Appl Intell (2014) 40:305–321
DOI 10.1007/s10489-013-0460-6

A memetic algorithm for the capacitated *m*-ring-star problem

Zizhen Zhang · Hu Qin · Andrew Lim

Published online: 18 August 2013

Abstract The Capacitated m-Ring-Star Problem (CmRSP) models a network topology design problem in the telecommunications industry. In this paper, we propose to solve this problem using a memetic algorithm that includes a crossover operation, a mutation operation, a local search involving three neighborhood operators, and a population selection strategy that maintains population diversity. Our approach generates the best known solutions for 131 out of 138 benchmark instances, improving on the previous best solutions for 24 of them, and exhibits more advantages on large benchmark instances when compared with the best existing approach. Additionally, all existing algorithms for this problem in literature assume that the underlying graph of the problem instance satisfies the triangle inequality rule; our approach does not require this assumption. We also generated a new set of 36 larger test instances based on a digital data service network price structure to serve as a new benchmark data set for future researchers.

Keywords Capacitated m-ring-star · Memetic algorithm · Fiber-optic communications network

Z. Zhang (✉) · A. Lim
Department of Management Sciences, City University of Hong Kong, Tat Chee Avenue, Kowloon Tong, Kowloon, Hong Kong
e-mail: zzzhang@cityu.edu.hk

A. Lim
e-mail: lim.andrew@cityu.edu.hk

H. Qin
School of Management, Huazhong University of Science and Technology, No. 1037, Luoyu Road, Wuhan, China
e-mail: tigerqin@hust.edu.cn

H. Qin
e-mail: tigerqin1980@gmail.com

1 Introduction

The Capacitated m-Ring-Star Problem (CmRSP) is an important and practical problem in the field of network topology design for telecommunications, particularly when dealing with fiber-optic communication networks. It models the situation where a central telephone exchange (called the *depot*) is required to provide telecommunications services to a set of customers using m networks so as to minimize the total connection costs. Additionally, there are a number of transition points that can serve as way stations to reduce costs. Each network connects certain customers and/or transition points to the depot in a ring structure via high-quality fiber-optic cables, which prevents the failure of a single network node or link from causing the entire network to fail. For the remaining customers not in a ring, each is connected directly to a point in an existing ring using possibly cheaper cables.

The CmRSP is defined on a mixed graph $G = (V, E \cup A)$. The set of nodes $V = \{0\} \cup U \cup W$, where node 0 is the depot, U is the set of customers and W is the set of transition points (also called *Steiner nodes*). The edge set $E = \{(i, j) : i, j \in V, i \neq j\}$ is a complete set of undirected edges connecting all nodes in V; each edge $(i, j) \in E$ has an associated non-negative *routing cost* $c(i, j)$ representing the cost of connecting nodes i and j in a ring. In addition, each customer i can be directly connected to a subset of nodes $C_i \subseteq U \cup W$; the arc set $A = \{(i, j) : i \in U, j \in C_i\}$ is the set of all such possible connections. For each arc $(i, j) \in A$, there is an associated non-negative *allocation cost* $d(i, j)$ corresponding to the cost of directly connecting customer i to node j.

A *ring* R is a simple cycle consisting of edges in E that visits a subset of nodes including the depot. For a given ring R, there may be a number of customers $i \notin R$ that are con-

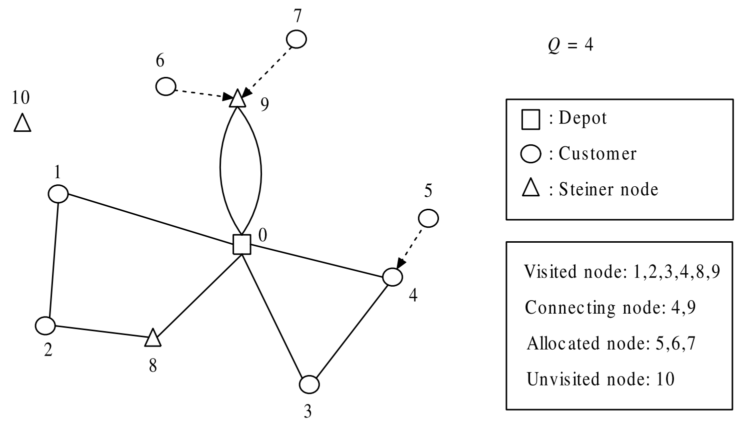


Fig. 1 An example of a feasible C*m*RSP solution

nected to nodes $j \in R$ by arcs $(i, j) \in A$. The resultant network topology is called a *ring-star*, denoted by $\tilde{R}$. The task of the C*m*RSP is to find a set of m ring-stars such that each customer is assigned to exactly one ring-star, each Steiner node is visited at most once and the number of customers in each ring-star does not exceed the capacity Q, where $Q \geq |U|/m$. The objective is to minimize the total routing and allocation costs.

