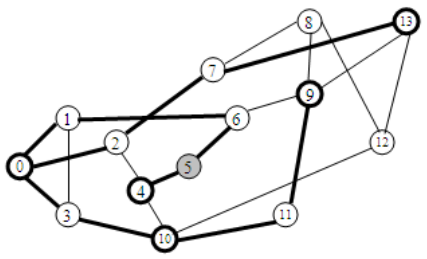
3) The Crossover Operation

A two-point crossover operator, with a crossover rate of 1, is applied to each selected pair of parents. Based on the representation of an ordered set of paths, the paths between two randomly generated points in one parent are selected and replaced by the corresponding paths in another parent. Note that some selected paths may share the same links with some remaining paths. Such links will not be removed to ensure the tree is connected. To avoid loops in generating the new multicast tree, the selected path will be replaced by mounting the new path from the destination node until it connects to an on-tree node.



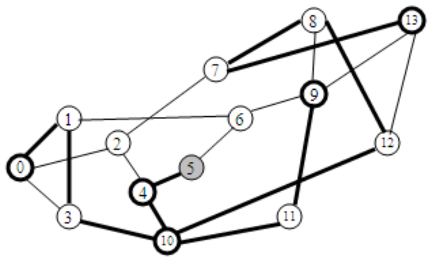
Parent 1	
Gene	Path
g_1	5-4-10-3-1-0
g_2	5-4
g_3	**5-4-10-12-8-7-13-9**
g_4	5-4-10
g_5	5-4-10-12-8-7-13

(a) Parent 1: $C(T)$ = 11.2, $DM(T)$ = 60, $\alpha(T)$ = 0.67, $DA(T)$ = 38.4, $DV(T)$ = 53



Parent 2	
Gene	Path
g_1	5-6-1-0
g_2	5-4
g_3	**5-6-1-0-3-10-11-9**
g_4	5-6-1-0-3-10
g_5	5-6-1-0-2-7-13

(b) Parent 2: $C(T)$ = 9.2, $DM(T)$ = 75, $\alpha(T)$ = 0.73, $DA(T)$ = 49, $DV(T)$ = 68



Individual	
Gene	Path
g_1	5-4-10-3-1-0
g_2	5-4
g_3	**5-4-10-11-9**
g_4	5-4-10
g_5	5-4-10-12-8-7-13

(c) New offspring tree after crossover: $C(T)$ = 11.4, $DM(T)$ = 55, $\alpha(T)$ = 0.6, $DA(T)$ = 33.4, $DV(T)$ = 48

Figure 6. 8 An example of the two-point crossover operation

An example of the crossover operation is shown in Figure 6.8, where path g_3 = (5-4-10-12-8-7-13-9) between the selected two crossover points in parent 1 shown in Figure 6.8(a) is replaced by the new path (5-6-1-0-3-10-11-9) of g_3 in parent 2 shown in Figure 6.8(b). To avoid loops in the generated multicast tree, the new path (5-6-1-0-3-10-11-9) is mounted to the tree by starting from the destination node 9, and adding only the path 9-11-10 until it connects to the on-tree node 10. A new tree is then generated as shown in Figure 6.8(c).

The simple representation of multicast trees facilitates an easy implementation of crossover operations. By mounting the selected path(s) in parent 2 from the destination node, the newly generated offspring is guaranteed to be feasible. Note that while replacing g_3 in parent 1, the links along the path 5-4-10-12-8-7-13 still remain in the multicast tree as they also appear in g_5 in the original tree of parent 1.

In multicast trees, some paths share common links, especially those near the root of the tree, i.e. links near the source node appear multiple times in the solution representation. This is due to the nature of the multicast tree that all paths should pass a subset of the *d* links from the source node. The multiple appearances of some (partial) paths in the tree are adaptively selected through the evolution process of MOSAGLS if they present to be good attributes in the individuals.

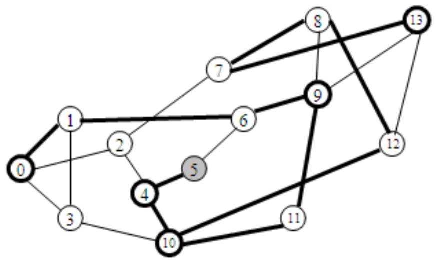
4) The Mutation Operation

For a selected individual, the mutation operation randomly replaces a path by using an alternative path stored in a routing table, which is the same as that was devised in (Crichigno and Baran, 2004a). The routing table for the destination $r_d \in R$ consists of *k* least cost (for objective (1)), *k* least delay (for objective (2) (4) (5)) and *k* least used paths (for objective (3)) by using the *k*-shortest path algorithm (Eppstein, 1998). As objectives (2), (4) and (5) (see Section 6.3) are all related to the delay, they share the same *k* least delay paths in the routing table. A new randomly selected path in the routing table then replaces the original path from the source to the destination r_d for gene g_d.

An illustration of the mutation operation is shown in Figure 6.9 for the offspring generated in Figure 6.8(c). Figure 6.9(a) shows the routing table of destination r_1 = 0, listing 5 paths for each objective, i.e. *k* = 5 in this example. If a random path (5-4-10-11-9-6-1-0) is selected from the routing table, a new solution is generated in Figure 6.9(b) by replacing the original path (5-4-10-3-1-0) of g_1 in Figure 6.8(c).

ID	Least cost paths	ID	Least delay paths	ID	Least used path
1	5-6-1-0	1	5-4-2-0	1	5-4-10-3-0
2	5-4-2-0	2	5-6-1-0	2	5-6-1-0
3	5-6-9-11-10-3-0	3	5-4-10-3-0	3	5-4-2-0
4	5-6-1-3-0	4	5-4-2-7-8-9-6-1-0	4	**5-4-10-11-9-6-1-0**
5	5-4-10-3-0	5	5-6-9-8-7-2-0	5	5-4-10-3-1-0

(a) Routing table of destination node r_1 = 0 for the multicast tree in Figure 6.8(c)



Solution	
Gene	Path
g_1	**5-4-10-11-9-6-1-0**
g_2	5-4
g_3	5-4-10-11-9
g_4	5-4-10
g_5	5-4-10-12-8-7-13

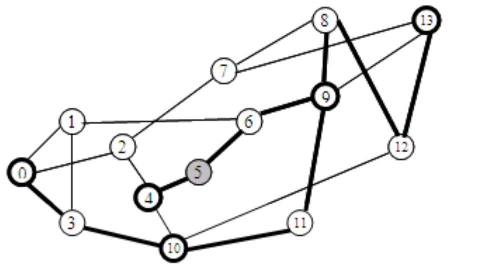
(b) Multicast tree after mutation: $C(T)$ = 11.4, $DM(T)$ = 71, $\alpha(T)$ = 0.6, $DA(T)$ = 36.6, $DV(T)$ = 64

Figure 6. 9 An example of the mutation operation

6.3.1.3 The Local Search in MOSAGLS

Instead of reproducing the offspring directly to the next generation, a local search is applied to further enhance the offspring, simulating the maturing phenomenon in the nature. The local search is based on the Prim's minimum spanning tree algorithm (Betsekas and Gallager, 1992) which finds a tree with the minimal total weights of the links spanning a subset of nodes in the graph. To apply the local search, each offspring solution (a multicast tree) is firstly represented by a binary array of $|V| = n$ bits, each representing to a node in the network $G = (V, E)$. Each bit is assigned 1 if the corresponding node is in the multicast tree; 0 otherwise. This representation has been widely used in the literature and shows to be effective for local search and genetic operations for multicast routing problems (Skorin-Kapov and Kos, 2003; 2006). The same representation is used in the previous work (Xu and Qu, 2009) as well.

In MOSAGLS, the local search operates on the initial tree, i.e. the offspring multicast tree generated by the mutation operation, and repeatedly flips a bit in the initial tree to create a new spanning tree until a new better tree is found or a maximum number of nodes have been flipped. According to the definition of Pareto dominance, a new better tree means it dominates the initial tree by comparing the value of each objective one by one. After the local search is applied to the above solution in Figure 6.9(b), a new solution is shown in Figure 6.10.



$C(T) = 8.8$, $DM(T) = 53$, $\alpha(T) = 0.6$, $DA(T) = 27$, $DV(T) = 46$

Solution	
Gene	Path
g_1	5-6-9-11-10-3-0
g_2	5-4
g_3	5-6-9
g_4	5-6-9-11-10
g_5	5-6-9-8-12-13

Figure 6. 10 An example solution after the local search on the solution in Figure 6.9(b)

6.3.1.4 Simulated Annealing Strategies in MOSAGLS

In MOSAGLS, SA strategies have been adopted to 1) set the adaptive mutation rate, 2) to make decision of the acceptance of the new offspring in the genetic local search, and 3) to tune the search directions at the later stage of the evolution.

1) SA based Adaptive Mutation Probability

In the proposed MOSAGLS algorithm, mutation is carried out with an adaptive probability by using both the evolutionary information and SA strategy. The adaptive probability p_m is based on not only the current temperature but also the difference between the fitness of the new offspring and the average fitness of the current population, as shown in formula (6.6). The mutation operation is always applied to an offspring if it is better than the average of the current population. For a worse new offspring, the mutation probability p_m is calculated as follows:

$$p_m = \begin{cases} 1 & F_{new} > F_{avg} \\ e^{-(F_{avg} - F_{new})/t} & F_{new} \le F_{avg} \end{cases} \tag{6.6}$$

where the average fitness of the current population F_{avg} and the fitness of the new offspring F_{new} are calculated by using the SPEA scheme defined in Eq. (6.5); t: the current temperature.

On the one hand, better offspring near the Pareto front will be given higher chance to be evolved by the mutation operation, making the evolution more effective by investing computational time on more promising offspring. On the other hand, the mutation rate is higher if the current temperature t is higher. This means at the early stage of the evolution, worse offspring still has the chance to be mutated, giving the whole population the chance to explore more areas of the search space. At the later stage when the mutation rate decreases to a small value along with the temperature decrement, worse solutions are rarely evolved to encourage the convergence of the algorithm.

2) Simulated Annealing based Acceptance Probability

Based on the physical process of SA, the SA metaheuristic accepts a non-improved state/solution with a probability depending on the temperature in the cooling schedule at that time. As a result, it is able to escape from local optima by intelligently visiting worse solutions and thus effectively explore the search space of complex multi-objective optimisation problems. To assist the decision making of accepting a new generated offspring, the weighted scalarising function in Eq. (6.1) of a convex combination of different objectives is used in the SA strategy. As defined in Eq. (6.1), the weighted scalarising function of solution *x* is: $g^{(ws)}(x, \lambda) = \sum_{i=1}^{m} \lambda_i f_i(x)$, where λ is a weight vector, $\lambda_i \in [0,1]$, $i = 1, \ldots, m$, m is the number of objectives and $\sum_{i=1}^{m} \lambda_i = 1$; each λ_i is associated with one objective function $f_i(x)$.

In MOSAGLS, the probability of a new solution x' to replace a solution x is calculated based on the formula defined in Eq. (6.2): if $\Delta g(x, x', \lambda) < 0$, $P(x, x', \lambda, t) = 1$; otherwise, $P(x, x', \lambda, t) = e^{-\Delta g(x,x',\lambda)/t}$, where $\Delta g(x, x', \lambda) = g^{(ws)}(x', \lambda) - g^{(ws)}(x, \lambda)$; $t > 0$ is the current temperature.

In the initialization of the MOSAGLS algorithm, a normalised weight vector λ^i is randomly generated for each solution T_i. At each generation, a new solution T' is generated after the local search upon the offspring produced from the chosen pair of parent solutions T_i and T_j by the GA operations. If T' is different from T_i and T_j and is not in current population *Pop*, then one parent T_c of lower quality is replaced by the newly generated tree T' with the probability calculated by $P(T_c, T', \lambda^c, t)$, where T_c is the parent with a higher probability by comparing $P(T_i, T', \lambda^i, t)$ and $P(T_j, T', \lambda^j, t)$.

