Available online at www.sciencedirect.com
ScienceDirect
Computers & Operations Research 35 (2008) 2049–2070
computers & operations research
www.elsevier.com/locate/cor

Genetic local search for multicast routing with pre-processing by logarithmic simulated annealing

M.S. Zahrani[a], M.J. Loomes[b], J.A. Malcolm[a], A.Z.M. Dayem Ullah[c], K. Steinhöfel[d], A.A. Albrecht[a,*]

[a] *University of Hertfordshire, School of Computer Science, Hatfield, Herts AL10 9AB, UK*
[b] *Middlesex University, School of Computing Science, Hendon, London NW4 4BT, UK*
[c] *Imperial College London, Department of Computing, 180 Queen's Gate, London SW7 2AZ, UK*
[d] *King's College London, Department of Computer Science, Strand, London WC2R 2LS, UK*

Available online 14 November 2006

Abstract

Over the past few years, several local search algorithms have been proposed for various problems related to multicast routing in the off-line mode. We describe a population-based search algorithm for cost minimisation of multicast routing. The algorithm utilises the partially mixed crossover operation (PMX) under the elitist model: for each element of the current population, the local search is based upon the results of a landscape analysis that is executed only once in a pre-processing step; the best solution found so far is always part of the population. The aim of the landscape analysis is to estimate the depth of the deepest local minima in the landscape generated by the routing tasks and the objective function. The analysis employs simulated annealing with a logarithmic cooling schedule (logarithmic simulated annealing—LSA). The local search then performs alternating sequences of descending and ascending steps for each individual of the population, where the length of a sequence with uniform direction is controlled by the estimated value of the maximum depth of local minima. We present results from computational experiments on three different routing tasks, and we provide experimental evidence that our genetic local search procedure that combines LSA and PMX performs better than algorithms using either LSA or PMX only.

Keywords: Multicast routing; Genetic local search; Simulated annealing; Steiner trees; Quality of service (QoS)

1. Introduction

Multicast routing has become an important topic in combinatorial optimisation. A recent overview on multicast routing and associated optimisation algorithms has been presented by Oliveira and Pardalos [1]. The focus of this overview, as in most papers on multicast routing, are on-line algorithms [2,3]. An early summary of problems and technical solutions related to multicast communication was given by Diot et al. [4]. Great effort has been undertaken to incorporate quality of service (QoS) into data communication networks such as ATM and IP networks [5–11]. Many multicast applications, such as video conferencing, distance-learning, and multimedia

* Corresponding author. Tel.: +44 1707 284 247; fax: +44 1707 284 303.
E-mail address: A.Albrecht@herts.ac.uk (A.A. Albrecht).

0305-0548/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2006.10.001

broadcasting are QoS-sensitive in nature and thus they should benefit from the QoS improvement in the underlying networks.

Designing multicast routing algorithms is a complex and challenging task. Among the various issues involved are the design of optimal routes taking into consideration different cost functions, the minimisation of network load and the avoidance of loops and traffic congestion, and the provision of basic support for reliable transmission. In the present paper, we focus on the design of optimal routes in the off-line mode, as discussed, e.g., in the survey [9] (see Section 2 therein) and [12,13].

The problem of minimising the tree costs of single requests under the constraint that all path capacities are within a user-specified capacity bound, i.e. the requests are executed simultaneously, is referred to as the capacity constrained multicast routing problem (CCMRP) [1,4,14,15].

The CCMRP can be formalised as a constrained Steiner tree problem, which is known to be NP-complete [16]. We note that in applications like video conferencing, multimedia broadcasting, and distance-learning the routing procedure is updated only from time to time, e.g. when new customers register to use one of the services. In such cases, off-line routing algorithms are an appropriate way to solve the routing problem. Since we are dealing with an NP-complete problem, local search methods are a natural choice to tackle the problem; see [9,12,13].

In [17–24], algorithms utilising genetic algorithms (GA) or tabu search are presented. For an overview of search methods, in particular, GA applied to various problem settings in multicast routing, we refer the reader to [25, cf. p. 20–21 therein]. Here, we discuss only a few of the issues raised on this topic. We note that most of the papers are dealing with single trees, but not with routing multiple requests (trees) simultaneously.

The GA proposed in [17,18] assume that several messages all have to be transferred from several sources to multiple destinations, and this has to be executed simultaneously without any order or priority for certain messages. The GA uses a population of chromosomes, where each chromosome is a permutation of the numbers that are assigned to the requests. The algorithms start with a subset of k out of n requests. By using a Steiner tree algorithm, the k requests are routed in the order they appear in the chromosome (partial permutation). Then, to pairs of chromosomes the partially mixed crossover (PMX) operation and the new population (of the same fixed size) is generated by roulette wheel selection, where a sector of a "roulette wheel" is assigned to each offspring whose size is proportional to the fitness measure. The algorithm runs a fixed number of steps, and then k is increased by one in order to check whether $(k+1)$ requests can be scheduled conflict-free. The same procedure is repeated for $(k+1)$ until either $k=n$, or repeated attempts to schedule simultaneously k requests are unsuccessful. The search-based methods from [17,18] are, in part, incorporated into our approach and are discussed in more detail in Section 4.1.

The paper by Ericsson et al. [19] demonstrates a variety of routing problems that can be tackled by GA. The authors apply GA to a routing problem where the link weights are assigned by the network operator, i.e. the problem setting is somewhat different from ours. Then the weight setting problem seeks a set of weights that optimises network performance. Given a set of projected demands, the weight assignment problem, with the objective of minimising network congestion, is NP-hard. The individuals of the population are weight vectors, where the range of components is from 1 to $2^{16}-1$. The crossover operator acts on one elite and one non-elite parent and selects each component of the resulting weight vector according to independently chosen random numbers from (0, 1). The evaluation of the fitness function is rather complicated, since it involves the whole process of routing and the computation of arc loads. The method was successfully tested on the AT&T Worldnet backbone with projected demands, and on several synthetic networks.

Barolli et al. [20] focus on creating a robust path finding solution for mobile ad hoc networks (MANTETs). Since the nodes are mobile, the creation of routing paths is affected by the addition and deletion of nodes, i.e. the topology of the network may change rapidly and unexpectedly. Therefore, QoS is only guaranteed as long as a signal to the node actually exists. The authors propose a genetic algorithm for mobile ad hoc networks (GAMAN) where the network and, respectively, the individuals of the population are represented by trees. The GAMAN algorithm uses the single point crossover and a mutation operation where the "tree junctions" are chosen randomly in the range from zero up to $1/\ell$, for $\ell=$ length of individuals. The algorithm employs the elitist model, where the individual with the highest fitness value in a population is left unchanged in the next generation. The simulation results show that the algorithm is reasonably fast on small to medium size networks.

Yang [21] devised a tabu search algorithm for finding a single, feasible multicast tree efficiently that satisfies a number of QoS constraints. The method is tested on randomly generated networks with 100 nodes (and on 8×8

meshes). A similar setting (Steiner tree computation under certain constraints by tabu search) has been investigated by Skorin-Kapov and Kos [22]. The tests on a large number of benchmark problems have shown that the tabu search heuristic from [22] is superior in quality for medium sized problems.

Wang et al. [23] discuss the same problem as in [21,22]. The authors propose an efficient algorithm based on tabu search for delay constrained, low cost multicast trees (TSDLMRA). To evaluate the efficiency of TSDLMRA, the authors utilise a random link generator, which yields networks with an average node degree of 4–6. The link delay function is defined as the propagation delay of the link. The TSDLMRA algorithm is shown to be of low time complexity, with the ability to find multicast trees if such solutions exist.

Yang and Wen [12] apply tabu search to the problem of pre-planning delay-constrained backup paths for multicast trees to minimise the total cost of all the backup paths. The neighbourhood structure of the search algorithm is based upon the random selection of a single link in the current solution for backup paths. The computational experiments were carried out on networks with 30–50 nodes.

Apart from GA and tabu search, simulated annealing-based search [26–30] has been utilised recently for multicast routing, in particular, under QoS considerations [13,31–35]. In [31], the QoS issue is reduced to a path constraint problem (multiple requests are not considered), where along a path from source to destination each link has to obey a vector of weight restrictions. The constraint vector is transformed into an energy function by a max-operation over the component-wise ratio of link weights and capacity constraints. The search for appropriate paths is then executed by simulated annealing. The paper [32] demonstrates how different QoS requirements, like available CPU resources, buffer resources, error rates, queuing delay and sending delay at each node as well as available bandwidth, transmission delay and error rate at each link, can be incorporated into a single energy function for a given potential multicast routing solution (see Section 3.2 for more details). Simulated annealing is then applied to this energy function (the experiments are executed on small networks), where the underlying model is a homogeneous Markov chain (cf. Section 3.1). Since the specific QoS requirements considered in [32] do not affect the general methodology, we have chosen only two QoS parameters for the calculation of the energy function (cf. Sections 2 and 3.2). In [13] (see also [9], Section 2), an overlay multicast network infrastructure is proposed which forms a multicast data delivery backbone. The overlay topology is continuously adapted (off-line) with changes in the distribution of the clients as well as changes in network conditions. The performance optimisation is executed by a simulated annealing-based algorithm defined by homogeneous Markov chains. The paper [35] employs a QoS setup similar to [32] and investigates three different search methods to calculate routing trees: simulated annealing, tabu search, and GA. The three methods are tested on small (14 nodes, 21 edges) and relatively large (100 nodes, 800 edges; randomly generated) networks. The findings suggest that simulated annealing can solve multicastrouting problems efficiently with high-quality solutions, tabu search-based algorithms show a good time performance when the group size is large, and that GA genetic-based methods slightly outperform SA in terms of the solution quality.