A feasible solution to the C*m*RSP is represented by a set of m ring-stars $S = \{\tilde{R}_1, \tilde{R}_2, \ldots, \tilde{R}_m\}$. We say that a node $i \in S$ is a *visited node* if i is in a ring; otherwise it is an *allocated node*. Furthermore, a visited node i is also called a *connecting node* if there exists an edge $(i, j) \in S$ where j is an allocated node; we say that customer j is allocated to node i. By convention, we do not regard the depot as a visited node. Figure 1 gives an example of a feasible C*m*RSP solution. The C*m*RSP is easily to be shown as NP-hard in the strong sense, since we can reduce the well-known traveling salesman problem to it when $m = 1$, $Q = |U|$, $W = A = \emptyset$.

In the seminal work on the C*m*RSP by [2] that first introduced the problem, the authors generated two classes of test instances that have since been regarded as the standard benchmark test data by subsequent researchers. These instances have between 26 and 101 nodes, 3 to 5 ring-stars and varying ring-star capacities. It was stated in [2], *"Note that the set of routing costs of all generated instances satisfied the triangle inequality, therefore any optimal CmRSP solution could not use a Steiner node without assigning it a customer."* However, [12] found that the routing costs in some instances do not in fact satisfy the triangle inequality, which makes the conclusion of this statement invalid. In particular, there were instances where $c(i, j) + c(j, k) < c(i, k)$ and node j is a Steiner node, whereupon the inclusion of node j would reduce the total connection cost even if no customer is allocated to it. Unfortunately, due to this small oversight, the published solutions by [2] (and possibly other researchers) do not consider non-connecting Steiner nodes. Note that the routing costs are not required to fulfill the triangle inequality in the definition of the C*m*RSP in [2]. Therefore, [2] as well as some subsequent researchers such as [22, 23] did not actually study the C*m*RSP, but had instead studied a variant of the C*m*RSP in which non-connecting Steiner nodes could not be used.

The non-satisfaction of the triangle inequality is entirely reasonable. Although the connection and allocation costs are usually related to the distance between the connected nodes, the costs are seldom strictly proportional to the distance for many practical applications. A direct connection that traverses a long distance is likely to involve the excavation of difficult or highly populated terrain in order to lay the cables, which increases the costs involved. This terrain may be circumvented by using multiple indirect, shorter connections whose total cost is less than installing the direct connection. Another common scenario involves different fixed and variable costs for laying cables of different lengths [34], which also violates the triangle inequality.

There are three main contributions in this paper. Firstly, we propose a new memetic algorithm (MA) to solve the C*m*RSP, which combines a genetic algorithm with a local search component. Our approach is simple and practically implementable, consisting primarily of a crossover, a mutation and three basic neighborhood search operations. Furthermore, our approach does not assume that the routing costs satisfy the triangle inequality, and therefore it considers solutions with non-connecting Steiner nodes. Secondly, we thoroughly test our approach with the existing benchmark data generated by [2] and [23], some of which violate the triangle inequality. For the variant of the C*m*RSP that disallows non-connecting Steiner nodes, our MA approach with a simple repair procedure obtained the best known solutions for 131 out of 138 benchmark instances and improved on the previously best known solutions for 24 of them, outperforming all approaches in existing literature. Moreover, experimental results clearly show that our approach is more suitable to solve the instances of large size when compared with the best existing approach. We also present the solutions of the benchmark instances directly obtained by our MA approach and found that the costs could be further reduced for more than one-third of them when non-connecting Steiner nodes are allowed. Thirdly, we generate a new CmRSP data set containing larger instances based on the cost structure detailed in [34] for digital data service networks, which expands on the existing benchmark test instances. The results by our MA approach on this new data set can serve as a baseline for future researchers.