3) Simulated Annealing based Competition and Search Direction Tuning

Two SA based adaption strategies, i.e. the competition between similar members and the two-phase search directions tuning, have shown to be effective for solving the TSP instances (Li and Landa-Silva, 2008). They are designed to diversify the search directions at the later stage and avoid solutions being trapped in local optima during the evolutionary process. In the proposed MOSAGLS, two similar adaptation strategies have been adopted as follows:

- The first strategy is defined concerning the competition between close solutions in the current population. Euclidean distance, which is widely used in the multi-objective literature, has been used in MOSAGLS to measure the distance of two solutions in the objective space. After a new solution T' is generated, the closest solution T_k measured by the Euclidean distance in the current population is chosen as a neighbour of T'. MOSAGLS then replaces T_k with T' if T_k is worse than T' by comparing their weighted scalarising function values, i.e. if $g^{(ws)}(T', \lambda^k) < g^{(ws)}(T_k, \lambda^k)$.
- The second strategy is to adaptively tune the weight vectors according to the closest non-dominated neighbouring solution in the current population when the current temperature is below a threshold. It aims

to tune the search direction when the search is getting closer to the Pareto front at the later stage of the evolution. The pseudo-code of the search direction tuning is presented in Figure 6.11.

```
TuningSearchDirections ($t$)
{ // $t$: current temperature; $\mu$: the constant for tuning the search direction
    foreach $T_i \in P$ do {
        Find the closest non-dominated solution $T_i'$ of $T_i$ in the current population by
        using the Euclidean distance between two solutions;
        foreach $obj \in \{1, \ldots, m\}$ do // $m$: number of objectives
        {
            if $f_{obj}(T_i') < f_{obj}(T_i)$ then $\lambda^i_{obj} = \mu\lambda^i_{obj}$;
            else $\lambda^i_{obj} = \lambda^i_{obj} / \mu$;
        } end of foreach
    Normalise $\lambda^i$ by setting $\lambda^i_{obj} = \lambda^i_{obj} / \sum_{p=1}^{m} \lambda^i_p$ }
}
```

Figure 6. 11 The pseudo-code of search directions tuning based on the SA strategy

The two adaptive strategies aim to avoid solutions being trapped in local optima and adaptively tune the search directions towards diverse solutions in the Pareto front when the temperature is below a threshold. Both strategies have also been investigated in the SA based multi-objective algorithm in (Xu and Qu, 2010), but shown to be less effective when variable neighbourhood structures are used in the algorithm for the multi-objective multicast routing problem. In the present work, they are hybridised with the new genetic local search for solving the problem.

6.3.2 Experiments and Results

A large number of simulations have been carried out to test the propsoed algorithms on both the benchmark and random networks with different objectives. Two variants of algorithms have been implemented and evaluated in the following subsections:

1) MOSAGA, the proposed algorithm without local search, and
2) MOSAGLS, the proposed algorithm with local search

Firstly, different components of the proposed algorithms on a set of random networks with two objectives (1) and (2), i.e. the cost and delay, are evaluated. Based on the observations obtained, the proposed algorithms are then compared with two evolutionary algorithms MOEA1 (Crichigno and Baran, 2004a) and MOEA2 (Crichigno and Baran, 2004b) in the literature on two benchmark networks with different number of objectives. Finally, the experimental results demonstrate the effectiveness and efficiency of the proposed algorithms upon a set of random networks with all the five objectives given in Section 6.3. The random networks are generated in the same manner as described in Section 6.2.2. All simulations were run on a Windows XP computer with PIV 3.4GHZ and 1G RAM by using the multicast routing simulator MRSIM.

6.3.2.1 Parameter Settings

All parameter settings were determined after a set of tests to find the balance between the quality of solutions and the running time. For example, to obtain a proper range of values for the population size and the starting temperature, a set of initial tests were carried on the NSF network shown in Figure 3.2 to compare the performance of MOSAGA with different parameter settings. The optimal Pareto Front (*PF*) of NSF benchmark problem that composes of 16 solutions has been found by an exhaustive search method in (Crichigno and Baran, 2004b), where four objectives (1), (2), (3) and (4) descried in section 6.3 have been considered in the algorithm. In this group of tests, the same four objectives are considered in MOSAGA. Table 6.10 presents the maximum, minimum and average number of non-dominated solutions found by MOSAGA with different parameter settings in 50 runs. The running time is set to 60 seconds for each run. The setting of population size |*Pop*| = 50 and the starting temperature t_{max} = 50 provides the best solutions is thus selected in the proposed algorithms.

Table 6. 10 Comparison of MOSAGA with different parameter settings for solving the NSF network (|*NDS*|: the number of non-dominated solutions; Running time = 60 seconds in each run)

|*Pop*|	t_{max}	Max |*NDS*|	Min |*NDS*|	Average |*NDS*|
50	**50**	**16**	**16**	**16**
50	100	16	13	14
100	50	16	12	15
100	100	16	13	14

In MOSAGA, the population size is set to 50, the number of generations is set to 500, and the crossover rate is set to 1. These are also kept the same as those in MOEA1 and MOEA2 for a fair comparison. Instead of using a fixed mutation rate 0.3 in MOEA1 and MOEA2, an adaptive mutation rate is employed in MOSAGA, see the description in Section 6.3.1.4. Based on these initial tests, k = 25 is set to generate the routing table of alternative paths in the mutation operation. In the SA process of MOSAGA, the parameters are set as follows: the initial temperature t_{max} = 50, the final temperature t_{min} = 5, the temperature decrement

$t_{step} = 5$, and the temperature threshold $t_c = 25$. These parameter settings are set as the same as those in (Li and Landa-Silva, 2008). For the simplicity and fair comparisons, the same parameter settings are kept for MOSAGA and MOSAGLS on all the tested instances in the experiments unless otherwise stated.

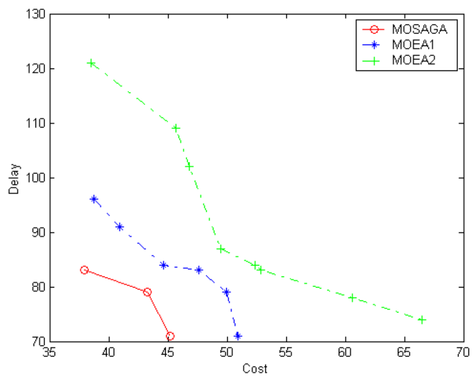
6.3.2.2 Comparisons on Random Networks

In order to visually analyse the impact of the SA strategies, the adaptive mutation and the local search on the performance of the proposed algorithms, a series of experiments has been carried out on random networks with two objectives including the cost and delay. Four random networks have been generated with different network sizes ($|V| = 50$ and $|V| = 100$) and group sizes (number of destination nodes $|R| = 20\% \times |V|$ and $|R| = 30\% \times |V|$).

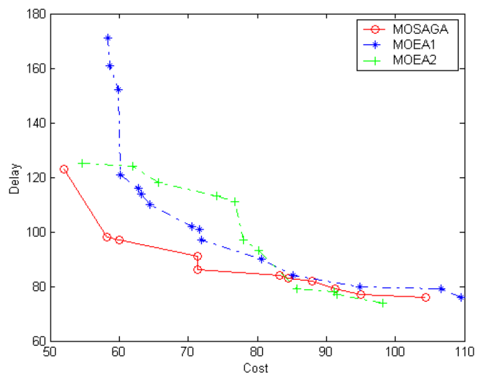
1) The Impact of the SA Strategies

The first group of experiments is designed to identify the impact of the SA strategies in the proposed algorithm. For a fair comparison, MOSAGA without the local search is firstly compared with other two evolutionary algorithms, MOEA1 and MOEA2, by using the same parameter settings as mentioned in the above section. The non-dominated solutions found by the three algorithms after 50 runs are shown in Figure 6.12, with their average computing time in Table 6.11.

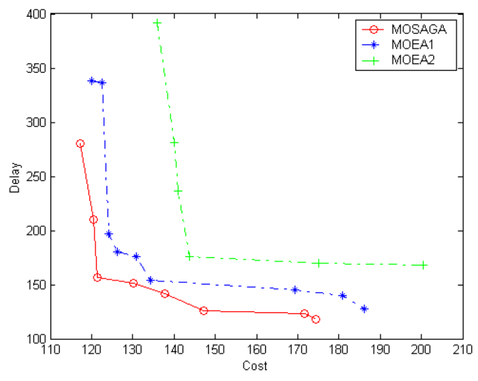
The experimental results clearly show that the MOSAGA algorithm obtains a set of better non-dominated solutions compared with those from the two MOEA algorithms on 3 out of 4 problems, i.e. Figure 6.12(a), (c) and (d). For the problem in Figure 6.12(b), MOSAGA is able to find better non-dominated solutions except those three solutions found by MOEA2. However, for all other three problems, MOEA2 performs the worst. With regard to the computing time, MOSAGA requires the least computing time. This demonstrates that the SA strategies in MOSAGA can guide the search direction of genetic algorithm to find better solutions in less computing time in comparison with the MOEA algorithms.



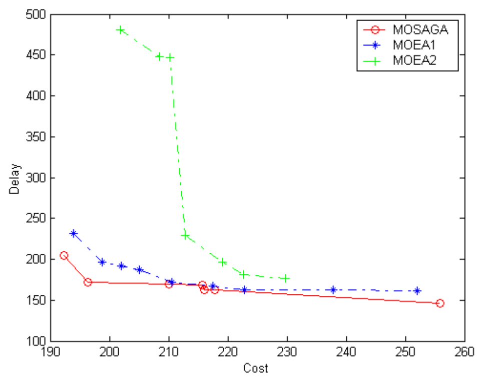
(a) Cost vs. Delay (|*V*| = 50, |*R*| = 10)



(b) Cost vs. Delay (|*V*| = 50, |*R*| = 15)



(c) Cost vs. Delay (|*V*| = 100, |*R*| = 20)



(d) Cost vs. Delay (|*V*| = 100, |*R*| = 30)

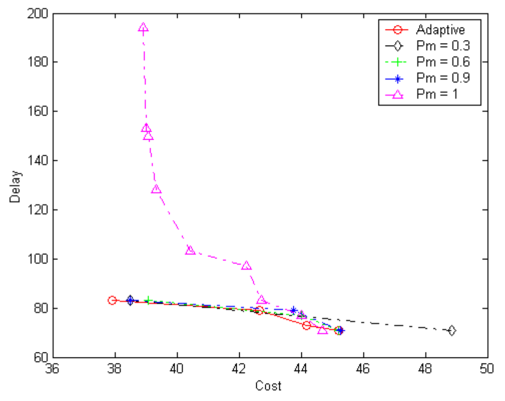
Figure 6. 12 The non-dominated solutions found by MOSAGA, MOEA1 and MOEA2 on random networks with two objectives in 50 runs

Table 6. 11 Average computing time of MOSAGA, MOEA1 and MOEA2 on the random networks

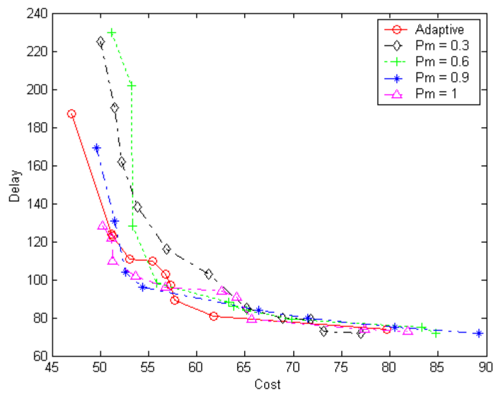
Network size	Group size	Computing time (sec)		
		MOSAGA	MOEA1	MOEA2
50	10	**7.068**	25.211	25.544
50	15	**6.208**	34.901	36.703
100	20	**13.342**	88.411	96.489
100	30	**18.338**	126.517	132.015

2) The Impact of the Adaptive Mutation

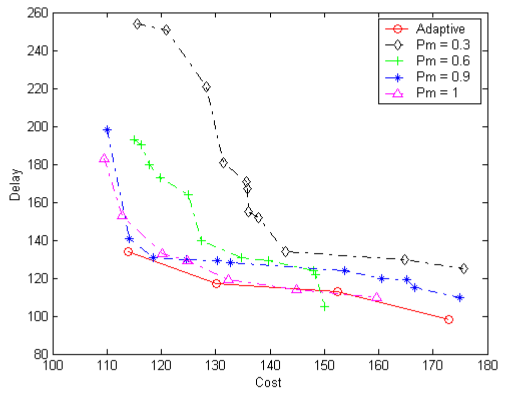
In the second group of experiments, the effect of the adaptive mutation rate is tested by comparing it with four fixed mutation rates, i.e. P_m = 0.3, 0.6, 0.9, and 1 in MOSAGA. The non-dominated solutions found by MOSAGA with different mutation rates after 50 runs are shown in Figure 6.13. In general, MOSAGA with the adaptive mutation rate has the best overall performance among all the other variants with fixed mutation rates, although the Pareto fronts found by variants of the algorithm interweave at certain part for some networks, i.e. Figure 6.13(b) and 6.13(d). With fixed mutation rates, MOSAGA occasionally finds several better non-dominated solutions, but fails to obtain the majority of solutions at the Pareto front found by MOSAGA with the adaptive mutation rate for all four networks. The adaptive mutation rate based on both the temperature and the quality of the new solution and the current population makes the search more effective.