In the present paper, we introduce an new search method that combines landscape analysis with a genetic local search procedure. The notion of landscape analysis was first mentioned in [36] and has become a major topic in combinatorial optimisation in recent years [37–39]. Our tool for landscape analysis is logarithmic simulated annealing (LSA) [28,29,40], i.e. in our approach we employ simulated annealing based on inhomogeneous Markov chains. The annealing procedure allows us to estimate the depth of the deepest local minima. Recently, simulated annealing algorithms, in particular variants based on inhomogeneous Markov chains, have been used to investigate problems from Computational Biology [41,42]. Another motivation for choosing simulated annealing is based upon recent advances in GA research [43], where the convergence of GA to optimum solutions has been ensured by employing simulated annealing-based selection in a variety of ways (see Section 10 in [43] for a summary of results). Since we focus on algorithmic aspects of multicast routing in off-line mode, we reduce the QoS requirements to bandwidth and delay constraints (see Section 3.2). The present paper is an extension of the short conference presentation [44]. The competitiveness of LSA in relation to GA and tabu search was demonstrated in [45] for the job shop scheduling problem, which is one of the hardest NP-complete problems.

We performed computational experiments on three instances of the OR library [46] (steinb10, steinb11, steinb18), which were modified for multicast routing. The results provide evidence that our genetic local search heuristic performs better than “pure” LSA.

The paper is structured as follows: in Section 2, we provide a formal definition of the multicast routing problem, along with explanations about parameters and cost functions. In Section 3, we describe LSA pre-processing as a tool for landscape analysis. In Section 4, we introduce our genetic local search heuristic, and in Section 5 we present the

results from computational experiments on the modified instances no. steinb10, steinb11, and steinb18, including the results from the landscape analysis which is performed in a pre-processing step.

2. Formal definition of multicast routing

Communication networks consist of nodes connected through links. The nodes are the originators and receivers of information, while the links serve as the transport between nodes. Nodes can be either endpoint nodes or intermediary nodes. Both nodes and links have a limited capacity of information flow they can handle, depending on features such as speed of information flow and the cost of transferring the information at the required speed.

Given a graph $G = (V, E)$ that represents a communication network with node set V and edges E, we define two non-negative weight functions $Co : E \to \Re$ and $Ca : E \to \Re$, where Co is the cost function and Ca is the capacity function on E, respectively.

Each of the point-to-multipoint requests has a source node $s \in V$ and a set of destination nodes $D \subseteq V$. We define a multicast request R by setting

$$R = [v_s \Rightarrow (v_1, v_2, \ldots, v_n); C],$$

where

$$v_s = \text{the source node of } R,$$

$$D = \{v_1, \ldots, v_n\} = \text{the destination nodes},$$

$$C = \text{the capacity required by each of the transmissions } v_s \Rightarrow v_i. \tag{1}$$

The multicast problem P is then defined by

$$\mathsf{P} = [G; Co; Ca; R_1, \ldots, R_n]. \tag{2}$$

Usually, multicast routing algorithms are based on the following assumptions [1,4,17,18]:: each R from P is routed separately by a minimum Steiner tree with root v_s and destination nodes $D(R)$. The cost of the Steiner tree is the sum $\sum_e Co(e)$ of the costs of the edges in the tree, while the capacity $Ca(e)$ on each edge on a path from v_s to each $v_i \in D$ obeys $Ca(e) \geqslant C(R)$ for the capacity C of R, see (1).

The problem of multicast routing is a complex one. The naïve approach to solving the point-to-multipoint routing problem is through the separation of the problem into several point-to-point routing problems, according to the number of destination points. This simple method is very inefficient. The same information might flow on the same link many times. It creates unnecessary traffic on the link, which could be avoided. Therefore, we consider the simultaneous execution of requests, where messages from a single source are combined to form a single request.

To minimise $\sum_e Co(e)$ for given G, v_s, D, and Ca is NP-complete [16]. Numerous heuristics have been devised to find good approximations of minimum solutions efficiently [21,22,47–52]. Since in our approach single requests may have to be rerouted many times in order to satisfy capacity constraints, we employ the simple but efficient KMB algorithm [48]. It has been estimated that the cost of a tree generated by the KMB algorithm averages 5% more than the cost of a Steiner minimal tree [49]. Koch and Martin [51] obtained minimum values for a large number of benchmark problems by using a polyhedral method. The algorithm proposed in [52] implies that each request can be routed in polynomial time within $\frac{5}{3}$ of the minimum cost of a Steiner tree. Of course, the KMB algorithm we are using can be substituted by any efficient Steiner tree algorithm.

In [32,35], various QoS constraints are taken into account in order to define the objective function: every node $v \in V$ is labelled by the following parameters: available CPU resources $c(v)$, available buffer resources $b(v)$, queuing delay $t(v)$, and sending delay $\tau(v)$. Furthermore, every link $e = (v, u)$ is labelled by: available bandwidth $w(e)$ and transmission delay $\delta(e)$. The QoS constraints are then given by

$$\forall v(c(v) \geqslant C; \quad b(v) \geqslant B; \quad w(e) \geqslant W) \tag{3}$$

for some constants C, B and W, where additionally the source-destination delay, calculated from $t(v)$, $\tau(v)$ and the sum of $\delta(e)$, is bounded by Δ. The task is to minimise a function $(\sum_v h(v) + \sum_e H(e) + Q)$, where h and H are

heuristic cost functions and Q is the weighted number of source-destination pairs that violate the delay bound; h and H are defined by

$$h(v) := \frac{\kappa_1}{c(v)} + \frac{\kappa_2}{b(v)}, \quad H(e) := \frac{\kappa_3}{w(e)} \tag{4}$$

for some constants κ_1, κ_2 and κ_3. Thus, the QoS constraints are incorporated into a single additive function. Since we are interested in algorithmic aspects, i.e. the presentation of the landscape approach and the new genetic search procedure, we consider a simplified version of the objective function. Moreover, in contrast to, e.g., [32,35] we consider the simultaneous routing of several requests (ranging from 9 to 20 in our experiments, see Section 5). Our cost function *Co* can represent, e.g., the delay along a single link, and the capacity *Ca* can be related to the bandwidth. In our setting, feasible solutions have to obey the bandwidth constraints, and adding $c(v)$, $b(v)$, $t(v)$ and $\tau(v)$ to the selection of feasible solution would not change the overall approach.

Our objective function is basically chosen as in [17,18] and represents a combined measure of transmission costs and capacity constraints: let $T(R)$ denote the set of edges of the tree associated with the request R from configuration $S \in \mathsf{M}$. We first define

$$W(R) := C \cdot \sum_{e \in T(R)} Co(e), \tag{5}$$

where *Co* is from (2) and C the capacity request of R; see (1). The value of the objective function $Z(S)$, $S \in \mathsf{M}$, is then simply given by

$$Z(S) := \sum_{R \text{ from } S} W(R). \tag{6}$$

3. LSA pre-processing

Simulated annealing was introduced as an optimisation tool independently in [26,27]; see also [30]. The underlying algorithm acts within a configuration space in accordance with a specific neighbourhood structure, where the transition steps are controlled by the objective function.

3.1. Simulated annealing in the multicast routing context

The configuration space consists of all feasible solutions for a given multicast problem $\mathsf{P} = [G; Co; Ca; R_1, \ldots, R_n]$, i.e. the capacity conditions according to (1) are not violated. We denote the configuration space by

$$\mathsf{M} = \{S \mid S = [R_{i_1}, \ldots, R_{i_n}];\ R_{i_1}, \ldots, R_{i_n} \text{ are SMT-routed}\}. \tag{7}$$

Here, *SMT-routed* means that each R_{i_j} from the ordered sequence S is routed by the KMB algorithm [48] that approximates a Steiner minimal tree (SMT) in accordance with the capacity constraints and the given cost function.

By N_S we denote the neighbourhood of S, and $Z(S)$ denotes the value of underlying objective function; both are specified in Section 3.2. The neighbours N_S are all required to be feasible, and S itself is an element of N_S.

The probability of performing a transition from S to $S' \in N_S$ is defined by

$$\mathbf{P}r\{S \to S'\} = \begin{cases} G[S, S'] \cdot A[S, S'] & \text{if } S' \neq S; \\ 1 - \sum_{H \neq S} G[S, H] \cdot A[S, H] & \text{otherwise,} \end{cases} \tag{8}$$

where G denotes the probability of generating a specific neighbour from N_S, and A is the probability of accepting the neighbour once it has been generated according to G. The generation probability is uniform and defined by

$$G[S, S'] := \begin{cases} \dfrac{1}{|N_S|} & \text{if } S' \in N_S, \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

The "uniform" definition assumes that all potential neighbours are tested if they are feasible or not in order to determine N_S and $|N_S|$, which would require a large number of SMT calculations. In the actual implementation (see Section 5.1),

a potential neighbour is chosen randomly and its feasibility is tested. If conflicts arise in simultaneous routing, a new potential neighbour is chosen; see Section 3.2.