This paper is organized as follows. Section 2 provides an overview of the existing literature on the CmRSP and related problems. We then explain our memetic algorithm in Sect. 3, describing our chromosome representation, crossover and mutation operators, local refinement scheme and population composition strategy. Our computational experiments are given in Sect. 4, which include experiments to decide parameter values; we also provide results on both existing and newly generated CmRSP benchmark test data. We conclude our article in Sect. 5 with some closing remarks.

2 Literature review

The premise behind the CmRSP is to create a *survivable network system* that can avoid the scenario where the entire network fails as the result of the failure of a small number of critical components. The general *survivable network design problem* aims to find the most cost-effective network where there are at least l disjoint paths between any pair of nodes. The problem can be defined in terms of *node-disjoint* or *edge-disjoint* paths, which ensures that the network survives up to $l-1$ node or edge failures, respectively. In the excellent survey on designing survivable networks by [10], the authors conducted polyhedral studies on the problem and proposed some optimization methods, such as heuristics and branch-and-cut, to solve it.

For many practical applications, it is sufficient to consider topologies that can survive a single failure (i.e., $l=2$). A simple and common tactic is to use a ring topology. For example, [34] addressed a particular digital data service network design problem, where the task is to interconnect a set of customer locations through a ring of end offices so as to minimize the total tariff cost and provide reliability.

An alternative is the ring-star topology. Labbé et al. [13] studied a generic telecommunication network and introduced the Ring Star Problem (RSP), which seeks the most cost-effective solution to connect all nodes in a single (uncapacitated) ring-star configuration. The article also analyzed the polyhedral properties of the problem and developed a branch-and-cut algorithm that could solve instances involving up to 300 nodes optimally. The ring-star topology is also considered to be a variant of the location-routing problem, which generalizes the allocation of customers to the facilities and the allocation of customers to the routes. Related problems in this area include the Vehicle Routing Allocation Problem [3], the Median Cycle/Tour Problem [5, 14], and the Plant Cycle Location Problem [15]. For more information about location-routing problems, we refer the reader to [21] and [30].

As a generalization of the RSP, the CmRSP was first proposed by [2], who presented two integer programming formulations for the problem along with a branch-and-cut algorithm that could solve small instances optimally in reasonable time. Subsequently, [11, 12] proposed another two exact algorithms, namely branch-and-price and branch-and-cut-and-price algorithms, to attack this problem. The size of instances that can be solved optimally by these exact algorithms is very limited. For the large instances widely encountered in practice, the best approaches so far have made use of efficient meta-heuristics. The first meta-heuristic for the CmRSP was proposed by [19], which is a GRASP algorithm [29] that incorporates the tabu search strategy to escape from local optima. More recently, [22, 23] designed several local search operations based on properties of the problem and then integrated them into a variable neighborhood search (VNS) framework. Currently, the best approach on existing benchmark instances is the VNS algorithm by [23] involving an initialization procedure, an improvement procedure, an integer linear programming based procedure, a modified assignment procedure, a Lin-Kernighan procedure and a shaking (perturbation) procedure combined with a threshold accepting criterion.

Aside from the CmRSP, two other variants of the RSP have recently received attention in the literature. Liefooghe et al. [17] investigated a bi-objective ring star problem that consists of locating a simple cycle through a subset of nodes of a graph while optimizing two kinds of cost. Baldacci and Dell'Amico [1] studied a multi-depot ring-star problem and proposed a primal-based heuristic, which can also be used for the CmRSP.

3 Memetic algorithm

The term *memetic algorithm* (MA) has been used to describe the incorporation of domain knowledge (e.g., in the form of a customized search procedure) into an evolutionary algorithm. The idea is to combine the ability of evolutionary algorithms to explore diverse regions of the search space together with domain-specific local search in order to leverage on the strengths of both types of approaches. We refer the reader to [24] for the full details on MA. Several recent studies have successfully applied MA to a variety of combinatorial optimization problems, such as the vehicle routing problems [7, 25], the graph coloring problem [18], the job shop scheduling problem [26] and the quadratic multiple container packing problem [31].