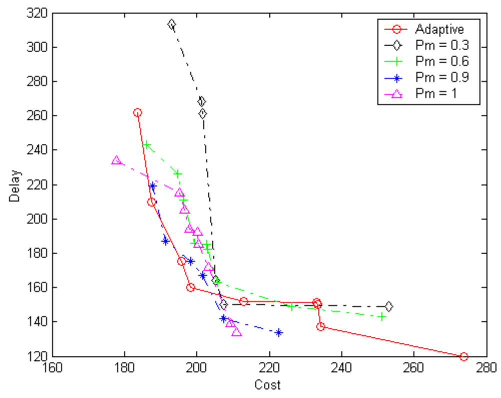
(a) Cost vs. Delay (|V| = 50, |R| = 10)



(b) Cost vs. Delay (|V| = 50, |R| = 15)



(c) Cost vs. Delay (|V| = 100, |R| = 20)

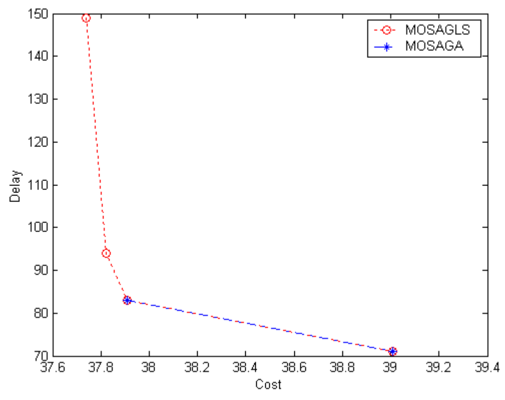


(d) Cost vs. Delay (|V| = 100, |R| = 30)

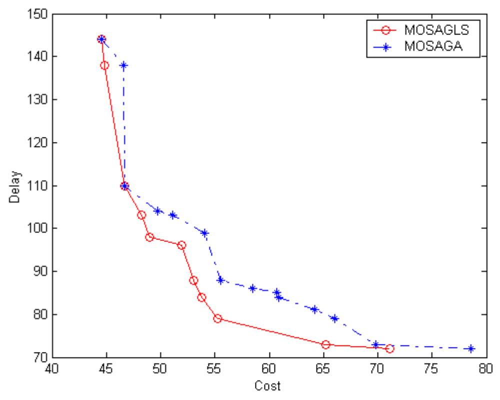
Figure 6. 13 The non-dominated solutions found by MOSAGA with different mutation rates on random networks with two objectives in 50 runs

3) The Impact of the Local Search

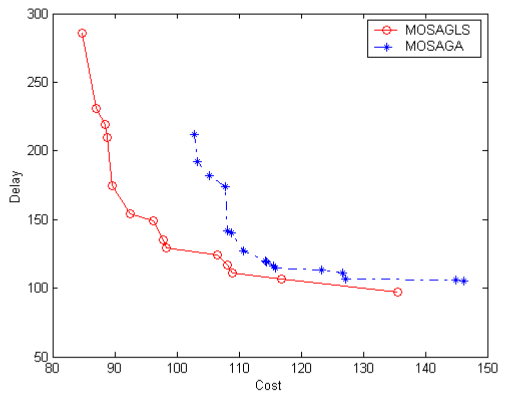
In the third group of experiments, the MOSAGA and MOSAGLS algorithms are compared to show the effect of the local search in the proposed algorithms. The local search in MOSAGLS stops after a number of nodes (set as 10 here) have been flipped or a better tree has been found.



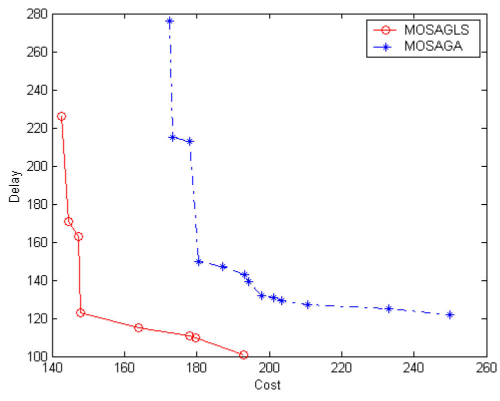
(a) Cost vs. Delay (|*V*| = 50, |*R*| = 10)



(b) Cost vs. Delay (|*V*| = 50, |*R*| = 15)



(c) Cost vs. Delay (|*V*| = 100, |*R*| = 20)



(d) Cost vs. Delay (|*V*| = 100, |*R*| = 30)

Figure 6. 14 The non-dominated solutions found by MOSAGLS and MOSAGA
on random networks with two objectives in 50 runs

The non-dominated solutions found by MOSAGA and MOSAGLS for the four random networks in 50 runs are shown in Figure 6.14, clearly showing the improvement made by the local search in MOSAGLS. For the small network in Figure 6.14(a), MOSAGLS found four non-dominated solutions, while MOSAGA found only 2 non-dominated solutions. For the same networks with larger group size and for networks of large size, it is obvious that MOSAGLA dominates MOSAGA by finding much better non-dominated solutions. This demonstrates the efficiency of the local search on the performance of the proposed MOSAGLS algorithm.

Table 6.12 presents the average computing time of MOSAGA and MOSAGLS on the four random networks, showing (not surprisingly) that MOSAGLS finds better solutions at the expenses of longer computing time compared with the MOSAGA algorithm.

Table 6. 12 Average computing time of MOSAGA and MOSAGLS on the random networks

Network size	Group size	Computing time (sec)	
		MOSAGA	MOSAGLS
50	10	0.573	4.076
50	15	0.684	5.331
100	20	1.62	22.198
100	30	1.616	36.274

6.3.2.3 Comparisons on Benchmark and Random Problems

1) Results on the NSF Network with Four Objectives

Based on the above observations, the performance of MOSAGLS and MOSAGA are compared on the benchmark NSF network (see Figure 3.2 in Chapter 3) with the experimental results of MOEA algorithms. As mentioned in Section 6.3.2.1, the optimal Pareto front of 16 solutions for the NSF problem has been found in (Crichigno and Baran, 2004b), concerning four objectives (1), (2), (3) and (4). The same four objectives are thus considered in this group of experiments. The adaptive mutation rate has been applied in MOSAGLS and MOSAGA. Table 6.13 presents the maximum, minimum and average number of non-dominated solutions and the lowest percentage of non-dominated solutions found by each algorithm belonging to *PF* in 50 runs. The same parameter settings are kept in the four algorithms as mentioned in Section 6.3.2.1, and the same generation number for each run is 500.

Table 6. 13 Number of non-dominated solutions found by different algorithms on the NSF network with four objectives (|*NDS*|: the number of non-dominated solutions; % in *PF*: the lowest percentage of non-dominated solutions belonging to *PF*, i.e. Min{|*NDS*|}/|*PF*|, here |*PF*| = 16)

Algorithms	Max |*NDS*|	Min |*NDS*|	Average |*NDS*|	% in *PF*
MOSAGLS	**16**	**16**	**16**	**100%**
MOSAGA	**16**	**16**	**16**	**100%**
MOEA1	16	14	15	87.5%
MOEA2	16	10	15	62.5%

For this small NSF problem, the proposed MOSAGLS and MOSAGA algorithms outperform the two MOEA algorithms, i.e. always find all 16 solutions in the optimal Pareto front in 50 runs. This, together with the above observations, indicates that SA based MOSAGA and MOSAGLS with adaptive mutation rate are more effective than the two conventional MOEA algorithms for the multi-objective multicast routing problem.

2) Results on the NTT Network with Five Objectives

MOSAGLS, MOEA1, and MOEA2 are compared on the NTT network in Figure 3.3 (see Section 3.5.4) with respect to all the five objectives defined in Section 6.3. Two different multicast groups have been tested, as shown in Table 6.3. As the optimal Pareto front for the NTT network with all the five objectives is not known, a six-step procedure devised in (Diego and Baran, 2005) is used to obtain an approximation of the Pareto front for each problem. The six-step procedure has been used in the previous work (Xu and Qu, 2011) and is presented as follows:

1) Each of the algorithms tested is run 10 times. Here, three algorithms have been tested, i.e. MOSAGLS, MOEA1, and MOEA2.
2) For each algorithm, 10 sets of non-dominated solutions Y_1, Y_1, …, Y_{10} are obtained, one from each run.
3) For each algorithm, an aggregate population Y_T is obtained, i.e. $Y_T = \bigcup_{i=1}^{10} Y_i$.
4) The dominated solutions are removed from Y_T to obtain the non-dominated solution set for each algorithm, denoted by Y_{alg}, i.e. $Y_{MOSAGLS}$, Y_{MOEA1}, and Y_{MOEA2}.
5) A set of solutions Y' is obtained by combining all Y_{alg}, i.e. $Y' = \bigcup_{i=1}^{3} Y_{\mathrm{alg}i}$.
6) The non-dominated solutions in Y' are selected to form an approximation of the true optimal Pareto solution set, called Y_{PF}.

Table 6.14 presents the total number non-dominated solutions in Y_{PF} obtained by the three algorithms on the two NTT instances by using the six-step procedure. For a fair and comprehensive comparison, all algorithms tested have been given the same computational time, i.e. 160 seconds and 320 seconds, respectively, for each run.

Table 6. 14 The total number of non-dominated solutions in Y_{PF} calculated by using the six-step procedure for the two instances of the NTT network with respect to all five objectives

Running Time	160 seconds		320 seconds	
Multicast Group	group1	group2	group1	group2
$\mid Y_{PF} \mid$	44	16	42	17

Based on this approximation of the optimal Pareto solution set Y_{PF}, MOSAGLS, MOEA1 and MOEA2 are compared on the NTT benchmark problem with two different group sizes and five objectives. Each algorithm was run 10 times on each instance. To have an insight of the quality of solutions obtained by each algorithm with regard to Y_{PF}, the following notations are used:

- $\in Y_{PF}$: the average number of solutions that are in Y_{PF} found by each algorithm in each run;
- $Y_{PF}\succ$: the average number of solutions found by each algorithm that are dominated by Y_{PF} in each run;
- $|\bar{Y}|$: the average number of solutions found by each algorithm in each run, i.e. $\sum_{i=1}^{10}|Y_i|/10$;
- $\%Y_{PF}$: the average percentage of solutions in Y_{PF} found by each algorithm in each run, i.e. $\in Y_{PF} / |Y_{PF}|$.

Table 6. 15 Results of different algorithms for solving the NTT network with small multicast group 1

Algorithms	Running Time = 160 seconds							
	$\in Y_{PF}$	$Y_{PF}\succ$	$\|\bar{Y}\|$	$\% Y_{PF}$	Coverage	$Y_{MOSAGLS}$	Y_{MOEA1}	Y_{MOEA2}
MOSAGLS	4.1	14.6	18.7	9.32%	$Y_{MOSAGLS}$	\	0.57	0.8
MOEA1	1.4	25.7	27.1	3.18%	Y_{MOEA1}	0.33	\	0.32
MOEA2	1.8	14.1	15.9	4.09%	Y_{MOEA2}	0.56	0.5	\
Algorithms	Running Time = 320 seconds							
	$\in Y_{PF}$	$Y_{PF}\succ$	$\|\bar{Y}\|$	$\% Y_{PF}$	Coverage	$Y_{MOSAGLS}$	Y_{MOEA1}	Y_{MOEA2}
MOSAGLS	5	17.1	22.1	12.9%	$Y_{MOSAGLS}$	\	0.57	0.82
MOEA1	4.8	18.6	23.4	11.4%	Y_{MOEA1}	0.71	\	0.96
MOEA2	1.9	13	14.9	4.75%	Y_{MOEA2}	0.57	0.49	\

Table 6. 16 Results of different algorithms for solving the NTT network with large multicast group 2

Algorithms	Running Time = 160 seconds							
	$\in Y_{PF}$	$Y_{PF}\succ$	$\|\bar{Y}\|$	$\% Y_{PF}$	Coverage	$Y_{MOSAGLS}$	Y_{MOEA1}	Y_{MOEA2}
MOSAGLS	7.3	11.2	18.5	45.63%	$Y_{MOSAGLS}$	\	1	1
MOEA1	0	17.8	17.8	0	Y_{MOEA1}	0	\	0.64
MOEA2	0	10.9	10.9	0	Y_{MOEA2}	0	0	\
Algorithms	Running Time = 320 seconds							
	$\in Y_{PF}$	$Y_{PF}\succ$	$\|\bar{Y}\|$	$\% Y_{PF}$	Coverage	$Y_{MOSAGLS}$	Y_{MOEA1}	Y_{MOEA2}
MOSAGLS	6.8	13.5	20.3	40%	$Y_{MOSAGLS}$	\	0.95	1
MOEA1	0.1	16	16.1	0.59%	Y_{MOEA1}	0.06	\	1
MOEA2	0	12.1	12.1	0	Y_{MOEA2}	0	0	\

Table 6.15 and 6.16 present the results obtained for the two instances of the NTT network. Both tables show that MOSAGLS found more non-dominated solutions $\in Y_{PF}$ than that of MOEA algorithms in each run. To evaluate the approximation to the Pareto front, a measurement of calculating the coverage (Zitzler

and Thiele, 1999) between the non-dominated solution set obtained by each algorithm is applied. Both tables show that the non-dominated solutions found by MOSAGLS cover the majority of the non-dominated solutions found by two MOEA algorithms. This becomes more obvious for the NTT network with large group size, where MOSAGLS obtains almost all non-dominated solutions, and the MOEA algorithms find just one or even no solution that belongs to Y_{PF}.