The acceptance probabilities $A[S,\ S']$, $S' \in N_S$, are derived from the underlying analogy to thermodynamic systems:

$$A[S, S'] := \begin{cases} 1 & \text{if } Z(S') - Z(S) \leqslant 0, \\ \mathrm{e}^{-(Z(S')-Z(S))/c} & \text{otherwise,} \end{cases} \tag{10}$$

where c is a control parameter having the interpretation of a *temperature* in annealing procedures. The actual decision, whether or not S' should be accepted in case of $Z(S') > Z(S)$ is performed in the following way: S' is accepted, if

$$\mathrm{e}^{-(Z(S')-Z(S))/c} \geqslant \rho, \tag{11}$$

where $\rho \in [0, 1]$ is produced by a random number generator. The value ρ is generated in each trial where $Z(S') > Z(S)$.

Let $\mathbf{a}_S(k)$ denote the probability of being in configuration $S \in \mathsf{M}$ after k steps according to (8),…,(11). The probability $\mathbf{a}_S(k)$ is given by

$$\mathbf{a}_S(k) := \sum_H \mathbf{a}_H(k-1) \cdot \mathbf{P}r\{H \to S\}, \tag{12}$$

where $\mathbf{P}r\{H \to S\}$ is from (8). The recursive application of (12) defines a Markov chain of probabilities $\mathbf{a}_S(k)$, where $S \in \mathsf{M}$ and $k = 1, 2, \ldots$. If the parameter $c = c(k)$ in (10) is a constant c, the chain is said to be a *homogeneous* Markov chain; otherwise, if $c(k)$ is lowered at each step, the sequence of probability vectors $\vec{\mathbf{a}}(k)$ is an *inhomogeneous* Markov chain.

In contrast to [13,31–35], we consider inhomogeneous Markov in our search procedure. The motivation for this choice is based upon the convergence properties of the two types of Markov chains: convergence propositions about homogeneous Markov chains rely on an infinite number of transitions at fixed "temperatures" c. The probability distribution approached in the limit is the Boltzmann distribution $\mathrm{e}^{-Z(S)/c}/V$, where V is a normalisation value. If $c \to 0$, the Boltzmann distribution tends to the distribution over optimum configurations. In practice, however, it is infeasible to perform an infinite number of transitions at fixed temperatures. The convergence analysis of inhomogeneous Markov chains avoids the intermediate step, and in our approach the "temperature" $c(k)$ changes in accordance with

$$c(k) = \frac{\Gamma}{\ln(k+2)}, \quad k = 0, 1, \ldots . \tag{13}$$

The choice of $c(k)$ is motivated by Hajek's theorem [28] on logarithmic cooling schedules. We denote by $F_{\min}$ the set of optimum solutions. Basically, Hajek's theorem states

Theorem 1.

Under some natural assumptions about the configuration space F *and the neighbourhood* N_f*, the asymptotic convergence* $\sum_{f \in \mathsf{F}_{\min}} \mathbf{a}_f(k) \underset{k \to \infty}{\to} 1$ *of LSA is guaranteed if and only if* Γ *from* (13) *is lower bounded by the maximum value of the minimum escape height from local minima.*

Albrecht [40] proved that after $(n/\delta)^{O(\Gamma)}$ neighbourhood transitions, the probability to be in an optimum solution is at least $1 - \delta$, where n is an upper bound for the size of the neighbourhoods.

Unfortunately, due to the complex nature of our configuration space M, we cannot decide whether or not Theorem 1 applies. However, logarithmic simulated annealing has proved to be an efficient method in similar settings; cf. [45]. For an illustration of Γ for a particular local minimum A, see Fig. 1.

3.2. Neighbourhood relation

There are numerous ways to define the neighbourhood relation; in the present study, we focus on one example only. Given $S \in \mathsf{M}$ by $S = [R_{i_1}, \ldots, R_{i_n}]$, the neighbourhood N_S includes S itself and is defined by the following procedure:

> Two integers a and b, where $1 \leqslant a < b \leqslant n$, are randomly chosen, and the order of all requests from number i_a to number i_b is reversed. Thus, a new potential configuration S' is generated (see Fig. 2).

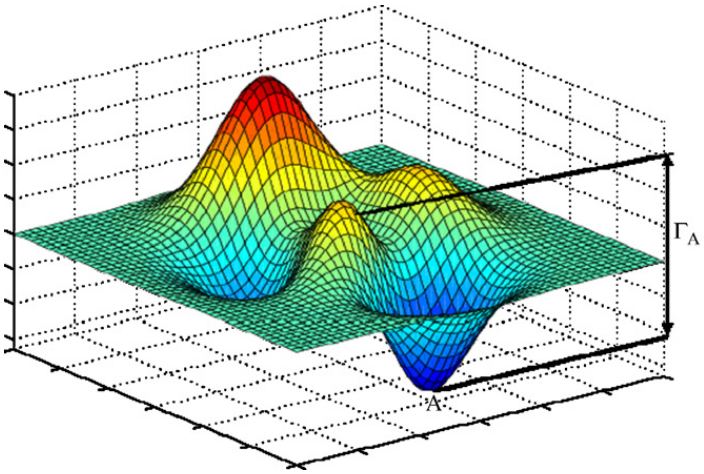

Fig. 1. Γ example.

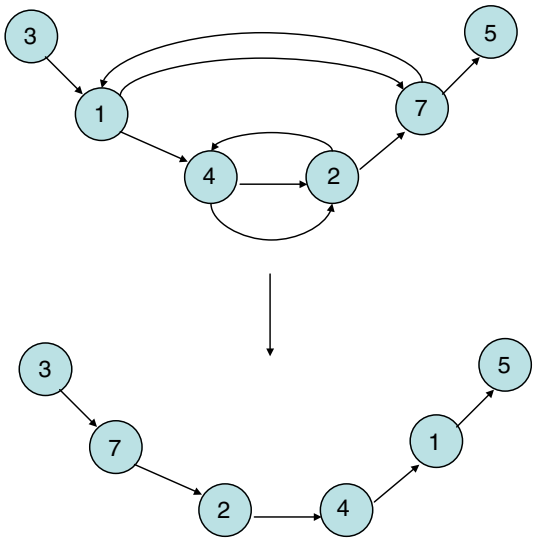

Fig. 2. A neighbourhood transition example; here $a = 2$, $b = 5$.

The potential configuration S' is validated for feasibility, i.e. we try to simultaneously schedule all the requests from R_{i_b} upwards. If a conflict occurs, a new pair (a, b) is generated.

If $S' \in \mathsf{M}$, the value of the objective function $Z(S)$ is calculated.

3.3. Initial feasible solutions

We randomly select an order $(i_1, i_2, \ldots, i_n)$ of $i_j \in [1, 2, \ldots, n]$. If the request R_{i_j} has been scheduled successfully by the KMB algorithm [48], $j \geqslant 1$, then the capacity function Ca is updated by

$$\forall e(e \in E \rightarrow Ca(e) := Ca(e) - C), \tag{14}$$

where $C = C(R_{i_j})$ is the capacity request of R_{i_j}. We then try to schedule request $R_{i_{j+1}}$ by the KMB algorithm. Before KMB is applied, all edges $E \in G$ with updated values $Ca(e) < C(R_{i_{j+1}})$ are removed from G, i.e. the underlying network is modified in accordance with $[R_{i_1}, \ldots, R_{i_j}]$. If capacity constraints are violated, i.e. $R_{i_{j+1}}$ cannot be scheduled by KMB, a new random order $(i'_1, i'_2, \ldots, i'_n)$ is generated, and we start again with R_{i_1}. After 100 unsuccessful attempts, the search for a feasible solution of $\mathsf{P} = [G; Co; Ca; R_1, \ldots, R_n]$ is terminated.

3.4. Landscape analysis

Merz and Freisleben [37] present different methods of landscape analysis as part of genetic local search methods (memetic-style algorithms). We introduce a new method that basically estimates Γ in (13), i.e. provides an upper bound for the maximum escape height from local minima.

To estimate Γ, we first have to decide about the number of transitions T after starting with a solution S_0 generated by the procedure described in Section 3.3. Based upon previous work on LSA [45] for one of the hardest NP-complete problems, namely job shop scheduling, we decided to choose T in the region of $T \approx 10^4, \ldots, 2 \times 10^4$, which, actually, has been confirmed as an appropriate choice by the computational experiments; see Section 5.1. The basic algorithm is summarised in Algorithm 3.1.

For a pre-defined number of transitions T, we employ the following procedure in order to find an estimation of Γ: first, the procedure tries to estimate the intermediate increase G_{est} of the objective function between two successive improvements of the best value $Z(S)$ found so far. Then, we establish a conjecture about Γ_{est} that is based on G_{est}, and subsequent computational experiments are executed for different settings of Γ in (13); see Section 5.1. To find at first a value for G_{est}, we proceed as follows:

Two initial solutions S_0^1 and S_0^2 are generated by the procedure from Section 3.3, and $G_{\text{est}} := |Z(S_0^1) - Z(S_0^2)|$ is the initial estimation, where we assume $Z(S_0^1) \neq Z(S_0^2)$.

We set $S_{\text{best}} := S_0^i$, where $Z(S_0^i)$ is the smaller value out of $\{Z(S_0^1), Z(S_0^2)\}$, $i \in \{1, 2\}$. The procedure from Section 3.1 is started with S_{best} and G_{est} in (13); an auxiliary parameter $\Delta_0 := 0$ is initialised.

At each step $k \leqslant T$, if $Z(S_k) > Z(S_{k-1})$ and $Z(S_{\text{best}}) + \Delta_s < Z(S_k)$, we update Δ_s by $\Delta_s := Z(S_k) - Z(S_{\text{best}})$, $0 \leqslant s \leqslant k$.