Our solution approach is a memetic algorithm (Algorithm 1), which combines a genetic algorithm with a local

Algorithm 1: The memetic algorithm framework for the CmRSP

1 Construct the initial population P;
2 **for** *num_gens generations* **do**
3   Offspring population $P_O \leftarrow \emptyset$;
4   **while** P_O *is not full* **do**
5   Randomly select two parent chromosomes from P;
6   Produce two offspring from the parent chromosomes using *Crossover* operator;
7   Put offspring chromosomes into P_O;
8   **foreach** *chromosome* $\chi \in P_O$ **do**
9   Convert χ into solution S;
10   Perform *Mutation* operation on S;
11   **for** *hc_iters times* **do**
12   Apply *Extraction-assignment* operator on S;
13   Apply *Steiner node insertion* operator on S;
14   Apply *Steiner node removal* operator on S;
15   Revert S into chromosome χ;
16     $P_N \leftarrow P \cup P_O$;
17   Update population P using P_N;

refinement procedure. The algorithm begins by constructing an initial population P consisting of $|P|$ chromosomes corresponding to feasible CmRSP solutions. In each generation, we first produce a set of offspring P_O from P using crossover operations. Next, we perform a mutation operation on each chromosome in P_O. We then perform a simple hill-climbing procedure on each solution in P_O that utilizes several neighborhood operators (lines 11–14). Finally, we update the population P using $P_N = P \cup P_O$ in a manner that maintains some population diversity (line 17). At any stage of the algorithm, we update the best solution found so far. We repeat this process for *num_gens* generations.

In this section, we will describe the various components of our MA in detail, namely the chromosome representation, crossover operation, mutation operation, local refinement procedure (including the neighborhood operators used) and the way of updating the population P at the end of each generation.

3.1 Chromosome representation

In our MA, each chromosome χ is a set of m rings (not ring-stars), i.e. $\chi = \{R_1, R_2, \ldots, R_m\}$, where $R_k = (0, v_k^1, v_k^2, \ldots, v_k^{q_k}, 0)$ is the k-th ring and q_k is the number of visited nodes. Let u_k be the number of customers in the k-th ring; in order for the solution to be feasible, we must have $u_i \le Q$ ($\forall i = 1, \ldots, m$) and no two rings can share a common node except the depot. We use the notation $V(\chi)$ to denote the set of nodes in the m rings in χ, and $V(R_k)$ to denote the set of nodes in ring R_k.

This chromosome representation χ corresponds to the best CmRSP solution where the ring portion of each ring-star is given by χ. To find the best way to allocate the remaining customers such that allocation cost is minimized, we can solve an *assignment problem* or *transportation problem* [4]. Let $G' = (X, Y, A')$ be a bipartite graph, where $X = U - V(\chi)$ is the set of customers to be allocated, and $Y = \{1, \ldots, m\}$ corresponds to the set of rings in χ. Each arc (i, k), $i \in X$, $k \in Y$ in the arc set A' has an associated *minimum allocation cost* $f(i, k) = \min\{d(i, j) | j \in V(R_k)\}$, which is the minimum cost to allocate customer i to a node in ring R_k.

To obtain the minimum allocation cost $Z_a(\chi)$, we can solve the following mathematical model:

$$Z_a(\chi) = \min \sum_{i \in X} \sum_{k \in Y} f(i, k) \cdot x_{ik} \tag{1}$$

subject to

$$\sum_{k \in Y} x_{ik} = 1, \quad \forall i \in X \tag{2}$$

$$\sum_{i \in X} x_{ik} \le Q - u_k, \quad \forall k \in Y \tag{3}$$

$$x_{ik} \in \{0, 1\} \tag{4}$$

This model can be solved in time polynomial to the size of sets X and Y. The values of the decision variables x_{ik} determine the best allocations of the customers. In this manner, a chromosome $\chi = \{R_1, R_2, \ldots, R_m\}$ can be converted into its corresponding CmRSP solution $S = \{\tilde{R}_1, \tilde{R}_2, \ldots, \tilde{R}_m\}$.