3) Results on Random Networks with Five Objectives

Finally, the effectiveness and efficiency of MOSAGLS upon a set of random networks of different characteristics with respect to all five objectives are tested in comparison with MOEA1 and MOEA2. 10 random graphs for each network size of $|V| = 50$ and $|V| = 100$, and with different group sizes ($|R| = 20\%\times|V|$ and $|R| = 30\%\times|V|$) are generated. Therefore, there are 4 types of network and 40 random networks in total.

For a fair comparison, the same running time, i.e. 320 seconds, is set for all algorithms in each run. Table 6.17 presents the average number of non-dominated solutions $|\overline{Y}_{PF}|$ obtained from 10 runs by the three algorithms on the 10 random networks for each network type with different characteristics by using the six-step procedure described above.

Table 6. 17 The average number of non-dominated solutions on each type of random network

Network Size	$\|V\| = 50$		$\|V\| = 100$	
Group Size	$\|R\| = 10$	$\|R\| = 15$	$\|R\| = 20$	$\|R\| = 30$
$\|\overline{Y}_{PF}\|$	264.4	196.2	91.4	105.2

Table 6.18 and Table 6.19 present average quality of solutions found by the three algorithms from 10 runs with respect to the non-dominated solutions Y_{PF} obtained in each run on the 10 random graphs for each network type. Both tables show that MOSAGLS significantly outperforms the two MOEA algorithms by finding more non-dominated solutions in Y_{PF}. The non-dominated solution set obtained by MOSAGLS covers most of the non-dominated solutions found by MOEA1 and MOEA2 on average. This again demonstrates the effectiveness of the proposed MOSAGLS algorithm on the random networks with all five objectives concerned.

Table 6. 18 Comparison of different algorithms on random networks of $|V| = 50$ with different group sizes. (Computational time for each run is 320 seconds)

Algorithms	Group Size $\|R\| = 10$							
	$\in Y_{PF}$	$Y_{PF} \succ$	$\|\overline{Y}\|$	$\% Y_{PF}$	Coverage	$Y_{MOSAGLS}$	Y_{MOEA1}	Y_{MOEA2}
MOSAGLS	37.72	66.08	103.8	14.69%	$Y_{MOSAGLS}$	\	0.64	0.82
MOEA1	6.48	110.46	116.94	2.91%	Y_{MOEA1}	0.01	\	0.55
MOEA2	2.76	94.84	97.6	1.16%	Y_{MOEA2}	0.01	0.19	\
Algorithms	Group Size $\|R\| = 15$							
	$\in Y_{PF}$	$Y_{PF} \succ$	$\|\overline{Y}\|$	$\% Y_{PF}$	Coverage	$Y_{MOSAGLS}$	Y_{MOEA1}	Y_{MOEA2}
MOSAGLS	30.3	61.9	92.2	15.91%	$Y_{MOSAGLS}$	\	0.7	0.84
MOEA1	3.26	95.7	98.96	1.67%	Y_{MOEA1}	0.04	\	0.76
MOEA2	1.38	85.76	87.14	0.54%	Y_{MOEA2}	0.02	0.14	\

Table 6. 19 Comparison of different algorithms on random networks of $|V| = 100$ with different group sizes (Computational time for each run is 320 seconds)

Algorithms	Group Size $\|R\| = 20$							
	$\in Y_{PF}$	$Y_{PF} \succ$	$\|\overline{Y}\|$	$\% Y_{PF}$	Coverage	$Y_{MOSAGLS}$	Y_{MOEA1}	Y_{MOEA2}
MOSAGLS	2.15	16.36	18.51	2.9%	$Y_{MOSAGLS}$	\	0.9	0.97
MOEA1	0.05	17.60	17.66	0.05%	Y_{MOEA1}	0.004	\	0.71
MOEA2	0.004	14.72	14.72	0.004%	Y_{MOEA2}	0.008	0.13	\
Algorithms	Group Size $\|R\| = 30$							
	$\in Y_{PF}$	$Y_{PF} \succ$	$\|\overline{Y}\|$	$\% Y_{PF}$	Coverage	$Y_{MOSAGLS}$	Y_{MOEA1}	Y_{MOEA2}
MOSAGLS	0.928	4.146	5.074	0.93%	$Y_{MOSAGLS}$	\	0.9	1
MOEA1	0.13	5.62	5.75	0.08%	Y_{MOEA1}	0.002	\	0.7
MOEA2	0	4.59	4.59	0	Y_{MOEA2}	0	0.17	\

To summarize, the large amount of experiments on a range of multi-objective multicast routing problems with different features demonstrate the efficiency and effectiveness of our proposed MOSAGLS. The simulated annealing strategies in MOSAGA have shown not only to significantly improve the solution quality, but also to reduce the computing time required by the algorithm for the multi-objective multicast routing problem compared with conventional multi-objective evolutionary algorithms. The adaptive mutation contributes a better performance than that of fixed mutation rates. The local search further improves the performance of MOSAGLS. Experiments on both the benchmark problems and random networks demonstrate that the proposed MOSAGLS has the best performance among variants of algorithms, showing that MOSAGLS is able to obtain high quality solutions for the multi-objective multicast routing problem.

The investigations of the MOSAGLS algorithm have been reported in a paper which is now under the second review at the European Journal of Operational Research.

6.4 Summary

This chapter presents two novel hybrid metaheuristic approaches for the multi-objective multicast routing problem in telecommunications. The first approach is a hybrid evolutionary multi-objective simulated annealing (EMOSA) algorithm and its hybridization with variable neighbourhoods (VEMOSA) for solving the problem. Instead of employing a single neighbourhood structure, a number of neighbourhood structures are designed to deal with each specific objective during the evolution in the complex search space of the multi-objective problem. By comparing the VEMOSA with EMOSA algorithm and some standard multi-objective evolutionary algorithms in the literature, the efficiency and effectiveness of the VEMOSA algorithms are demonstrated via a set of extensive experiments for solving both the benchmark and random multi-objective multicast routing instances. The second approach is a simulated annealing based multi-objective genetic local search (MOSAGLS) for solving multi-objective multicast routing problems. Integrated with simulated annealing strategies, the hybrid multi-objective genetic local search algorithm is able to efficiently search towards the Pareto front and diversify the population with regard to the multiple objectives in the problem. A large amount of extensive simulations has been performed to evaluate the proposed MOSAGLS on both benchmarks and two sets of random networks with different objectives and different features. It demonstrates that the simulated annealing based adaptive mutation, the simulated annealing strategies of competition and the two-phase search direction tuning significantly improve the efficiency and effectiveness of the proposed algorithm compared against another two conventional MOEA algorithms in the literature with respect to both the solution quality and computing time for the multi-objective multicast routing problem. Compared with the simulated annealing based multi-objective genetic algorithm MOSAGA without local search, the local search in MOSAGLS further intensifies the search by exploring more promising neighbouring solutions but at a larger computational expense. Both proposed hybrid metaheuristic algorithms have shown to be ideal approaches for solving the multi-objective multicast routing problem, and are flexible to be extended to solve other multi-objective optimisation problems.

CHAPTER 7. Conclusions and Future Work

7.1 Summary of the Thesis

The work in this thesis investigates the important QoS multicast routing problem emerging from the increasing developments of multimedia applications in computer communication networks. With the development of computer networks, many real-world applications, such as video/audio conferencing, distance education, E-commerce and distributed games, etc., have emerged, which require the networks to support multicast routing. Multicast routing is a key communication technique to send information from the source to a set of destinations simultaneously. The underlying mathematic model of the problem is the Steiner tree problem in graphs which is a well-known NP-complete problem. Multicast routing problems with various QoS constraints have been proven to be NP-complete optimisation problems and thus attracted significant research attention for almost two decays in the literature. To solve the challenging QoS multicast routing problem, the optimisation techniques -- metaheuristics are used in this thesis. Metaheuristics are a family of high level frameworks of search strategies which are very useful and effective for tackling a wide range of complex combinatorial optimisation problems in the area of operational research.

The research in the thesis focuses on designing efficient and effective metaheuristic approaches for solving two QoS multicast routing problems, the DCLC multicast routing problem and the multi-objective multicast routing problem. In Chapter 4, four new metaheuristic algorithms, including variants of variable neighborhood descent search (VNDMR), a jumping particle swarm optimisation algorithm (JPSO), a hybrid GRASP algorithm, and a scatter search with path relinking algorithm (SSPR), have been proposed and investigated for the DCLC multicast routing problem. Intensive experimental results show that these proposed metaheuristic algorithms outperform other existing algorithms in the literature and are promising approaches for the problem. The three simple yet effective neighbourhood structures designed in VNDMR can effectively explore the search space of the complex DCLC multicast routing problem, thus VNDMR is capable of obtaining good search results in a short of time. The cooperative nature of particles in JPSO is the main characteristic for its success on both the Steiner tree problem and the DCLC multicast routing problem. The proposed JPSO has shown to be very robust. However, the shortcoming of JPSO is its relatively high computing time among the four metaheuristic approaches proposed in the thesis. The hybrid GRASP approach combined with VNDMR in the local search phase and the post-processing pilot method offers high quality solutions for the DCLC multicast routing problem efficiently. The SSPR algorithm which combines the scatter search and path relinking with a variable neighbourhood search algorithm as the improvement method has so far the best performance in terms of average tree cost on a set of random networks for the DCLC multicast routing problem.

In order to investigate the underlying landscape features of the DCLC multicast routing problem, the first landscape analysis of the DCLC multicast routing problem has been conducted in Chapter 5. Two landscape analysis techniques, the fitness distance correlation analysis and the auto-correlation analysis, have been used to analyze the landscapes of the problem on a set of problem instances generated based on the benchmark Steiner tree problems in the OR-library. The analysis reveals that the landscape of the DCLC multicast routing problem is highly instance dependent where different landscape characteristics can be observed. The delay constraint has shown a great influence on the underlying landscape features of the problem. The analysis of fitness distance landscape has been used to evaluate the performance of different local search methods for the DCLC multicast routing problem. In addition, the auto-correlation analysis has been used to compare the effectiveness of different neighbourhood operators for solving the problem of different sizes and delay bounds. According to the analysis, both the node-based neighbourhood and the link-based neighbourhood operators are effective and suitable for solving the DCLC multicast routing problem. The fitness landscape analysis techniques applied in this thesis have shown to be useful for analyzing the underlying properties of the DCLC multicast routing problem and predicting the effectiveness of local search algorithms and neighbourhood operators on the problem. The analysis aims to provide a case study of theoretically predicting the performance of metaheuristics, thus more advanced search algorithms on various related applications of multicast routing problems may be proposed in the future.

Two novel hybrid metaheuristic approaches, an evolutionary multi-objective simulated annealing algorithm with variable neighborhoods and a simulated annealing based genetic local search algorithm, have been proposed in Chapter 6 for the more complex multi-objective multicast routing problem. A large amount of experiments demonstrate the effectiveness and efficiency of both algorithms by obtaining more and better non-dominated Pareto optimal solutions for the problem in comparison with some traditional multi-objective genetic algorithms.

The main contributions of the research in the thesis are summarized in the following subsections.

7.1.1 Contributions on the DCLC Multicast Routing Problem

For the widely studied DCLC (delay-constrained least-cost) multicast routing problem, four new metaheuristic approaches have been designed and investigated:

– The thesis, for the first time, applies the VNS metaheuristic to the problem. Variants of variable neighbourhood descent search (a basic VNS) algorithms have been designed.
– The thesis, for the first time, proposes a jumping particle swarm optimisation algorithm for the problem.
– A new hybrid GRASP approach for the problem has been developed, where a VNS algorithm is applied in the local search phase and a pilot method is introduced as the post enhancement procedure.
– The thesis, for the first time, investigates a hybrid scatter search and path relinking metaheuristic for the problem.
– By using the multicast routing simulator MRSIM, intensive experiments have been carried out to evaluate the performance of these proposed metaheuristic approaches for the problem. Experimental results show that these proposed new metaheuristic approaches are competitive solutions to the problem and outperform other existing algorithms in the literature.
– The fitness landscape analysis on the problem has been conducted for the first time. Two fitness landscape methods, the fitness distance correlation analysis and the autocorrelation analysis, have been used on some benchmark instances to investigate the landscape characteristics of the problem.