At each step $k \leqslant T$, if $Z(S_k) < Z(S_{k-1})$, $Z(S_k)$ is compared to $Z(S_{\text{best}})$: if $Z(S_k) < Z(S_{\text{best}})$, then we set $S_{\text{best}} := S_k$, we update $G_{\text{est}} := \max\{\Delta_s, G_{\text{est}}\}$, and we initialise again $\Delta_{s+1} := 0$.

After finishing step $k = T$, we set $G_{\text{est}} := \max\{\Delta_{s(T)}, G_{\text{est}}\}$, where $\Delta_{s(T)}$ is the latest update of Δ.

Thus, every time the value $Z(S_{\text{best}})$ is updated, i.e. when a potential local minimum has been reached, a new estimation of G_{est} is started. If the initial value $G_{\text{est}} := |Z(S_0^1) - Z(S_0^2)|$ appears to be too small, the initial estimation can be chosen in the region of $c \cdot |Z(S_0^1) - Z(S_0^2)|$, where $c \approx 3, \ldots, 5$; cf. [45].

Algorithm 3.1. The program description

```
1:  Read all the network information; choose Γ.
2:  Determine the initial temperature of the simulated annealing process by c(0) = Γ/ln(2).
3:  Determine an initial feasible solution; see Section 3.3.
4:  repeat
5:      Begin the process of simulated annealing at temperature c(k) = Γ/ln(k + 2).
6:      Generate a neighbourhood solution S′ from the current solution S; see Section 3.2.
7:      Determine the Steiner trees using KMB Algorithm for the requests of S′.
8:      if S′ is feasible then
9:          Determine Z(S′); see (5) and (6);
            else increment k; goto 6.
10:     end if
11:     Z(S′) < Z(S) then
12:         the new configuration is accepted and we move to S′;
            else
13:         if Z(S′) > Z(S) then
14:             generate uniformly ρ ∈ [0, 1];
15:             if e^{−(Z(S′)−Z(S))/c(k)} ⩾ ρ then
```

16: S' is accepted as the new configuration;
else increment k; **goto** 6.
17: **end if**
18: **end if**
19: **end if**
20: increment k.
21: **until** $k = T_{\text{bound}}$.
We exit the program and return $Z(S_{\text{best}})$.

We recall that Γ itself is related to the maximum value of the minimum escape height from local minima. It is unlikely that G_{est} is close to the minimum escape height, and therefore further experiments are required to establish a relationship between Γ and G_{est}. On the other hand, G_{est} provides some information about the structure of the underlying landscape. Thus, G_{est} together with different settings for Γ in (13) are used to find a good estimation Γ_{est}.

As we will see in Section 5.1, G_{est} and settings for Γ in (13) that resulted in the best values of the objective function obtained by LSA on all routing tasks defined for the three networks, differ by a factor close to 10, and this relation is relatively independent of the particular network structure G in (2), i.e. the relation is basically determined by the functions Co and Ca in (2).

4. Genetic local search for multicast routing

GA are based on nature's selection process and the concept of survival of the fittest [53,54]. GAs utilise random mutation, crossover and selection procedures to create better solutions from a random starting population. The population contains several initial solutions. Each solution is evaluated and its fitness is calculated. Then a new generation is created from the current population by crossover and mutation, where usually the size of the population is kept unchanged by applying the fitness function. Based on a convergence result by Rudolph [55], the best solution found so far is always maintained in the population, i.e. we follow the so-called elitist approach.

4.1. The partially mixed crossover (PMX) operation

Since we rely on similar notations and basically the same configuration space as in [17,18], we explain in more detail their GA-based method. The key element of the method is the PMX operation. The algorithm starts with a randomly chosen population. The PMX is applied to pairs of individuals: two strings are aligned and two crossing sites are picked uniformly at random along the strings; see Fig. 3. The positions P_1 and P_2 define a matching section where the requests are exchanged position by position. This operation may generate duplicate occurrences of requests. Therefore, the following procedure is applied to the rest of the positions (outside the matching section): if R has a duplicate R' in the same string (like 3 in offspring-1 in position 5 and 8), then the duplicate R' outside the matching region is substituted by the request R'' from the other offspring (offspring-2) that was swapped with R.

Finally, the new population of the same fixed size is then generated by roulette wheel selection, where a sector of a "roulette wheel" is assigned to each offspring whose size is proportional to the fitness measure, and then a random position is chosen on the wheel.

Zhu et al. [17] employ this operation in the following procedure: starting from a random initial population, where each individual has only $k < n$ requests from the same k-subset of n requests, the algorithm executes a fixed number of PMX operations. Then k is increased by one in order to check whether $(k + 1)$ requests can be scheduled conflict-free, and the same fixed number of PMX operations is applied, until either $k = n$ or repeated attempts to schedule simultaneously $k + x < n$ requests are unsuccessful. The heuristic was evaluated for a population size is 100 and 100 crossover/selection steps for each $k \leqslant n = 20$. The $n = 20$ requests were defined in a network with 61 nodes and 133 edges. Each request R has at least eight destination nodes, and the capacity C was between five and nine; see [17]. Feasible solutions were found for $k \leqslant 18$. Since the network information is not provided in detail by Zhu et al. [17], we are unfortunately unable to apply our approach.

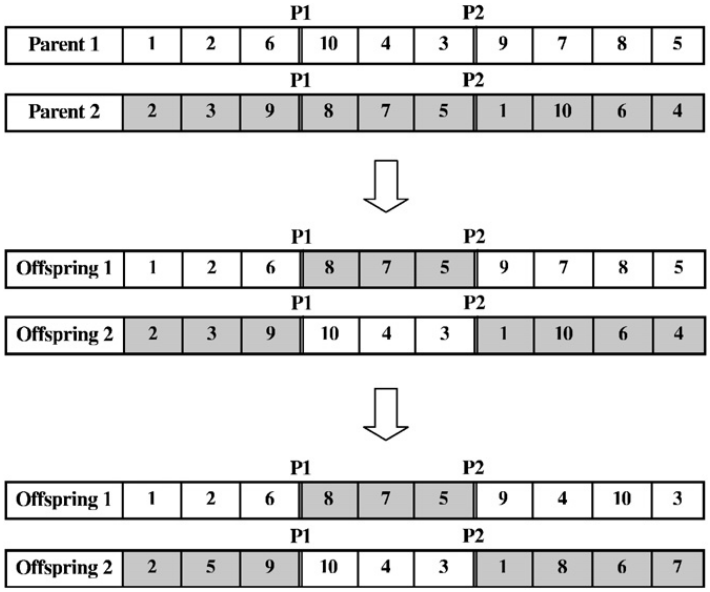

Fig. 3. Two individuals before and after the PMX from [56].

4.2. Genetic local search

Over the past few years, genetic local search has been investigated in the context of a variety of combinatorial optimisation problems; cf. [18,37] and the literature therein. The basic idea is relatively simple: a (quasi-)deterministic local search with continuous improvements of the objective function is executed for all individuals of a population; if the individual runs are stuck in local minima, a crossover operation is applied in order to leave local minima. Here, quasi-deterministic means that the "downward" steps may have a random component, i.e. the neighbours with improved values of the objective function might be chosen randomly. In our heuristic, we employ such a "modest random" procedure.

The parameters of our genetic local search procedure are:

(1) A multicast routing problem P as defined in Eq. (2), i.e. with n requests as defined in Eq. (1);
(2) Γ_{est} as defined in Section 3.4 and estimated in Eq. (15);
(3) The population size M;
(4) The number K of maximum steps between two successive executions of the PMX operations;
(5) The number N of maximum executions of the PMX operation applied to a single element of the population.

Algorithm 4.2. The Genetic local search description

Read all the network information; choose Γ.
Determine an initial feasible solution; see Section 3.3.
repeat
  Run M computations for a predefined number of steps K.
  Generate a neighbourhood solution S' from the current solution S; see Section 3.2.
  Determine the Steiner trees using KMB Algorithm for the requests of S'.
  if S' is feasible **then**
    D etermine $Z(S')$; see (5) and (6);
    else goto 5.
  end if
  (First mode) Do only downwards steps.
  if $Z(S') < Z(S)$ **then**

12: the new configuration is accepted and we move to S';
13: **if** local minimum is reached **then**
14: **goto** 23.
else goto 10.
15: **end if**
16: **end if**
end if
17: **if** $Z(S') > Z(S)$ **then**
18: we ignore it and try a new neighbour.
19: **if** NO S' with $Z(S') < Z(S)$ for L tries found **then**
20: treat S as a local minimum. **goto** 23.
21: **end if**
22: **end if**
23: (Second mode) Do only upwards steps.
24: **if** Z (*Local minimum*) $+ \Gamma$ is reached **then**
25: switch again to the first mode. **goto** 10.
else goto 23.
26: **end if**
27: **until** K steps done for all M computations
28: **for** each pair of intermediate solutions **do**
29: apply partially mixed crossover (PMX);
30: **end for**
31 Restart again K steps with the M best solutions from PMX crossover, including elitist solution.
32: All this is repeated N times (thus, for a single strain of computations we get $K * N$ steps);
33: After N repeated K steps, we take the best solution out of the M results.