The total routing cost $Z_r(\chi)$ for all rings in χ is computed by

$$Z_r(\chi) = \sum_{k=1}^{m} \left[c\big(0, v_k^1\big) + \sum_{j=2}^{q_i} c\big(v_k^{j-1}, v_i^j\big) + c\big(v_k^{q_i}, 0\big) \right]$$

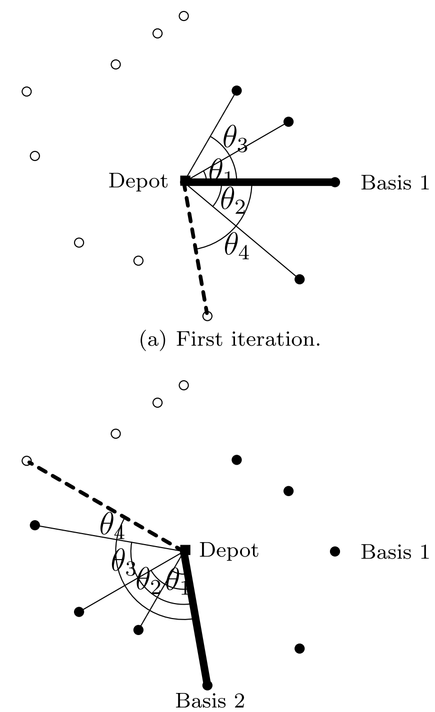
Hence, the total cost of the solution represented by χ is $Z(\chi) = Z_r(\chi) + Z_a(\chi)$. Note that the local refinement phase of our MA employs neighborhood operators that act on the full ring-star solution representation, although all chromosomes are re-optimized by solving the above model at the end of the phase.

3.2 Initial population

One member of our initial population P is generated using the initialization procedure by [23], which was derived from the clustering procedure proposed by [8]. This approach begins with a set of "seed" nodes S that initially contains only

Algorithm 2: Modified sweep algorithm

Randomly select a node j as the basis;
for $k = 1$ **to** m **do**
  foreach $i \in U$ **do**
    if *i is not marked* **then**
      $\cos\theta_i = (c(0,j)^2 + c(0,i)^2 - c(j,i)^2)/(2 \cdot c(0,j) \cdot c(0,i))$;
  $U' \leftarrow$ unmarked nodes in U sorted in decreasing order of $\cos\theta$;
  Label first node as $seed_k$;
  Ring k is set to be $(0, seed_k, 0)$;
  Mark the first $\lceil |U|/m \rceil$ nodes in U';
  Set the $(\lceil |U|/m \rceil + 1)^{th}$ node in U' to be basis node j;
Randomly permute the nodes in U;
foreach $i \in U$ **do**
  if *i is not a seed* **then**
    Assign i to its best position;



(a) First iteration.

(b) Second iteration.

Fig. 2 An example of modified sweep algorithm

the depot node 0. It then repeatedly locates the customer that is furthest from all nodes in S, marks it as a seed customer, and inserts it into S, until m seed customers have been marked. Next, each of the seed customers are connected to the depot to construct m rings. Finally, each remaining non-seed customer is chosen in a random sequence and assigned to its best feasible position (either as a visited or allocated node). The set of rings in this solution is one chromosome in our population. Note that Steiner nodes are not considered in this process.

In order to generate the remaining members of the initial population, we use a *sweep* algorithm similar to the approach by [9] to determine a further $|P| - 1$ sets of seed customers. Assume that the depot is the pole (i.e., the origin of a Cartesian system), and that all customers have Cartesian coordinates corresponding to their locations. Each customer i is represented by its polar coordinate (θ_i, ρ_i), where θ_i is the angle and ρ_i is the ray length. We first sort the customers in increasing order of θ_i, and then choose the customers at index $\alpha, \alpha + \lceil |U|/m \rceil, \alpha + 2 \times \lceil |U|/m \rceil, \ldots, \alpha + (m-1) \times \lceil |U|/m \rceil$ to be seeds, where α is a uniformly randomly selected integer from $[0, \lceil |U|/m \rceil - 1]$. Finally, we use the same process of randomly sequentially assigning the remaining customers to their best positions to generate a chromosome from the seed customers. This is repeated for each chromosome with a newly selected value of α.