7.1.2 Contributions on the Multi-objective Multicast Routing Problem

For the more complicated multi-objective multicast routing problem, two novel hybrid metaheuristic approaches have been investigated:

– A hybrid evolutionary multi-objective simulated annealing algorithm with variable neighbourhoods has been proposed for the first time for solving the problem. Compared with single neighbourhood based algorithm, the variable neighbourhoods specially designed for multiple objectives are capable of greatly improving the performance of the hybrid algorithm.

– A simulated annealing based multi-objective genetic local search has been investigated for solving the problem. The simulated annealing strategies in the hybrid multi-objective genetic local search algorithm have shown to efficiently search towards the Pareto front and diversify the population with regard to the multiple objectives in the problem.

– A large amount of experiments have been designed to evaluate the effectiveness and efficiency of the two algorithms in comparison to some traditional evolutionary algorithms in the literature.

7.1.3 List of Presentations

– Ying Xu, Variable neighbourhood descent search for the delay-constrained least-cost multicast routing problem. The Learning and Intelligent Optimisation (LION 3), Trento, Italy, Jan. 14-18, 2009.

– Ying Xu, Particle Swarm Optimisation for the Steiner Tree problem in Graphs and the Delay-constrained multicast routing problem. Student Conference in Operational Research, Lancaster University, UK, Mar. 27-29, 2009.

– Ying Xu, A GRASP approach for the Delay-constrained Multicast routing problem. The 4th Multidisplinary International Scheduling Conference (MISTA2009), Dublin, Ireland, Aug. 10-13, 2009

7.1.4 List of Publications

1. Qu R., Xu Y., Kendall G. 2009. A Variable Neighbourhood Descent Search Algorithm for Delay-Constrained Least-Cost Multicast Routing. In: Stützle T. (Ed.), Proceedings of Learning and Intelligent Optimisation (LION3), Lecture Notes in Computer Science, 5851. pp. 15-29. The contribution of this paper is shown in Chapter 4.
2. Xu Y., Qu R. 2009. A GRASP Approach for the Delay-constrained Multicast routing problem. In: Proceedings of the 4th Multidisplinary International Scheduling Conference (MISTA4). Dublin, Ireland. The contribution of this paper is shown in Chapter 4.
3. Xu Y., Qu R. 2011. Solving Multi-objective Multicast Routing Problems by Evolutionary Multi-objective Simulated Annealing Algorithms with Variable Neighbourhoods. Journal of the Operational Research Society. 62, 313-325. This journal paper contributes to Chapter 6.
4. Qu R., Xu Y. A Simulated Annealing based Genetic Local Search Algorithm for Multi-objective Multicast Routing Problems. Under the second review at the European Journal of Operational Research, May 2010. The contribution of this work is shown in Chapter 6.
5. Qu R., Xu Y., Castro-Gutiérrez J., Landa-Silva D. Particle Swarm Optimisation for the Steiner Tree problem in Graphs and Delay-Constrained Multicast Routing Problems. Under review at the Journal of Heuristics, May 2010. The contribution of this paper is shown in Chapter 4.
6. Xu Y., Qu R. 2010. A Hybrid Scatter Search Meta-heuristic for Delay-constrained Multicast Routing Problems. Applied Intelligence. DOI: 10.1007/s10489-010-0256-x. The contribution of this paper is shown in Chapter 4.
7. Xu Y., Qu R. Fitness Landscape Analysis for the Delay-constrained Multicast Routing Problem. Under review at Computer &Communications, October 2010. The contribution of this paper is shown in Chapter 5.

7.2 Future Work

Several interesting research directions are worthy of further investigation in the future work. It will be interesting to extend these proposed single-objective metaheuristic approaches in Chapter 4 for the DCLC multicast routing problem to solve diverse QoS multicast routing problems with other various QoS constraints (e.g. the bandwidth, delay-variation, node degrees, and packet loss ratio) in real world multimedia applications. The investigation of metaheuristics for other challenging QoS multicast routing problems will also be interesting. For example, in this thesis, all proposed algorithms are for the static case where the states of the network are known and without change. In reality, networks are mostly dynamic with many nodes leaving and joining in the multicast group at various times. This leads to the more complicated dynamic multicast routing. The current proposed algorithms in this thesis only consider a single multicast session. However, multiple multicast requests may occur simultaneously in the network to share the limited network resources. This creates another interesting network optimisation problem, the multicast routing packing problem, to schedule the multiple multicast trees in the network so that the utilisation of network resources is minimised.

The fitness landscape analysis techniques discussed in Chapter 5 have shown to be useful for measuring the underlying characteristics of the DCLC multicast routing problem and predicting the performance of search algorithms and different neighbour operators. The fitness landscape analysis methods used in this thesis can be used to analyse the DCLC multicast routing problem of larger size or other QoS multicast routing problems with additional real life constraints to motivate the development of advanced metaheuristic algorithms. In addition, more advanced landscape analysis techniques can be conducted on both the DCLC multicast routing problem with larger set of instances and various other QoS multicast routing problems.

The proposed two new hybrid metaheuristic algorithms (VEMOSA and MOSAGLS) in Chapter 6 have shown to be promising approaches for solving the multi-objective multicast routing problem. Another interesting research direction would be to extend the two proposed hybrid multi-objective multicast routing approaches for solving the multi-objective multicast routing with more objectives besides the 5 objectives considered in this thesis. These hybridised metaheuristics can also be adapted and investigated for other multi-objective optimisation problems in the area of operational research.

Another interesting direction would be extending these proposed metaheuristic approaches for solving various multicast routing problems in different types of networks, such as the multicast support for mobile computing in wireless networks, and the multicast routing in optical networks. Due to the new features included in the scenario of different networks, variety of multicast routing problems can be defined with respect to diverse objectives and constraints. The proposed metaheuristic approaches can be adapted and investigated for these different multicast routing problems.

Furthermore, the performance of the proposed metaheuristic approaches may be further improved by hybridising them with some exact methods, such as linear programming, branch-and-bound, and dynamic programming, etc. Metaheuristics are ideal choices for solving combinatorial optimisation problems since metaheuristics are flexible and effective to explore a large search space within a short time while exact methods can explore a specific small area exhaustively. Combination of the strengths of both metaheuristics and exact methods may lead to better quality solutions for the QoS multicast routing problem as well as other combinatorial optimisation problems.

References

Aarts E.H.L., Ten Eikelder H.M.M. 2002. Simulated annealing. In Pardalos P.M., Resende M.G.C. (Eds.), Handbook of Applied Optimisation. Oxford: Oxford University Press, pp. 209-220.

Adibi M.A., Zandieh M., Amiri M. 2010. Multi-objective scheduling of dynamic job shop using variable neighbourhood search. Expert Systems with Applications. 37(1): 282-287.

Amiri A., Pirkul H. 1997. Routing and capacity assignment in backbone communication networks. Computers & Operations Research. 24(3): 275-287.

Ballardie A., Crowcroft J., Francis P. 1993. Core Based Trees (CBT) - an architecture for scalable interdomain multicast routing. In: Proceedings of ACM Sigcomm'93, pp.85-95. ACM SIGCOMM.

Balram S., Nazish H., Shweta J. 2010. Orthogonal simulated annealing for multiobjective optimization. Computers & Chemical Engineering. 34(10): 1618-1631.

Barahona F., Ladanyi L. 2006. Branch and cut based on the volume algorithm: Steiner trees in graphs and Max-Cut. RAIRO Operations Research. 40, 53-73.

Bastos M.P., Ribeiro C.C. 2001. Reactive tabu search with path relinking for the Steiner tree problem in graphs. In Ribeiro C.C., Hansen P. (Eds.), Essays and Surveys in Metaheuristics. Kluwer Academic Publishers, pp. 39-58.

Bauer F., Varma A. 1995. Degree-constrained multicasting in point-to-point networks. In: proceedings of IEEE INFOCOM, Boston. pp. 369-376.

Beasley J.E. 1990. OR-Library: distributing test problems by electronic mail. Journal of the Operational Research Society. 41(11): 1069-1072. http://people.brunel.ac.uk/~mastjjb/jeb/orlib/ steininfo.html.

Beausoleil R.P. 2006. MOSS: Multiobjective scatter search applied to non-linear multiple criteria optimisation. European Journal of Operational Research. 169(2): 426-449.

Bertsekas D., Gallager R. 1992. Data Networks. Second Edition. Prentice Hall.

Binato S., Hery W.J., Loewenstern D.M., Resende M.G.C. 2001. A greedy randomized adaptive search procedure for job shop scheduling. In Hansen P., Ribeiro C. C. (Eds), Essays and surveys on metaheuristics. Kluwer Academic Publishers, pp. 59-79.

Black D., Carlson M., Davies E., Wang Z., Weiss W. 1998. An architecture for differentiated services. RFC: 2475.

Blum C., Blesa Aguilera M.J., Roli A., Sampels M. 2008. Hybrid metaheuristics: an emerging approach to optimisation. Springer.

Blum C., Roli A. 2003. Metaheuristics in combinatorial optimisation: overview and conceptual comparison. ACM Computing Surveys. 35(3): 268-308.

Boese K. 1995. Cost versus distance in the traveling salesman problem. Tech. Rep. TR-950018, UCLA CS Department.

Bouleimen K., Lecocq H. 2003. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its references multiple mode version. European Journal of Operational Research. 149, 268-281.

Braden R., Zhang L., Berson S., Herzog S., Jamin S. 1997. Resource reservation protocol (RSVP)-version 1 functional specification. RFC 2205.

Buriol L.S., Resende M.G.C., Ribeiro C.C., Thorup M. 2005. A hybrid genetic algorithm for the weight setting problem in OSPF/IS-IS routing. Networks. 46, 36-56.

Burke E.K., Curtois T.E., Post G., Qu R., Veltman B. 2008. A hybrid heuristic ordering and variable neighbourhood search for the nurse rostering problem. European Journal of Operational Research. 188(2): 330-341.

Burke E.K., Curtois T., Qu R., Vanden Berghe G. 2010. A scatter search for the nurse rostering problem. Journal of Operational Research Society. 61: 1667-1679.

Canuto S.A., Resende M.G.C., Ribeiro C.C. 2001. Local search with perturbations for the prize-collecting Steiner tree problem in graphs. Networks. 38, 50-58.

Casner S., Deering S. 1992. First IETF Internet audiocast. ACM Computer Communications Review. 22(3): 92-107.

Castro-Gutiérrez J.P., Landa-Silva D., Moreno-Pérez J.A. 2009. Exploring feasible and infeasible regions in the vehicle routing problem with time windows using a multi-objective particle swarm optimisation approach. In Melian B., Moreno-Perez J.A., Moreno-Vega J.M., Pelta D., Krasnogor N. (Eds.), Studies in Computational Intelligence. 236, 103-114.

Cayley A. 1989. A theorem on trees. Journal of Math. 23, 376-378.

Chen S., Günlük O., Yener B. 1998. Optimal packing of group multicastings. In: Proceedings IEEE INFOCOM'98. pp. 980-987.

Chen S., Nahrstedt K. 1998. An overview of quality-of-service routing for the next generation high-speed networks: problems and solutions. IEEE Network. 12(6):64-79.

Chen S., Nahrstedt K., Shavitt Y. 2000. A QoS-aware multicast routing protocol. IEEE Journal on Selected Areas in Communications. 18(12):2580-2592.

Chen S., Shavitt Y. 2004. A scalable distributed QoS multicast routing protocol. In: IEEE International Conference on Communications. pp. 1161-1165.

Coello C.A.C., Van Veldhuizen D.A., Lamont G.B. 2002. Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publishers.

Consoli S., Moreno-Perez J.A., Darby-Dowman K., Mladenović N. 2008. Discrete particle swarm optimisation for the minimum labelling Steiner tree problem. In Krasnogor N., Nicosia G., Pavone M., Pelta D. (Eds), Nature Inspired Cooperative Strategies for Optimisation, 129 of Studies in Computational Intelligence, pp. 313-322, Springer-Verlag.

Cook W.J., Cunningham W.H., Pulleyblank W.R., Schrijver A. 1998. Combinatorial Optimisation. Wiley, New York.

Cormen T.H., Leiserson C.E., Revest R.L. 1997. Introduction to algorithms. MIT Press, Cambridge.

Corne D.W., Knowles J.D. 2000. The Pareto envelop-based selection algorithm for multiobjective optimisation. In: Proceedings of sixth international conference on parallel problem solving from Nature, pp. 839-848, Springer, Berlin.

Costa A.M., Cordeau, J.-F., Laporte G. 2006. Exact and approximate algorithms for a class of Steiner tree problems arising in network design and lot sizing. 4th US-European Workshop on Logistics and Supply Chain Management, Hamburg.