Due to the extremely large number of potential neighbours, we do need two more auxiliary parameters: if after $L = 50$ unsuccessful trials no neighbour with a better value of the objective function could be found, the current solution with objective value $Z(S)$ is declared to be a potential local minimum, and the procedure switches from downward steps to a sequence of upward steps. The upward steps are executed until either an S' with $Z(S') \geqslant Z(S) + \Gamma_{\text{est}}$ has been reached, or after $L = 50$ unsuccessful trials no neighbour with a larger value of the objective function could be found. In either of the two cases, the procedure switches back to downward steps. Thus, for each individual a random walk through the landscape is executed, and after K steps, the walk is interrupted by an PMX operation in order to generate a new population of the same size in the elitist model. The heuristic is described in Algorithm 4.2.

5. Computational experiments

The algorithms described in Sections 3.4 and 4.2 have been implemented in Java. Particular attention has been paid to the implementation of the KMB algorithm, which uses Dijkstra's shortest path algorithm and Kruskal's minimal spanning tree algorithm. The experiments were executed on a 2 GHz Pentium4 Processor with 512 MB RAM. The population-based computations were simulated by subsequent sequential runs, which caused restrictions on the population size.

The underlying graphs are the instances steinb10 (75 nodes, 150 edges), steinb11 (same number of nodes and edges), and steinb18 (100 nodes, 200 edges) from the OR library [46]. Each edge was randomly assigned a cost value $Co(e) \in \{1, 2, \ldots, 10\}$; the capacity of edges was set by $Ca(e) = 12$.

5.1. Computational experiments for LSA pre-processing

For the steinb-instances, 20 requests were generated randomly. For comparison purposes, the same set of requests was chosen for all three underlying labelled graphs (the cost labels are different). The set of requests $\{R_1, \ldots, R_{20}\}$ is presented in Table 1. From the 20 requests, we derived 12 multicast routing problems P_i, with P_1 defined by

Table 1
Set of single requests for graphs steinb10-11 and steinb18 [46]

Request no.	Source node	Destination node(s)	Capacity
1	36	7, 23, 25, 40	3
2	17	15, 30, 31, 40, 41, 46	2
3	48	36, 58	8
4	41	13, 22, 27, 35, 50	2
5	2	6, 14, 18, 23, 27, 33, 47, 49	4
6	13	28	7
7	50	5, 12, 28, 31, 44, 45	2
8	24	20, 29, 30	3
9	52	9, 13, 22, 55	2
10	53	13, 14, 28, 41, 52, 55	1
11	10	5, 20, 31, 40	3
12	66	18, 20, 22, 23	2
13	14	6, 16, 36	4
14	61	15, 20, 33, 38	6
15	55	4, 21, 41	5
16	14	9, 16, 31, 43, 44	3
17	67	23, 29	6
18	9	4, 6, 7, 30, 31, 35	2
19	69	10, 40, 54	2
20	75	33, 57	7

Table 2
Solutions and $Z(S_{\text{best}})$ for $\Gamma = G_0/8$ and $T = 10^4$ (steinb10)

P_i	Solution for $T = 10^4$ and $\Gamma = G_0/8$	$Z(S_{\text{best}})$
1	[1, 4, 6, 2, 5, 3, 7, 8, 9]	1098
2	[4, 9, 7, 2, 6, 10, 1, 8, 5, 3]	1164
3	[5, 4, 2, 10, 7, 3, 11, 6, 8, 1, 9]	1305
4	[2, 12, 1, 9, 10, 11, 5, 7, 3, 4, 6, 8]	1379
5	[2, 4, 5, 7, 3, 8, 13, 12, 6, 10, 1, 9, 11]	1535
6	[13, 10, 14, 1, 2, 7, 12, 8, 6, 5, 11, 4, 3, 9]	1825
7	[10, 7, 9, 13, 14, 5, 15, 8, 4, 3, 1, 6, 11, 2, 12]	2086
8	[8, 6, 13, 7, 2, 11, 1, 3, 10, 12, 14, 16, 15, 4, 9, 5]	2265
9	[7, 2, 4, 10, 3, 8, 13, 5, 17, 12, 9, 16, 11, 15, 6, 1, 14]	2321
10	[14, 7, 15, 1, 10, 17, 3, 4, 13, 5, 2, 9, 16, 11, 8, 6, 18, 12]	2468
11	[7, 16, 3, 18, 2, 13, 10, 6, 4, 14, 17, 9, 8, 1, 19, 12, 5, 15, 11]	2552
12	[5, 12, 20, 4, 8, 18, 3, 16, 13, 9, 7, 19, 10, 14, 6, 17, 15, 2, 11, 1]	2655

$\{R_1, R_2, \ldots, R_9\}$, and P_{12} defined by $\{R_1, R_2, \ldots, R_{20}\}$. For each P_i, we performed 12 computational experiments: for each of the two values of $T = 10^4, 2 \times 10^4$ the experiments were executed for six different values of Γ in (13), where $\Gamma := G_0/c$ for $c = 1, 2, 4, 8, 16, 20, 32$, and $G_0 := |Z(S_0^1) - Z(S_0^2)|$.

Table 2 shows an example of solutions (final order of requests for each P_i) for steinb10 and $T = 10^4$, $G_0/8$. We note that the values of $Z(S_{\text{best}})$ did not change for $T = 3 \times 10^4$. In Table 3, we present a complete picture of runs for steinb10 and all P_i with $T = 2 \times 10^4$ and $\Gamma := G_0/c$ for $c = 1, 2, 4, 8, 16, 20, 32$. The results demonstrate that the best values of the objective function are found for Γ values that are much smaller than the initial value derived from the two initial solutions. It is remarkable that the value of G_0/c where the value of the objective function stabilizes in incoherent for the different problems P_i. We note that for $G_0/20$ we obtain already the best values for $Z(S_{\text{best}})$.

Based on the results from Table 3, we estimated Γ by the procedure described in Section 3.4, i.e. $\Gamma := G_{\text{est}} := \max\{\Delta_{s(T)}, G_{\text{est}}\}$, and the implementation was executed for both values of T and each P_i, $i = 1, \ldots, 12$; cf. Table 4. If we now compare the outcomes for $G_0/20$ in Table 3, where we obtain the best results for $Z(S_{\text{best}})$, and for G_{est} in

Table 3
Z_{best} for $T = 2 \times 10^4$ and different Γ settings (steinb10)

No. of P_i	Size of P_i	G_0 values	$T = 2 \times 10^4$					
			$\Gamma = G_0/2$	$G_0/4$	$G_0/8$	$G_0/16$	$G_0/20$	$G_0/32$
1	9	19	1104	1098	1098	1098	1098	1098
2	10	19	1164	1164	1164	1164	1164	1164
3	11	28	1305	1305	1305	1305	1305	1305
4	12	49	1379	1379	1379	1379	1379	1379
5	13	108	1551	1535	1531	1531	1528	1528
6	14	148	1861	1817	1812	1812	1812	1812
7	15	156	2135	2131	2086	2080	2080	2080
8	16	156	2265	2251	2251	2238	2238	2238
9	17	156	2355	2321	2320	2314	2314	2314
10	18	178	2485	2485	2468	2448	2442	2442
11	19	190	2563	2557	2524	2520	2516	2516
12	20	194	2664	2636	2636	2636	2632	2632

Table 4
Z_{best} for G_{est} (steinb10)

Size of P_i	G_{est} values	$Z(S_{best})$ values	
		$T = 10^4$	$T = 2 \times 10^4$
9	6	1098	1098
10	6	1164	1164
11	17	1305	1305
12	30	1384	1379
13	59	1611	1551
14	68	1861	1825
15	71	2163	2135
16	143	2314	2279
17	146	2391	2355
18	157	2531	2486
19	169	2573	2567
20	171	2721	2693

Table 4, and if we take into account the relation between G_{est} and $G_0/20$, we conclude that

$$\Gamma_{est} \approx \frac{G_{est}}{10} \tag{15}$$

is an appropriate choice for Γ in (13); cf. Table 5.

We note that for $\Gamma_{est} = G_{est}/10$ we not always obtain the best solutions we achieved by applying LSA (cf. Tables 3, 5, 6, 8, 9, 11). But the relation demonstrates on one hand the big difference between G_{est} and Γ_{est}, and on the other hand it seems to be preferable for our search procedure to slightly over-estimate the value of Γ in (13) because it increases the certainty that the search procedure can "escape" from local minima.

In Tables 6–11 we present the corresponding results (for P_7,…, P_{12} only) for steinb11 and steinb18. The results demonstrate that the estimation according to (15) is largely independent of the underlying network structure, i.e. the results suggest that the relation between G_{est} and Γ_{est} rather depends on the functions *Co* and *Ca* than on structural parameters.

Thus, for a given multicast routing problem P with fixed edge capacities, as defined in (2), one can proceed as follows: firstly, the simulated annealing-based algorithm is executed for $\Gamma = G_0$, and G_{est} is estimated according to the procedure from Section 3.4. Secondly, the calculations are repeated for $\Gamma = G_{est}/10$, i.e. the total number of runs is reduced to two. If only LSA is used to minimise the value of the objective function, the constant c in $G_{est}/10$ can be chosen slightly larger than in (15).