Crichigno J., Baran B. 2004a. Multiobjective multicast routing algorithm. In: Lecture Notes in Computer Science, 3124. pp. 1029-1034.

Crichigno J., Baran B. 2004b. Multiobjective multicast routing algorithm for traffic engineering. In: Proceedings of the 13th International Conference on Computer Communications and Networks, pp. 301-306.

Cui X., Lin C., Wei Y. 2003. A Multiobjective Model for QoS Multicast Routing Based on Genetic Algorithm. In: Proceedings of International Conference on Computer Networks and Mobile Computing (ICCNMC'03), pp. 49-53.

Cung V.D., Mautor T., Michelon P., Tavares A. 1997. A scatter search based approach for the quadratic assignment problem. In: Proceedings of IEEE International Conference on Evolutionary Computation. pp. 165-169.

Czyzak P., Jaszkiewicz A. 1998. Pareto simulated annealing - a metaheuristic technique for multiobjective combinatorial optimisation. Journal of Multi-Criteria Decision Analysis. 7, 34-37.

Deb K., Agrawal S., Pratap A., Meyarivan T. 2000. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation: NSGA-II. In: Proceedings of the Parallel Problem Solving from Nature VI Conference, Paris, France, pp. 849-858.

Deb K. 2001. Multiobjective optimisation using evolutionary algorithms. Wiley-Interscience Series in Systems and Optimisation. John Wiley & Sons, Chichester.

Deb K. 2005. Multi-objective optimisation. In Burke E.K., Kendall G. (Eds), Search Methodologies: Introductory Tutorials in Optimisation and Decision Support Methodologies. Springer. pp. 273-316.

Deering S.E. 1989. Host extensions for IP multicasting. RFC 1112.

Deering S. *et al*. 1996. The PIM architecture for wide-area multicast routing. IEEE/ACM Transaction on Networking. 4, 153-162.

Diot C., Dabbous W., Crowcroft J. 1997. Multipoint communication: a survey of protocols, functions, and mechanisms. IEEE Journal on Selected Areas in Communications. 15(3): 277-290.

Doerner K., Gutjahr W.J., Hartl R.F., Strauss C., Stummer C. 2001. Ant conony optimisation in multiobjective portofolio selection. In: Proceedings of the 4th Metaheuristics International Conference. Porto, Portugal, pp. 243-248.

Donoso Y., Fabregat R., Marzo J.L. 2004a. Multiobjective optimisation algorithm for multicast routing with traffic engineering. IEEE 3rd International Conference on Networking (ICN'2004).

Donoso Y., Fabregat R., Marzo J.L. 2004b. Multi-Objective optimisation model and heuristic algorithm for multipath routing of static and dynamic multicast group. III Workshop on MPLS Networks. Girona, Spain.

Dorigo M. 1992. Optimisation, Learning, and Natural Algorithms. PhD Thesis. Politecnico di Milano, Italy.

Dorigo M., Di Caro G. 1999. The ant colony optimisation meta-heuristic. In: Corne D., Dorigo M., Glover F. (Eds.), New Ideas in Optimisation. London: McGraw-Hill. pp.11-32.

Dorigo M., Stützle T. 2002. The ant colony optimisation metaheuristic: algorithms, applications, and advances. In Glover F., Kochenberger G. (Eds.), Handbook of Metaheuristics, volume 57 of International Series in Operations Research & Management Science. Kluwer Academic Publishers, pp. 251-285.

Drias H., Khabzaoui M. 2001. Scatter search with random walk strategy for SAT and MAX-W-SAT problems. Lecture Notes in Computer Science, 2070. pp. 35-44.

Duin C.W., Voß S. 1997. Efficient path and vertex exchange in Steiner tree algorithms. Networks. 29, 89-105.

Duin C.W., Voß S. 1999. The pilot method: a strategy for heuristic repetition with application to the Steiner problem in graphs. Networks. 34, 181-191.

Ehrgott M., Gandibleux X. 2000. A survey and annotated bibliography of multiobjective combinatorial optimisation. OR Spektrum. 22(4): 425-460.

Eppstein D. 1998. Finding the k shortest paths. SIAM Journal on Computing. 28(2): 652-673.

Fabregat R., Donoso Y., Solano F., Marzo J.L. 2004. Multitree routing for multicast flows: a genetic algorithm approach. 7th Catalan Conference on Artificial Intelligence.

Fabregat R., Donoso Y., Baran B., Solano F., Marzo J.L. 2005. Multi-objective optimisation scheme for multicast flows: a survey, a model and a MOEA solution. In: Proceedings of the 3rd international IFIP/ACM Latin American Conference on Networking (LANC'05), New York, pp. 73-86.

Faloutsos M., Banerjea A., Pankaj R. 1998. QoSMIC: Quality of Service sensitive Multicast Internet protocol. In: Proceedings of the ACM SIGCOMM'98. pp. 144-153.

Feo T.A., Resende M.G.C. 1995. Greedy randomized adaptive search procedures. Journal of Global Optimisation. 6, 109-133.

Festa P., Resende M.G.C. 2009a. An annotated bibliography of GRASP, part I: algorithms. International Transactions in Operational Research. 16, 1-24.

Festa P., Resende M.G.C. 2009b. An annotated bibliography of GRASP, part II: applications. International Transactions in Operational Research. 16, 131-172.

Floyd R.W. 1962. Algorithm 97: Shortest paths. Communications of the ACM. 5, 345.

Fraser A. S. 1957. Simulation of genetic systems by automatic digital computers. I. Introduction. Australian Journal of Biological Sciences. 10, 484-491.

Gambardella L.M., Taillard E., Agazzi G. 1999. MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows. In: Corne D., Dorigo M., Glover F. (Eds.), New Ideas in Optimisation. London: McGraw-Hill, pp.63-76.

Gambardella L.M., Dorigo M. 2000. Ant colony system hybridized with a new local search for the sequential ordering problem. INFORMS Journal on Computing. 12(3): 237-255.

Gandibleux X., Mezdaoui N., Fréville A. 1997. A tabu search procedure to solve multiobjective combinatorial optimisation problems. Lecture Notes in Economics and Mathematical Systems, 455. pp. 291-300, Springer.

Gandibleux X., Sevaux M., Sörensen K., T'kindt V. 2004. Metaheuristics for Multiobjective Optimisation. Lecture Notes in Economics and Mathematical Systems, 535. Springer.

Garey M.R., Graham R.L., Johnson D.S. 1977. The complexity of computing Steiner minimal trees. SIAM Journal on Applied Mathematics. 32, 835-859.

Garey M.R., Johnson D.S. 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, New York.

Gaspero L.D., Schaerf A. 2001. Tabu search techniques for examination timetabling. Letcure Notes in Computer Science, 2079. pp. 104-117.

Gaspero L.D., Schaerf A. 2007. A composite-neighborhood tabu search approach to the traveling tournament problem. Journal of Heuristics. 13(2):189-207.

Gendreau M. 2002. Recent Advances in Tabu Search. In: Ribeiro C.C., Hansen P. (Eds.), Essays and Surveys in Metaheuristics. Kluwer Academic Publishers, pp.369-377.

Gendreau M., Hertz A., Laporte G. 1994. A tabu search heuristic for the vehicle routing problem. Management Science. 40, 1276-1290.

Gendreau M., Potvin J.-Y. 2005. Metaheuristics in combinatorial optimisation. Annals of Operations Research. 140, 189-213.

Ghaboosi N., Haghighat A.T. 2006. A tabu search based algorithm for multicast routing with QoS constraints. In: Proceedings of the 9th International Conference on Information Technology, pp. 33-39.

Ghaboosi N., Haghighat A.T. 2007. A path relinking approach for delay-constrained least-cost multicast routing problem. In: Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence, pp. 383-390.

Glover F. 1986. Future paths for integer programming and links to artificial intelligence. Computers and Operations Research. 13, 533-549.

Glover F. 1997. Tabu search and adaptive memory programming – advances, applications and challenges. In Barr R.S., Helgason R.V., Kennington J.L. (Eds.), Advances in Metaheuristics, optimisation and Stochastic Modeling Technologies, pp. 1-75. Kluwer Academic Publishers.

Glover F. 1998. A template for scatter search and path relinking. In Hao J.K., Lutton E., Ronald E., Schoenauer M., Snyers D. (Eds.), Lecture Notes in Computer Science, 1363. pp. 13-54, Springer.

Glover F., Laguna M. 2002. Tabu search. In: Pardalos P.M., Resende M.G.C. (Eds.), Handbook of Applied Optimisation. Oxford University Press, pp. 194-208.

Glover F., Kochenberger G.A. 2003. Handbook of Meta-heuristics. Kluwer Academic Publishers.

Glover F., Laguna M., Martí R. 2000. Fundamentals of scatter search and path relinking. Control and Cybernetics. 29(3): 653-684.

Glover F., Laguna M., Martí R. 2003. Scatter search and path relinking: advances and applications. In Glover F., Kochenberger G.A. (Eds.), Handbook of Metaheuristics. pp. 1-35. Kluwer Academic Publishers.

Goldberg D.E. 1989. Genetic Algorithms in Search, Optimisation and Machine Learning. Addison-Wesley.

Greistorfer P. 2003. A tabu scatter search metaheuristic for the arc routing problem. Computers and Industrial Engineering. 44(2): 249-266.

Gruber M., Raidl G.R. 2005. Variable neighbourhood search for the bounded diameter minimum spanning tree problem. In: Hansen P., Mladenović N., Pérez J.A.M., Batista B.M., Moreno-Vega J.M. (Eds.), Proceedings of the 18th Mini Euro Conference on Variable Neighbourhood Search, Tenerife, Spain.

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.

Haberman B.K., Rouskas G.N. 1997. Cost, delay, and delay variation conscious multicast routing. Technique Report, TR-97-03, North Carolina State University.

Haghighat A.T., Faez K., Dehghan M., Mowlaei A., Ghahremani Y. 2004. GA-based heuristic algorithms for bandwidth-delay-constrained least-cost multicast routing. Computer Communications Journal. 27(1): 111-127.

Haidine A., Lehnert R. 2008. Multi-case multi-objective simulated annealing (MC-MOSA): new approach for adapt simulated annealing to multi-objective optimisation. International Journal of Information Technology. 4(3): 206-214.

Hamdan M., El-Hawary M.E. 2004. Multicast routing with delay and delay variation constraints using genetic algorithm. Canadian Conference on Electrical and Computer Engineering, pp. 2363-2366.

Hamiez J.P., Hao J.K. 2002. Scatter search for graph coloring. Lecture Notes in Computer Science, 2310. pp. 168-179.

Hansen P., Mladenovi´c N. 2003. Variable Neighbourhood Search. In Glover F., Kochenberger G.A. (Eds.), Handbook of Metaheuristics. pp. 145-184. Kluwer Academic Publishers.

Hart W.E., Krasnogor N., Smith J.E. 2003. Recent Advances in Memetic Algorithms and Related Search Technologies. Springer.

Hart W.E., Krasnogor N., Smith J.E. 2004. Recent advances in memetic algorithms, volume 166 of Studies in Fuzziness and Soft Computing. Springer.

Henderson D., Jacobson S.H., Johnson A.W. 2003. The theory and practice of simulated annealing. In Glover F., Kochenberger G.A. (Eds.), Handbook of Metaheuristics. pp. 287-319. Kluwer Academic Publishers.

Holland J.H. 1975. Adaptation in natural and artificial systems. Ann Arbor: University of Michigan Press.

Holland J.H. 1992. Genetic algorithms. Scientific American. 267(1): 66-72.

Höller H., Melián B., Voß S. 2008. Applying the pilot method to improve VNS and GRASP metaheuristics for the design of SDH/WDM networks. European Journal of Operational Research. 191(3): 691-704.

Hordijk W. 1996. A measure of landscapes. Evolutionary Computation. 4(4): 335-360.

Horn J., Nafpliotis N., Goldberg D.E. 1994. A niched Pareto genetic algorithm for multiobjective optimisation. In: Proceedings of the first IEEE conference on evolutionary computation, IEEE world congress on computational intelligence. pp. 82-87. Piscataway: IEEE Service Center.

Huang J., Liu Y. 2010. MOEAQ: a QoS-aware multicast routing algorithm for MANET. Expert Systems with Applications. 37(2010): 1391-1399.

Huang L., Han H., Hou J. 2007. Multicast routing based on the ant system. Applied Mathematical Sciences. 1(57): 2827-2838.

Hwang F.K., Richards D.S. 1992. Steiner tree problems. Networks. 22(1): 55-89.

Ishibuchi H., Murata T. 1998. A multi-objective genetic local search algorithm and its application to flowshop scheduling. IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews. 28(3): 392-403.

Jari K., Teemu N., Olli B., Michel G. 2007. An efficient variable neighbourhood search heuristic for very large scale vehicle routing problems. Computers and Operations Research. 34(9): 2743-2757.

Jaszkiewicz A. 2002. Genetic local search for multi-objective combinatorial optimisation. European Journal of Operational Research. 137(1): 50-71.

Jensen M.T. 2003. Reducing the run-time complexity of multiobjective EAs: the NSGA-II and other algorithms. IEEE Trans Evolutionary Computation. 7(5): 503-515.

Jia X. 1998. A distributed algorithm of delay-bounded multicast routing for multimedia applications in wide area networks. IEEE/ACM Transactions on Networking. 6(6): 828-837.

Jiang X. 1992. Routing broadband multicast streams. Computer Communications. 15(1): 45-51.

Jones D.F., Mirrazavi S.K., Tamiz M. 2002. Multiobjective metaheuristics: an overview of the current state of the art. European Journal of Operational Research. 137(1): 1-9.

Jones T., Forrest S. 1995. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In: Eshelman L.J. (Ed.) Proceedings of the 6th International Conference on Genetic Algorithms, pp. 184-192. Morgan Kaufmann, San Francisco, CA.

Kallel L., Naudts B., Reeves C.R. 2001. Properties of Fitness Functions and Search Landscapes. In Kallel L., Naudts B., Rogers A. (Eds.), Theoretical Aspects of Evolutionary Computing, pp. 175-206. Springer.

Kennedy J., Eberhart R.C. 1995. Particle swarm optimisation. Proceedings of IEEE International Conference on Neural Networks, IV, pp. 1942-1948. IEEE Service Center, Piscataway, NJ.

Kennedy J., Eberhart R.C. 1997. A discrete binary version of the particle swarm algorithm. In: Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, pp. 4104-4109. IEEE Service Center, Piscataway, NJ.

Kennedy J., Eberhart R.C., Shi Y. 2001. Swarm intelligence. Morgan Kaufmann, San Francisco, CA.

Kirkpatrick S., Gelatt C.D., Vecchi M.P. 1983. Optimisation by simulated annealing, Science. 220(4598): 671-680.

Koch T., Martin A. 1998. Solving Steiner tree problems in graphs to optimality. Networks. 32, 207-232.

Kompella V.P., Pasquale J.C., Polyzos G.C. 1993a. Multicast routing for multimedia communication. IEEE/ACM Transactions on Networking. 1(3): 286-292.

Kompella V.P., Pasquale J.C., Polyzos G.C. 1993b. Two distributed algorithms for the constrained Steiner tree problem. In: Proceedings of the Second International Conference on Computer Communications and Networking, pp.343-349.

Konak A., David W.C., Alice E.S. 2006. Multi-objective optimisation using genetic algorithms: A tutorial. Reliability Engineering and System Safety. 91(9): 992-1007.

Kou L., Markowsky G., Berman L. 1981. A fast algorithm for Steiner tree. Acta Informatica. 15(2): 141-145.

Krasnogor N. Smith J.E. 2005. A tutorial for competent memetic algorithms: model, taxonomy and design issues. IEEE Transactions on Evolutionary Computation, 9(5):474- 488.

Kruskal J. 1956. On the shortest spanning subtree of a graph and the travelling salesman problems. In: Proceedings of the American Mathematical Society. 7(1): 48-50.

Kulturel-konak S., Smith A.E., Norman B.A. 2006. Multi-objective tabu search using a multinomial probability mass function. European Journal of Operational Research. 169(3): 918-931.

Laguna M. 2002. ScatterSearch. In: Pardalos P.M., Resende M.G.C. (Eds.), Handbook of Applied Optimisation. Oxford University Press, pp.183-193.

Laguna M., Martí R. 1999. GRASP and path relinking for 2-layer straight line crossing minimisation. INFORMS Journal on Computing. 11(1): 44-52.

Landa-Silva J.D, Burke E.K, Petrovic S. 2004. An introduction to multiobjective metaheuristics for scheduling and timetabling. In: Xavier Gandibleux, Marc Sevaux, Kenneth Sorensen, Vincent T'kindt (Eds). Metaheuristic for Multiobjective Optimisation. Lecture Notes in Economics and Mathematical Systems, 535. pp. 91-129. Springer.

Li C, Cao C, Li Y, Yu Y. 2007. Hybrid of genetic algorithm and particle swarm optimisation for multicast QoS routing. IEEE International Conference on Control and Automation. IEEE Xplore Press, Guangzhou, China. pp. 2355-2359.

Li H, Landa-Silva D. 2008. Evolutionary multi-objective simulated annealing with adaptive and competitive search direction. In: Proceedings of the 2008 IEEE Congress on Evolutionary Computation (CEC 2008), IEEE Press: Hong Kong. pp. 3310-3317.

Lian Z., Gu X., Jiao B. 2006. A similar particle swarm optimisation algorithm for permutation flow shop scheduling to minimise makespan. Applied Mathematics and Computation. 175(1): 773-785.

Liu D.S., Tan K.C., Huang S.Y., Goh C.K., Ho W.K. 2008. On solving multiobjective bin packing problems using evolutionary particle swarm optimization. European Journal of Operational Research. 190(2): 357-382.

Liu H., Abraham A., Choi O., Moon S.H. 2006. Variable neighbourhood particle swarm optimisation for multi-objective flexible job-shop scheduling problems. Lecture Notes in Computer Science, 4247. pp. 197-204. Springer.

Liu J. 1999. The impact of neighbourhood size on the process of simulated annealing: computational experiments on the flowshop scheduling problem. Computers & Industrial Engineering. 37, 285-288.

Liu T.K., Chen C.H., Jyh-Horng Chou J.H. 2010. Optimization of short-haul aircraft schedule recovery problems using a hybrid multiobjective genetic algorithm. Expert Systems with Applications. 37(3): 2307-2315.

Lu G., Liu Z. 2000. Multicast routing based on ant-algorithm with delay and delay variation constraints. In: Proceedings of the 2000 IEEE Asia-Pacific Conference. pp. 243-246.

Ma Q., Steenkiste P. 1997. Quality-of-service routing for traffic with performance guarantees. In: Proceedings of IFIP Fifth International Workshop on Quality of Service. pp. 115-126.

Maenhout B., Vanhoucke M. 2006. New computational results for the nurse scheduling problem: a scatter search algorithm. Lecture notes in Computer Science, 3906. pp. 159 -170. Springer.

Manderick B., de Weger M., Spiessens P. 1991. The genetic algorithm and the structure of the fitness landscape. In: Proceedings of the 4th International Conference on Genetic Algorithms, pp. 143-150.

Maniezzo V., Carbonaro A. 2002. Ant colony optimisation: an overview. In: Ribeiro C.C., Hansen P. (Eds.), Essays and surveys in Metaheuristics. pp. 21-44. Kluwer Academic Publishers.

Martins F., Costa C.A.V. 2010. Multiobjective optimization with economic and environmental objective functions using modified simulated annealing. Computer Aided Chemical Engineering. 28, 919-924.

Martins S.L., Ribeiro C.C., Rosseti I. 2004. Applications and parallel implementations of metaheuristics in network design and routing. Lecture Notes in Computer Science, 3285. pp. 205-213.

Martins S.L., Resende M.G.C., Ribeiro C.C., Pardalos P. 2000. A parallel GRASP for the Steiner tree problemin in graphs using a hybrid local search strategy. Journal of Global Optimisation. 17, 267-283.

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. Computer Communications. 29(5): 563-581.

Mendoza J.E., Castanier B., Guéret C., Medaglia A.L., Velasco N. 2010. A memetic algorithm for the multi-compartment vehicle routing problem with stochastic demands. Computers & Operations Research. 37(11): 1886-1898.

Merkle D., Middendorf M., Schmeck H. 2002. Ant colony optimisation for resource-constrained project scheduling. IEEE Transactions on Evolutionary Computation. 6(4): 333-346.

Merz P., Freisleben B. 1998. Memetic algorithms and the fitness landscape of the graph bi-partitioning problem, In: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature - PPSN V. Lecture notes in Computer Science, 1498. pp. 765-774. Springer.

Miettinen K. 1999. Nonlinear Multiobjective Optimisation. Kluwer Academic Publishers.

Miller B.L., Shaw M.J. 1996. Genetic algorithms with dynamic niche sharing for multimodal function optimisation. In: Proceedings of the third International Conference on Evolutionary Computation. pp. 786-791.

Minoux M. 1989. Network synthesis and optimum network design problems: models, solution methods and applications. Networks. 19(3): 313-360.

Mladenović N., Hansen P. 1997. Variable neighbourhood search. Computers and Operations Research. 24, 1097-1100.

Moreno-Pérez J.A., Castro-Gutiérrez J.P., Martinez-Garcia F.J., Melián B., Moreno-Vega J.M., Ramos J. 2007. Discrete particle swarm optimisation for the p-median problem. In: Proceedings of the 7th Metaheuristics International Conference, Montreal, Canada.

Moscato P., Cotta C. 2003. A gentle introduction to memetic algorithms. In: Glover F., Kochenberger G.A. (Eds.), Handbook of Metaheuristics, pp. 105-144. Kluwer Academic Publishers.

Moy J. 1994. MOSPF: analysis and experience. RFC 1585.

Murata T., Ishibuchi H. 1995. MOGA: multi-objective genetic algorithms. In: Proceedings of the 1995 IEEE international conference on evolutionary computation. pp. 289-294.

Noronha C., Tobagi F. 1994. Optimum routing of multicast streams. In: Proceedings of the 13th IEEE Networking for Global Communications (INFOCOM'94). pp. 865-873.

Ochoa G., Qu R., Burke E.K. 2009. Analyzing the landscape of a graph based hyper-heuristic for timetabling problems, In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation (GECCO'09). pp. 341-348.

Oliveira C.A.S., Pardalos P.M. 2005. A survey of combinatorial optimisation problems in multicast routing. Computers and Operations Research. 32(8): 1953-1981.

Onwubolu G.C., Clerc M. 2004. Optimal path for automated drilling operations by a new heuristic approach using particle swarm optimisation. International Journal of Production Research. 42(3) 473-491.

Osman, I. H. Kelly, J. P., 1996. Meta-heuristics: Theory and Applications. Kluwer Academic Publishers.

Papadimitriou C. H., Steiglitz K. 1982. Combinatorial Optimisation - Algorithms and Complexity. Dover Publications.

Pasquale J.C., Polyzos G.C., Xylomenos G. 1998. The multimedia multicasting problem. ACM Multidedia Systems. 6(1): 43-59.

Paul P., Raghavan S.V. 2002a. Survey of QoS routing. In: Proceedings of the 15th international Conference on Computer communication. pp. 50-75.

Paul P., Raghavan S.V. 2002b. Surveys of multicast routing algorithms and protocols. In: Proceedings of the 15th international Conference on Computer communication. pp. 902-926.

Prais M., Ribeiro C.C. 2000. Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. INFORMS Journal on Computing. 12(3): 164-176.

Prim R.C. 1957. Shortest connection networks and some generalizations. Bell System Technical Journal. 36, 1389-1401.

Pinto D., Baran B. 2005. Solving Multiobjective Multicast Routing Problem with a new Ant Colony Optimisation Approach. In: Proceedings of the 3rd international IFIP/ACM Latin American conference on Networking. pp. 11-19.

Qu R., Xu Y., Kendall G. 2009. A variable neighbourhood descent search algorithm for delay-constrained least-cost multicast routing. In: Stützle T. (Ed.), Proceedings of Learning and Intelligent Optimisation (LION3), Lecture Notes in Computer Science, 5851. pp. 15-29. Springer.

Queiroz M., Jr C.H. 2003. A heuristic for the continuous capacity and flow assignment. European Journal of Operational Research. 146(3): 444-459.

Rai S.C., Misra B.B., Nayak A. K., Mall R. Pradhan S.K. 2010. A Multi-Objective QoS optimization with fuzzy based parameter setting for real time Multicasting. International Journal of Communications, Network and System Sciences. 3, 530-539.

Randaccio L.S., Atzori L. 2006. A genetic algorithm based approach for group multicast routing. Journal of Networks. 1(4): 1-9.

Reeves C.R. 1996. Modern heuristic techniques. In: Rayward-Smith V.J., Osman H., Reeves C.R., Smith G.D. Modern Heuristic Search Methods. Wiley, Chichester. pp. 1-25.

Reeves C.R. 1998. Landscapes, operators and heuristic search. Annals of Operations Research. 86, 473-490.

Reeves C.R. 2003. Genetic algorithms. In: Glover F., Kochenberger G. (Eds.). Handbook of Metaheuristics. pp.55-82. Kluwer Academic Publishers.

Resende M.G.C., Sousa J.P. 2004. Metaheuristics: Computer Decision Making. Kluwer Academic Publishers.

Resende M.G.C., Ribeiro C.C. 1997. A GRASP for graph planarization. Networks. 29, 173-189.