Table 5
Z_{best} for $\Gamma_{est} \approx G_{est}/10$

Size of	$Z(S_{best})$ values	
P_i	$T = 10^4$	$T = 2 \times 10^4$
9	1098	1098
10	1164	1164
11	1305	1305
12	1379	1379
13	1535	1531
14	1823	1812
15	2113	2084
16	2249	2242
17	2331	2319
18	2462	2451
19	2540	2521
20	2648	2636

Table 6
Z_{best} for $T = 2 \times 10^4$ and different Γ settings (steinb11)

No. of	Size of	G_0	$T = 2 \times 10^4$					
P_i	P_i	values	$\Gamma = G_0/2$	$G_0/4$	$G_0/8$	$G_0/16$	$G_0/20$	$G_0/32$
7	15	200	2082	2022	1994	1974	1942	1942
8	16	227	2220	2177	2148	2108	2081	2081
9	17	302	2321	2290	2265	2230	2207	2207
10	18	296	2439	2412	2382	2353	2329	2329
11	19	337	2506	2457	2428	2406	2382	2382
12	20	419	2728	2702	2683	2668	2589	2589

Table 7
Z_{best} for G_{est} (steinb11)

Size of	G_{est}	$Z(S_{best})$ values	
P_i	values	$T = 10^4$	$T = 2 \times 10^4$
15	131	2084	2042
16	176	2224	2188
17	223	2310	2303
18	170	2447	2429
19	201	2531	2482
20	258	2748	2716

Table 8
Z_{best} for $\Gamma_{est} \approx G_{est}/10$

Size of	$Z(S_{best})$ values	
P_i	$T = 10^4$	$T = 2 \times 10^4$
15	1994	1977
16	2148	2113
17	2265	2239
18	2382	2367
19	2428	2411
20	2683	2674

Table 9
Z_{best} for $T = 2 \times 10^4$ and different Γ settings (steinb18)

No. of	Size of	G_0	$T = 2 \times 10^4$					
P_i	P_i	values	$\Gamma = G_0/2$	$G_0/4$	$G_0/8$	$G_0/16$	$G_0/20$	$G_0/32$
7	15	223	2273	2252	2236	2218	2204	2204
8	16	268	2421	2412	2396	2381	2375	2375
9	17	313	2625	2597	2582	2554	2532	2532
10	18	353	2705	2693	2679	2657	2641	2641
11	19	397	2807	2797	2765	2749	2735	2735
12	20	522	3096	3065	3018	2993	2988	2988

Table 10
Z_{best} for G_{est} (steinb18)

Size of	G_{est}	$Z(S_{best})$ values	
P_i	values	$T = 10^4$	$T = 2 \times 10^4$
15	203	2279	2263
16	206	2423	2414
17	235	2638	2607
18	302	2724	2695
19	329	2817	2802
20	418	3112	3076

Table 11
Z_{best} for $\Gamma_{est} \approx G_{est}/10$

Size of	Z_{best} values	
P_i	$T = 10^4$	$T = 2 \times 10^4$
15	2245	2234
16	2407	2389
17	2586	2562
18	2686	2670
19	2791	2755
20	3048	3009

5.2. Computational experiments for genetic local search

The implementation described in Section 3.4 has been extended by the PMX crossover algorithm and modified with respect to the quasi-deterministic local search presented in Algorithm 4.2, see Section 4.2. The multicast routing problems are the same as in Section 5.1, i.e. we analyze 12 routing tasks defined for three networks that are based on the instances no. steinb10, steinb11, and steinb18 from the OR library [46, cf. Table 1].

The parameters M, K, and N were chosen in such a way ($L = 50$ is fixed) that $M \cdot K \cdot N$ is in the region of T from our experiments with LSA; cf. Tables 4–11 in Section 5.1. Thus, the particular parameter settings were $M = 7, 10$, $K = 70, 80$, and $N = 20, 25$.

In Tables 12, 14, and 16 , the numbers in bold face indicate improvements in comparison to the results obtained by LSA, as shown in Tables 3, 6, and 9. The setting $N_{PMX} = 223$ in Tables 13, 15, and 17 is motivated in Section 5.3; see (20) and (21) (Tables 13–17).

As can be seen, the genetic local search with LSA pre-processing performs better on larger multicast routing instances compared to applications of either LSA or PMX only, especially for a total number of operations $M \cdot K \cdot N$ that is equivalent to $T = 2 \times 10^4$, where we achieved the best results for LSA. For all of the instances, LSA produces better

Table 12
Z_{best} for *GLS* with elitist PMX (steinb10)

Size of P_i	$N = 20$, $K = 70$, $L = 50$, $M = 7$	$N = 25$, $K = 80$, $L = 50$, $M = 10$
15	2082	2080
16	2242	**2236**
17	2316	2314
18	2446	2442
19	2520	2516
20	2636	**2626**

Table 13
Z_{best} for PMX alone

Size of P_i	$N_{PMX} = 223$ $M = 10$
15	2227
16	2396
17	2539
18	2657
19	2755
20	2999

Table 14
Z_{best} for *GLS* with elitist PMX (steinb11)

Size of P_i	$N = 20$, $K = 70$, $L = 50$, $M = 7$	$N = 25$, $K = 80$, $L = 50$, $M = 10$
15	1952	**1931**
16	2087	2070
17	2206	**2171**
18	2317	2284
19	2338	**2319**
20	2583	**2571**

Table 15
Z_{best} for PMX alone

Size of P_i	$N_{PMX} = 223$ $M = 10$
15	2094
16	2249
17	2340
18	2454
19	2540
20	2652

results than the use of PMX crossover only. We note that omitting elitist solutions in GLS runs produces worse results compared to LSA-based search.

5.3. Run-time analysis

As mentioned at the end of Section 5.1, the pre-processing step has to be executed only once for a given network structure and sample routing tasks in order to (roughly) estimate the value of Γ_{est}. Thus, the time complexity of our

Table 16
Z_{best} for *GLS* with elitist PMX (steinb18)

Size of P_i	$N = 20$, $K = 70$, $L = 50$, $M = 7$	$N = 25$, $K = 80$, $L = 50$, $M = 10$
15	2216	2204
16	2379	2375
17	2523	**2500**
18	2646	**2634**
19	2736	**2723**
20	2967	**2947**

Table 17
Z_{best} for PMX alone

Size of P_i	$N_{PMX} = 223$ $M = 10$
15	1978
16	2105
17	2223
18	2336
19	2390
20	2602

Table 18
Solution times for LSA-runs

P_i	steinb10			steinb11			steinb18		
	Z_I	Z_F	Time	Z_I	Z_F	Time	Z_I	Z_F	Time
1	1117	1098	2569	1207	1089	2863	1481	1335	2888
2	1183	1164	2650	1258	1137	2951	1496	1378	3156
3	1330	1305	3024	1343	1218	3173	1648	1486	3316
4	1428	1379	3402	1396	1275	3385	1778	1594	3525
5	1636	1528	3495	1558	1409	3527	1971	1722	3720
6	1966	1812	3557	1938	1757	3806	2326	2032	4117
7	2236	2080	3801	2235	1942	4065	2586	2204	4507
8	2392	2238	4169	2387	2081	4587	2775	2375	4659
9	2470	2314	4093	2584	2207	4836	3147	2532	5121
10	2618	2442	4389	2702	2329	4962	3299	2641	5238
11	2724	2516	5025	2784	2382	5052	3391	2735	5309
12	2836	2632	5376	3156	2589	5541	3706	2988	5797

approach is actually determined by the procedure described in Section 4.2. However, in both cases (LSA pre-processing and genetic local search), the most time-consuming part is the execution of the KMB algorithm [48] for n simultaneous routing requests R_i defined by the multicast routing problem $P = [G; Co; Ca; R_1, \ldots, R_n]$; see (2). For G with p nodes and a maximum number q of destination nodes in requests R_i (e.g., $q = 8$ for the requests in Table 1), the time complexity to execute the KMB algorithm is $O(q \cdot p^2)$ [48]. According to Section 3.2 and Fig. 2, the maximum number of requests R_i that have to be re-routed in a single neighbourhood transition is upper bounded by n. Thus, for T Markov chain transitions, the time complexity of the procedure described in Section 3.4 can be roughly upper bounded by

$$O(T \cdot n \cdot q \cdot p^2). \tag{16}$$

In Table 18, we report the values Z_I of the initial solutions, the values Z_F of final solutions and the computation time (in seconds) for the best runs by LSA as described in Section 3.4.

Table 19
Solution times for GLS-runs

P_i	steinb10		steinb11		steinb18	
	Z_F	Time	Z_F	Time	Z_F	Time
7	2080	7638	1931	7749	2204	7938
8	2236	8275	2070	7912	2375	8155
9	2314	8561	2171	8328	2500	8625
10	2442	8906	2284	8694	2634	9132
11	2516	9163	2319	9052	2723	9527
12	2626	9520	2571	9576	2947	9743

The time ranges from 43 min (P_1 and steinb10) to 97 min (P_{12} and steinb18). The improvement of the objective function ranges from 1.7% (P_1 and steinb10) to almost 20% for P_{12} and steinb18. The latter value seems quite significant, given the size of the problem and the time spent on the improvement.