Resende M.G.C., Ribeiro C.C. 2005. GRASP with path-relinking: recent advances and applications. In Ibaraki T., Nonobe K., Yagiura M. (Eds.), Metaheuristics: Progress as Real Problem Solvers, pp. 29-63. Springer.

Ribeiro C.C., Souza M.C. 2000. Tabu search for the Steiner problem in graphs. Networks. 36(2): 138-146.

Ribeiro C.C., Uchoa E., Werneck R.F. 2002. A hybrid GRASP with perturbations for the Steiner problems in graphs. INFORMS Journal of Computing. 14(3): 228-246.

Ribeiro C.C.C., Martins S.L., Rosseti I.C.M. 2007. Metaheuristics for optimisation problems in computer communications. Computer Communications. 30(4): 656-669.

Roy A, Banerjee N, Das S.K. 2002. An efficient multi-objective QoS-routing algorithm for wireless multicasting. In: Proceedings of IEEE 55th Vehicular Technology Conference. pp. 1160-1164.

Roy A., Das S.K. 2004. QM2RP: a QoS-based mobile multicast routing protocol using multi-objective genetic algorithm. Wireless Networks. 10(3): 271-286.

Salama H.F., Reeves D.S, Viniotis Y. 1997. Evaluation of multicast routing algorithms for real-time communication on high-speed networks. IEEE Journal on Selected Areas in Communications. 15(3): 332-345.

Sarker R., Liang K.H., Newton C. 2002. A new multiobjective evolutionary algorithm. European Journal of Operational Research. 140(1): 12-23.

Schaffer J.D. 1985. Multiple objective optimisation with vector evaluated genetic algorithms. In: Proceedings of the International Conference on Genetic Algorithm and their Applications. pp. 93-100.

Serafini P. 1992. Simulated annealing for multiobjective optimisation problems. In: Proceedings of the Tenth International Conference on Multiple Criteria Decision Making. pp. 87-96.

Sha D.Y., Hsu C.-Y. 2006. A hybrid particle swarm optimisation for job shop scheduling problem. Computers and Industrial Engineering. 51(4): 791-808.

Shaikh A., Shin K.G. 1997. Destination-driven routing for low-cost multicast. IEEE Journal on Selected Areas in Communication. 15(3): 373-381.

Shen J., Xu F., Zheng P. 2005. A tabu search algorithm for the routing and capacity assignment problem in computer networks. Computers and Operations Research. 32(11): 2785-2800.

Skorin-Kapov J. 1990. Tabu search applied to the quadratic assignment problem. ORSA Journal on Computing. 2(1): 33-45.

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 Teleconmmunications. pp. 443-448.

Skorin-Kapov N., Kos M. 2006. A GRASP heuristic for the delay-constrained multicast routing problem. Telecommunication Systems. 32(1): 55-69.

Souza M.C., Duhamel C., Ribeiro C.C. 2003. GRASP with path-relinking heuristic for the capacitated minimum spanning tree problem using a memory-based local search strategy. In Resende M.G.C, Souza J.P. (Eds), Metaheuristics: Computer Decision Making. Kluwer Academic Publishers. pp. 627-658.

Srinivas N., Deb K. 1994. Multiobjective optimisation using nondominated sorting in genetic algorithms. Evolutionary Computation. 2(3): 221-248.

Sriram R., Manimaran G., Siva Ram Murthy C. 1999. A rearrangeable algorithm for the construction of delay-constrained dynamic multicast trees. In: Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM '99). pp. 1073-1080.

Stadler P.F. 1992. Correlation in landscapes of combinatorial optimisation problems. Europhysics Letters. 20(6):479-482.

Stadler P.F. 1995. Towards a Theory of Landscapes. In: Lopez-Pena, R., Capovilla R., Garcia-Pelayo R., Waelbroeck H., Zertuche F. (Eds.), Complex Systems and Binary Networks, Lecture Notes in Physics, 461. pp. 77-163. Springer.

Striegel A., Manimaran G. 2002. A survey of QoS multicasting issues. IEEE Communications Magazine. 40(6): 82-87.

Stützle T., Dorigo M. 1999. ACO algorithms for the travelling salesman problem. In Miettinen K., Makela M., Neittaanmaki P., Periaux J. (Eds.), Evolutionary Algorithms in Engineering and Computer Science. John Wiley and Sons.

Suman B, Kumar P. 2006. A survey of simulated annealing as a tool for single and multiobjective optimisation. Journal of Operational Research Society. 57(10): 1143-1160.

Sun J., Liu J., Xu W. 2006. QPSO-based QoS multicast routing algorithm. Lecture Notes in Computer Science, 4247. pp. 261-268. Springer.

Sun Q., Langendoerfer H. 1995. Efficient multicast routing for delay-sensitive applications. In: Proceedings of the Second Workshop on Protocols for Multimedia Systems. pp. 452-458.

Sun Q., Langendoerfer H. 1998. An efficient delay-constrained multicast routing algorithm. Journal of High-Speed Networks. 7(1): 43-55.

Sun Q., Langendoerfer H. 1999. A distributed delay-constrained dynamic multicast routing algorithm. Telecommunication Systems. 11, 47-58.

Takahashi H., Matsuyama A. 1980. An approximate solution for the Steiner tree problem in graphs. Mathematica Japonica. 24(6): 537-577.

Tasgetiren M.F., Liang Y.C., Sevkli M., Gencyilmaz G. 2007. A particle swarm optimisation algorithm for makespan and total flowtime minimisation in the permutation flow shop sequencing problem. European Journal of Operational Research. 177(3): 1930-1947.

Thompson J. M., Dowsland K.A. 1998. A robust simulated annealing based examination timetabling system. Computers and Operations Research. 25, 637-648.

Thaler D., Estrin D., Meyer D. 1998. Border gateway multicast protocol (BGMP): protocol Specification. RFC 2362.

Tran H.T., Harris R.J. 2003. Genetic algorithm approach to rearrangement of QoS multicast tree. In: Proceedings of the 3rd ATcrc Telecommunications & Networking Conference and Workshop.

Ulungu E.L, Teghem J. 1994. Multiobjective combinatorial optimisation problems: a survey. Journal of Multi-Criteria Decision Analysis. 3, 83-104.

Ulungu E.L., Teghem J., Fortemps P.H., Tuyttens D. 1999. MOSA method: A tool for solving multiobjective combinatorial optimisation problems. Journal of Multi-Criteria Decision Analysis. 8(4): 221-236.

Voß S. 1992. Steiner's Problem in Graphs: Heuristic Methods. Discrete Applied Mathematics. 40(1): 45-72.

Voß S., Martello S., Osman I. H. 1999. Meta-heuristics: advances and trends in local search paradigms for optimisation. Kluwer Academic Publishers.

Waitzman D., Partridge C., Deering S. 1988. Distance vector multicast routing protocol. RFC 1075.

Wang C.F., Liang C.T., Jan R.H. 2002. Heuristic algorithms for packing of multiple-group multicasting. Computers and Operations Research. 29(7): 905-924.

Wang H., Fang J., Wang H., Sun Y.M. 2004. TSDLMRA: an efficient multicast routing algorithm based on tabu search. Journal of Network and Computer Applications. 27, 77-90.

Wang J., Wang X., Huang M. 2005. A hybrid intelligent QoS multicast routing algorithm in NGI. In: Proceedings of the Sixth International Conference on Parallel and Distributed Computing Applications and Technologies. pp. 723-727.

Wang J.Q., Qin J., Kang L.S. 2006. A new QoS mutlicast routing model and its immune optimisation algorithm. In: Ma J., Jin H., Yang L.T., Tsai J.J.-P. (Eds.), Ubiquitous Intelligence and Computing, Lecture Notes in Computer Science, 4159. pp. 369-378.

Wang Q.B., Hou J. 2000. Multicast routing and its QoS extension: problems, algorithms, and protocols. IEEE Network. 14(1): 422-436.

Wang X.L., Jiang Z. 2004. QoS multicast routing based on simulated annealing algorithm. In: Proceedings SPIE on Network Architectures, Management, and Applications. pp. 511-516.

Wang Z., Crowcroft J. 1996. Quality of service routing for supporting multimedia applications. IEEE Journal on Selected Areas in Communications. 14(7): 1228-1234.

Wang Z., Shi B., Zhao E. 2001. Bandwidth-delay-constrained least-cost multicast routing based on heuristic genetic algorithm. Computer Communications. 24, 685-692.

Waxman B.M. 1988. Routing of multipoint connections. IEEE Journal on Selected Areas in Communications. 6(9): 1617-622.

Wei W., Feng Y., Tan J., Li Z. 2009. Product platform two-stage quality optimization design based on multiobjective genetic algorithm. Computers & Mathematics with Applications. 57(11-12): 1929-1937.

Weinberger E. 1990. Correlated and uncorrelated fitness landscapes and how to tell the difference. Biological Cybernetics. 63(5): 325-336.

Widyono R. 1994. The design and evaluation of routing algorithms for realtime channels. International Computer Science Institute, U.C. Berkeley. Technical Report, ICSI TR-94-024.

Wright S. 1932. The roles of the mutation, inbreeding, crossbreeding, and selection in evolution. In: Proceddings of the Sixth Congress on Genetics. 1, 355-366.

Xiao X., Ni L.M. 1999. Internet QoS: a big picture. IEEE Network. 13, 8-18.

Xu Y., Qu R. 2009. A GRASP approach for the delay-constrained multicast routing problem. In: Proceedings of the 4th Multidisplinary International Scheduling Conference (MISTA4). Dublin, Ireland.

Xu Y., Qu R. 2011. Solving multi-objective multicast routing problems by evolutionary multi-objective simulated annealing algorithms with variable neighbourhoods. Journal of the Operational Research Society. 62, 313-325.

Yang C.B., Wen U.P. 2005. Applying tabu search to backup path planning for multicast networks. Computers and Operations Research. 32(11): 2875-2889.

Yen G.G., Lu H. 2003. Dynamic multiobjective evolutionary algorithm: adaptive cell-based rank and density estimation. IEEE Transactions on Evolutionary Computation. 7(3):253-274.

Yeo C.K., Lee B.S., Er M.H. 2004. A survey of application level multicast techniques. Computer Communications. 27(15): 1547-1568.

Youssef H., Sait S.M., Adiche H. 2001. Evolutionary algorithms, simulated annealing and tabu search: a comparative study. Engineering Applications of Artificial Intelligence. 14(2): 167-181.

Youssef H., Al-Mulhem A., Sait S.M., Tahir M.A. 2002. QoS-driven multicast tree generation using tabu search, Computer Communications. 25, 1140-1149.

Zachariasen M. 1999. Local search for the Steiner tree problem in the Euclidean plane. European Journal of Operational Research. 119(2): 282-300.

Zahrani M.S., Loomes M.J., Malcolm J.A., Albrecht A.A. 2006. Landscape analysis for multicast routing. Computer & Communications. 30(1): 101-116.

Zahrani M.S., Loomes M.J., Malcolm J.A. 2007. LSA-based Landscape Analysis for Multicast Routing. In: Proceedings of the Twenty-Sixth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence (AI-2006). pp. 187-200. Springer.

Zahrani M.S., Loomes M.J., Malcolm J.A., Dayem Ullah A.Z.M., Steinhöfel K., Albrecht A.A. 2008. Genetic local search for multicast routing with pre-processing by logarithmic simulated annealing. Computers and Operations Research. 35(6): 2049-2070.

Zhang B., Krunz M.M., Chen C. 2001. A fast delay-constrained multicast routing algorithm. In: Proceedings of IEEE International Conference on Communications. 9, 2676-2680.

Zhang H., Li H., Tam C.M. 2006. Permutation-based particle swarm optimisation for resource-constrained project scheduling. Journal of Computing in Civil Engineering. 20(2): 141-149.

Zhang K., Wang H., Liu F.Y. 2005. Distributed multicast routing for delay variation-bounded Steiner tree using simulated annealing. Computer Communications. 28, 1356-1370.

Zhang L., Cai L., Li M., Wang F. 2009. A method for least-cost QoS multicast routing based on genetic simulated annealing algorithm. Computer Communications. 32(1): 105-110.

Zhang Q., Li H. 2007. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Trans. on Evolutionary Computation. 11(6): 712-731.

Zhong W.L., Huang J., Zhang J. 2008. A novel particle swarm optimisation for the Steiner tree problem in graphs. IEEE World Congress on Evolutionary Computation. pp. 2460-2467.

Zhu Q., Parsa M., Garcia-Luna-Aceves J. J. 1995. A source-based algorithm for delay-constrained minimum-cost multicasting, In: Proceedings of the Fourteenth Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM'95). pp. 377-385.

Zitzler E., Thiele L. 1999. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Transactions on Evolutionary Computation. 3(4): 257-271.