We recall that the PMX operator requires the execution of the KMB algorithm for both offsprings of each pair from the current population of size M; see Fig. 3. This results in an upper bound of $2 \cdot M \cdot (M-1)/2 \cdot O(n \cdot q \cdot p^2)$, which is repeated N times. Furthermore, we execute $N \cdot K$ search steps of complexity $O(n \cdot q \cdot p^2)$ for each of the M individuals of the population; see (3)–(5) in Section 4.2. Here, we simply assume $L = O(1)$, since it is difficult to estimate how often the L-test for local minima is executed. If we now translate the parallel execution of M threads on a PC-cluster into sequential time, we obtain for genetic local search the upper bound

$$N \cdot M \cdot (M-1) \cdot O(n \cdot q \cdot p^2) + M \cdot N \cdot K \cdot O(n \cdot q \cdot p^2), \tag{17}$$

$$= M \cdot N \cdot K \cdot O(n \cdot q \cdot p^2) \cdot \left(\frac{M-1}{K} + 1 \right), \tag{18}$$

$$= O(M \cdot N \cdot K \cdot n \cdot q \cdot p^2), \tag{19}$$

if $K > M$ as in our choice of parameters. For $M \cdot K \cdot N = T$ we have in (19) the same upper bound as in (16), but one can expect the actual run-time to be larger compared to LSA-runs due to $L = 50$ and the auxiliary computations related to the communication between the M threads (individuals of the population).

In Section 5.2, we presented Z_{best} for $P_7, \ldots, P_{12}$ obtained by our genetic local search procedure. For $M \cdot K \cdot N = 2 \times 10^4$, Table 19 shows the corresponding run-times (in seconds) for all three steinb-instances and the associated multicast routing problems $P_7, \ldots, P_{12}$. The run-times are for a sequential simulation of M independent runs for each individual of the population, which allows a direct comparison to Table 18; see also (19). Since we have M independent runs, Z_I is not mentioned explicitly.

We see that in terms of sequential run-time, the amount of time increases by 68%, ..., 80% (steinb18) in order to obtain the better results by genetic local search. The time ranges from 127 min (P_7 and steinb10) to 163 min (P_{12} and steinb18). Compared to LSA-runs (see Table 18), the improvement of the objective function ranges from 0.09% (P_8 and steinb10) to almost 1.4% for P_{12} and steinb18. If the program is executed on a PC cluster of size M, the time has to be divided roughly by $M = 10$, i.e. one obtains better results in shorter time at the expense of hardware costs. In this case, the run-time would range from about 13 min (P_7 and steinb10) to 17 min (P_{12} and steinb18).

Finally, if PMX crossover is executed N_{PMX} times, where each execution time is upper bounded by $M \cdot (M-1) \cdot O(n \cdot q \cdot p^2)$, we obtain

$$N_{PMX} \cdot M \cdot (M-1) \cdot O(n \cdot q \cdot p^2). \tag{20}$$

If we require the same order of time complexity as in (16) and (19), we have to set $N_{PMX} \cdot M \cdot (M-1) = M \cdot N \cdot K = T$, i.e. we obtain

$$N_{PMX} = \frac{N \cdot K}{M-1} = \frac{T}{M \cdot (M-1)}. \tag{21}$$

Table 20
Solution times for PMX-runs

P_i	steinb10		steinb11		steinb18	
	Z_F	Time	Z_F	Time	Z_F	Time
7	2094	7503	1978	7638	2227	7859
8	2249	8124	2105	7795	2396	7981
9	2340	8311	2223	8248	2539	8533
10	2454	8735	2336	8462	2657	9058
11	2540	9032	2390	8941	2755	9375
12	2652	9384	2602	9407	2999	9582

Table 21
The problem P_{sample} with six requests

Request	Source	Destination(s)	Capacity
R_1	G	A, C, F	5
R_2	A	E, F, G	4
R_3	H	B, D, F, G	3
R_4	C	E	5
R_5	E	A, B, F	3

Thus, for $T = 2 \times 10^4$ we have $N_{PMX} \approx 223$. In Table 20, we present the run-times for $N_{PMX} := 223$ with N_{PMX} applications of the PMX operator. Since permanently PMX crossover is applied, these are sequential runs where a population of size M is maintained.

Compared to LSA-runs, all the solutions on P_7,..., P_{12} as well as the run-times are worse for PMX-runs. Although the order of the time complexity estimation is the same for (16) and (20) if $N_{PMX} := 223$ in (21), the fact that the run-times are worse for PMX-runs can be explained by auxiliary calculations related to the PMX operator (e.g., selection of M fittest solutions after each step), which are hidden in the constants of the time complexity estimation.

Since the GLS-runs produce on most instances better results than LSA-runs, the PMX-runs are worse on all instances compared to GLS (e.g. by about 1.8% on P_{12} for steinb18). The run-times of PMX-runs differ only marginally from the times for GLS-runs, which confirms (20) and the setting $N_{PMX} := 223$ according to (21). As mentioned above, the run-times of sequential GLS simulations can be divided roughly by M for parallel executions of the program. The PMX-runs could be parallelised by using M processors to handle the $M \cdot (M-1)/2$ crossover operations, but this would involve additional communication time between the processors.

5.4. Comparison to CPLEX solutions

Recently, the CPLEX programming package [57] has been employed to solve routing-related optimisation problems; see [58,59]. We used the publicly available version from [57] to compare CPLEX-solutions to our approach. Due to the limitations of the publicly available version (300 variables and 300 constraints), we executed the comparison on a small example at the limits of this version. The multicast routing tasks P_7,..., P_{12} from Section 5.1 (e.g., P_{12} with 20 requests defined on a graph with 100 nodes and 200 edges) would require the definition of about 9000 to 12 000 variables plus constraints.

The example $P_{sample} = [G; Co; 11; R_1, \ldots, R_5]$ is defined in Table 21, and the network information is shown in Fig. 4. The problem P_{sample} generates about 200 variables and close to 300 constraints as input to the CPLEX program, i.e. it is already close to the limits of the publicly available version.

The CPLEX program as well as the LSA procedure were started with the order of requests as given in Table 21. The CPLEX problem produced a solution of cost value 308 in a fraction of a second, while the LSA procedure arrived at 253 for the first time after $T = 11$ Markov chain transitions (approximately the same value in repeated runs), also in a fraction of a second. The value 253 was returned for the order $[R_4, R_2, R_5, R_3, R_1]$ of requests. Both results were

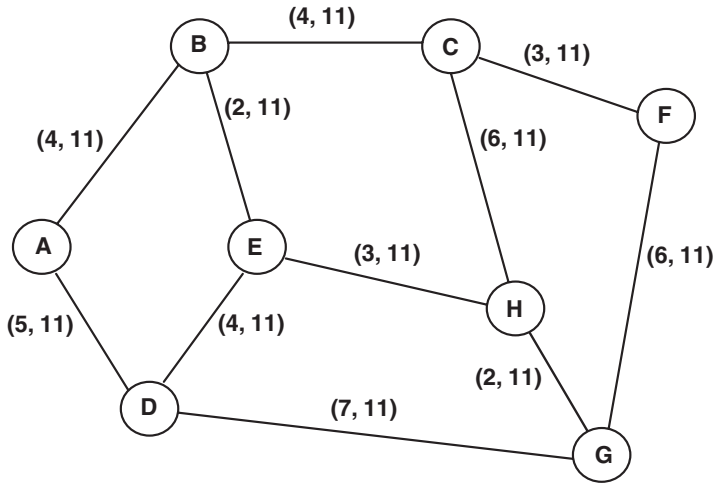

Fig. 4. The underlying network of P_{sample} with labels (cost, capacity = 11).

obtained on comparable hardware. Thus, due to the search-based nature and the neighbourhood relation from Fig. 2, the LSA procedure seems to have an advantage over CPLEX on this type of "order-sensitive" problems.

The value 253 remained the best value for $T = 10^4$ Markov chain transitions, and the LSA procedure terminated after 13.8 s (of course, for this small example, one could look-up all 120 permutations in shorter time). If we now adapt the results and methodology from [40] (see also paragraph after Theorem 1), in particular the empirical hypothesis from Section 4.1 in [40], we can estimate the confidence that 253 is the optimum solution: the value 253 was obtained for $\Gamma := 3.6 = G_{\text{est}}/10$, which was chosen according to (15); the maximum size n of the neighbourhood relation for P_{sample} is upper bounded by $5 \times 4/2 = 10$, i.e. from $T = 10^4 = (n/\delta)^{\Gamma} = (10/\delta)^{3.6}$ we obtain $\delta = 10/10^{4/3.6} \approx 0.77$. Thus, we have a confidence $1 - \delta$ of about 0.33 (33%) that 253 is the optimum solution for P_{sample}, if the routing of single requests is executed by the KMB algorithm.

6. Conclusion

We introduced a genetic local search heuristic that utilises logarithmic simulated annealing (LSA) in a pre-processing step for an analysis of the landscape generated by a multicast routing problem and the associated objective function. The genetic local search employs the partially mixed crossover (PMX) operation in-between sequences of downward and upward search steps, where the elitist model is applied. The PMX operation seems to be particularly suited to problems like multicast routing, since the outcome of the operation is always defined for two given parent routing orders. The computational experiments were executed on three synthetic networks that are based on the instances steinb10, steinb11, and steinb18 from the OR library. The results show that the random walk that is guided by a parameter obtained from the landscape analysis together with the PMX operation provide better results on most routing instances compared to "pure" simulated annealing-based search as well as to applications of PMX crossover only. We note that to achieve these results the elitist approach was essential. Future research will concentrate on implementations in distributed systems, where the size of the population can be chosen much larger than in the present study.

Acknowledgement

The author would like to thank the anonymous referees for their very careful reading of the manuscript and many helpful suggestions that resulted in an improved presentation.

References

[1] Oliveira CAS, Pardalos PM. A survey of combinatorial optimization problems in multicast routing. Computers & Operations Research 2005;32:1953–81.

[2] Wang Z, Bingxin S, Liu W. A distributed dynamic heuristic for delay-constrained least-cost multicast routing. Journal of Interconnection Networks 2000;1(4):331–44.
[3] Novak R, Rugelj J, Kandus G. A note on distributed multicast routing in point-to-point networks. Computers & Operations Research 2001;28:1149–64.
[4] Diot C, Dabbous W, Crowcroft J. Multipoint communication: a survey of protocols, functions, and mechanisms. IEEE Journal on Selected Areas in Communications 1997;15:277–90.
[5] Muchnik VB, Shafarenko AV. Dynamic evaluation strategy for fine-grain data-parallel computing. IEE Proceedings—Computers and Digital Techniques 1996;143:181–8.
[6] Salama HF, Reeves DS, Viniotis Y. Evaluation of multicast routing algorithms for real-time communication on high-speed networks. IEEE Journal on Selected Areas in Communications 1997;15:332–45.
[7] Mir-Fakhraei N. An efficient multicast approach in an ATM switching network for multimedia applications. Journal of Network and Computer Applications 1998;21:31–9.
[8] Yeo CK, Lee BS, Er MH. A framework for multicast video streaming over IP networks. Journal of Network and Computer Applications 2003;26:273–89.
[9] Yeo CK, Lee BS, Er MH. A survey of application level multicast techniques. Computer Communications 2004;27:1547–68.
[10] Li X, Veeravalli B. Design and performance analysis of multimedia document retrieval strategies for networked video-on-reservation systems. Computer Communications 2005;28:1910–24.
[11] Masip-Bruin X, Yannuzzi M, Domingo-Pascual J, Fonte A, Curado M, Monteiro E, Kuipers F, Van Mieghem P, Avallone S, Ventre G, Aranda-Gutiérrez P, Hollick M, Steinmetz R, Iannone L, Salamatian K. Research challenges in QoS routing. Computer Communications 2006;29: 563–81.
[12] Yang CB, Wen UP. Applying tabu search to backup path planning for multicast networks. Computers & Operations Research 2005;32: 2875–89.
[13] Banerjee S, Kommareddy C, Kar K, Bhattacharjee B, Khulle S. OMNI: an efficient overlay multicast infrastructure for real-time applications. Computer Networks 2006;50(6):826–41.
[14] Harrison T, Williamson C. A performance study of multicast routing algorithms for ATM networks. In: Proceedings of 21st annual IEEE conference on local computer networks. 1996. p. 191–201.
[15] Zhang QF, Leung YW. An orthogonal genetic algorithm for multimedia multicast routing. IEEE Transactions on Evolutionary Computation 1999;3:53–62.
[16] Karp RM. Reducibility among combinatorial problems. Complexity of computer computations. New York: Plenum Press; 1972 p. 85–103.
[17] Zhu L, Wainwright RL, Schoenefeld DA. A genetic algorithm for the point to multipoint routing problem with varying number of requests. In: Proceedings of IEEE international conference on evolutionary computation. 1998. p. 171–6.
[18] Galiasso P, Wainwright RL. A hybrid genetic algorithm for the point to multipoint routing problem with single split paths. In: Proceedings of ACM Symposium on Applied Computing. 2001. p. 327–32.
[19] Ericsson M, Resende MGC, Pardalos PM. A genetic algorithm for the weight setting problem in OSPF routing. Journal of Combinatorial Optimization 2002;6(3):299–333.
[20] Barolli L, Koyama A, Suganuma T, Shiratori N. GAMAN: a GA based QoS routing method for mobile Ad-Hoc networks. Journal of Interconnection Networks 2003;4(3):251–70.
[21] Yang WL. A heuristic algorithm for the multi-constrained multicast tree. Proceedings of the sixth IFIP/IEEE international conference on management of multimedia networks and services, Lecture Notes in Computer Science, vol. 2839, 2003. p. 78–89.
[22] Skorin-Kapov N, Kos M. The application of Steiner trees to delay constrained multicast routing: a tabu search approach. Proceedings of the seventh international conference on telecommunications, vol. 2, 2003. p. 443–8.
[23] Wang H, Fang J, Wang H, Sun YM. TSDLMRA: an efficient multicast routing algorithm based on tabu search. Journal of Network and Computer Applications 2004;27(2):77–90.
[24] Roy A, Das SK. QM^{2}RP: a QoS-based mobile multicast routing protocol using multiobjective genetic algorithms. Wireless Networks 2004;10(3):271–86.
[25] Kampstra P. Evolutionary computing in telecommunications. BMI paper, August 2005. Vrije Universiteit Amsterdam.
[26] Kirkpatrick S, Gelatt Jr CD, Vecchi MP. Optimization by simulated annealing. Science 1983;220:671–80.
[27] Černy V. A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm. Journal of Optimization Theory and Applications 1985;45:41–51.
[28] Hajek B. Cooling schedules for optimal annealing. Mathematics of Operations Research 1988;13:311–29.
[29] Catoni O. Rough large deviation estimates for simulated annealing: applications to exponential schedules. Annals of Probability 1992;20: 1109–46.
[30] Aarts EHL. Local search in combinatorial optimization. New York: Wiley; 1998.
[31] Cui Y, Xu K, Wu JP, Yu ZC, Zhao YJ. Multi-constrained routing based on simulated annealing. Proceedings IEEE international conference on communications, vol. 3, 2003. p. 1718–22.
[32] Wang XW, Cheng H, Cao J, Zheng LW, Huang M. A simulated-annealing-based QoS multicasting algorithm. In: Proceedings international conference on communication technology. 2003. p. 469–73.
[33] Wang XL, Jiang Z. QoS multicast routing based on simulated annealing algorithm. In: Proceedings SPIE on network architectures, management, and applications. 2004. p. 511–6.
[34] Kun Z, Heng W, Feng-Yu L. Distributed multicast routing for delay and delay variation-bounded Steiner tree using simulated annealing. Computer Communications 2005;28:1356–70.
[35] Wang X, Cao J, Cheng H, Huang M. QoS multicast routing for multimedia group communications using intelligent computational methods. Computer Communications 2006;29:2217–29.

[36] Wright S. The roles of mutation, inbreeding, crossbreeding and selection in evolution. Proceedings of the sixth international congress on genetics, vol. 1, 1932. p. 356–66.
[37] Merz P, Freisleben B. Fitness landscapes, memetic algorithms, and greedy operators for graph bipartitioning. Evolutionary Computation 2000;8(1):61–91.
[38] Reidys CM, Stadler PF. Combinatorial landscapes. SIAM Review 2002;44(1):3–54.
[39] Wales D. Energy landscapes. Cambridge: Cambridge University Press; 2003.
[40] Albrecht AA. A stopping criterion for logarithmic simulated annealing. Computing 2006;78:55–79.
[41] Ferreira FF, Fontanari JF, Stadler PF. Landscape statistics of the low-autocorrelation binary string problem. Journal of Physics A 2000;33: 8635–47.
[42] Wolfinger MT, Svrcek-Seiler WA, Flamm C, Hofacker IL, Stadler PF. Exact folding dynamics of RNA secondary structures. Journal of Physics A 2004;37:4731–41.
[43] Schmitt LM. Theory of genetic algorithms. Theoretical Computer Science 2001;259:1–61.
[44] Zahrani MS, Loomes MJ, Malcolm JA, Albrecht AA. Genetic local search for multicast routing, Proceedings of genetic and evolutionary computation conference (GECCO'06), 2006. p. 615–16.
[45] Steinhöfel K, Albrecht A, Wong CK. Fast parallel heuristics for the job shop scheduling problem. Computers & Operations Research 2002;29(2):151–69.
[46] Beasley JE. OR library: ⟨http://people.brunel.ac.uk/~mastjjb⟩.
[47] Hakimi SL. Steiner's problem in graphs and its implications. Networks 1971;1:113–33.
[48] Kou L, Markowsky G, Berman L. A fast algorithm for Steiner trees. Acta Informatica 1981;15:141–5.
[49] Doar M, Leslie IM. How bad is naïve multicast routing?. In: Proceedings of IEEE INFOCOM. 1993. p. 82–9.
[50] Alexander MJ, Robins G. New performance-driven FPGA routing algorithm. In: Proceedings of ACM/SIGDA design automation conference. 1995. p. 562–7.
[51] Koch T, Martin A. Solving Steiner tree problems in graphs to optimality. Networks 1998;32:207–32.
[52] Prömel HJ, Steger A. A new approximation algorithm for the Steiner tree problem with performance ratio 5/3. Journal of Algorithms 2000;36: 89–101.
[53] Goldberg DE. Genetic algorithms in search. Optimization and machine learning. New York: Addison-Wesley; 1989.
[54] Holland JH. Genetic algorithms. Scientific American 1992;267(1):66–72.
[55] Rudolph G. Convergence analysis of canonical genetic algorithms. IEEE Transactions on Neural Networks 1994;5:96–101.
[56] Baudet P, Azzaro C, Pibouleau L, Domenech S. A genetic algorithm for batch chemical plant scheduling. In: Proceedings of international congress of chemical and process engineering. 1996. p. 25–30.
[57] CPLEX: ⟨http://www.ilog.com/products/cplex/⟩.
[58] Pompili D, Lopez L, Scoglio C. Multicast algorithms in service overlay networks. Proceedings of IEEE INFOCOM, 2006.
[59] Guo S, Yang O. Minimum-energy multicast in wireless ad hoc networks with adaptive antennas: MILP formulations and heuristic algorithms. IEEE Transactions on Mobile Computing 2006;5(4):333–46.