When the input instance does not include location information for each customer, we employ the *modified sweep* algorithm given in Algorithm 2 to approximate our sweep algorithm. It begins by randomly selecting a customer j as a basis. Next, we sort the remaining customers by their angle from the virtual line segment from customer j to the depot using the Cosine Law, i.e., $\cos\theta = (a^2 + b^2 - c^2)/(2 \cdot a \cdot b)$, where a and b are the lengths of the adjacent sides and c is the length of the opposite side of the triangle. Note that greater values of $\cos\theta$ correspond to smaller values of angle θ, and θ is unsigned. We label the first node as a seed node and mark the first $\lceil |U|/m \rceil$ nodes; this effectively marks a "wedge" of nodes. An iteration ends when we set the $\lceil |U|/m \rceil^{th}$ node as the basis node for the next iteration. This is repeated m times, considering only the unmarked nodes each time, which produces a set of m seeds.

Figure 2 gives a conceptual example of our modified sweep algorithm, which assumes that the distances between each pair of nodes are their Euclidean distances. Given the basis node selected in Fig. 2(a) and $\lceil |U|/m \rceil = 3$, the algorithm would mark the "wedge" of nodes corresponding to the greatest $\cos\theta$ values, namely θ_1, θ_2 and θ_3. The node corresponding to θ_4 will then be the basis for the next iteration, as shown in Fig. 2(b).

3.3 Crossover

Our crossover operator is motivated by the work done by [6, 20, 33] and constructs two offspring from two parent chromosomes. It is a combination of two-point crossover, feasibility reparation and re-optimization.

Let $\chi_1 = \{R_1^1, R_2^1, \ldots, R_m^1\}$ and $\chi_2 = \{R_1^2, R_2^2, \ldots, R_m^2\}$ be two parent chromosomes. We define the *similarity* between two rings to be the number of common nodes appearing on both rings, and reorder the rings in the chromosomes such that the total similarity between the corresponding rings is maximized. To do so, we find the *maximum weight perfect matching* [27, 32] on the $m \times m$ similarity matrix $\mathbf{A}$ for χ_1 and χ_2, where each element a_{ij} of $\mathbf{A}$ is equal to the similarity of the two rings $R_i^1 \in \chi_1$ and $R_j^2 \in \chi_2$. The similarity between any chromosome pair, such as χ_1 and χ_2, is represented by the sum of weights associated with their maximum weight perfect matching. Without loss of generality, let the resultant (ordered) chromosomes for χ_1 and χ_2 be $p_1 = (R_1^1, R_2^1, \ldots, R_m^1)$ and $p_2 = (R_1^2, R_2^2, \ldots, R_m^2)$, respectively. We then randomly select two cross points x and y, $1 \le x \le y \le m$, and swap the genes in p_1 and p_2 within the range $[x, y]$. Hence, the resultant chromosomes are $o_1' = (R_1^1, \ldots, R_{x-1}^1, R_x^2, \ldots, R_y^2, R_{y+1}^1, \ldots, R_m^1)$ and $o_2' = (R_1^2, \ldots, R_{x-1}^2, R_x^1, \ldots, R_y^1, R_{y+1}^2, \ldots, R_m^2)$, respectively.

Since o_1' or o_2' may contain duplicate nodes, we proceed to repair both chromosomes. For o_1', we simply remove all duplicate nodes in the gene sequences $(R_1^1, \ldots, R_{x-1}^1)$ and $(R_{y+1}^1, \ldots, R_m^1)$, and similarly for o_2'. Finally, we reoptimize both chromosomes to produce the final offspring chromosomes o_1 and o_2, respectively, by considering in a random sequence each node i that is not in the chromosome. The node i is inserted to its best position only if it satisfies one of the following conditions:

- If i is a *customer node*, insert it to its best position if and only if its best position is to visit some ring.
- If i is a *Steiner node*, insert it to its best position if and only if its best position is to visit some ring and the additional cost is less than 0.

Figure 3 provides an example of our crossover operator. Suppose the set of customers $U = \{1, 2, 3, 4, 5\}$ and the set of Steiner nodes $W = \{6, 7\}$. Given the parent chromosomes $p_1 = ((1, 6, 3), (2, 5), (4))$ and $p_2 = ((1, 2, 5), (3, 7), (4))$ as shown in Fig. 3(a), the similarity matrix will be $T = [1\ 1\ 0; 2\ 0\ 0; 0\ 0\ 1]$. The maximum weight matching of T is $\{(1, 2), (2, 1), (3, 3)\}$ with the total weight of 4, resulting in the reordered parent chromosomes in Fig. 3(b). If the cross points are $x = 1$ and $y = 2$, we obtain $o_1' = ((1, 6, 3), (1, 2, 5), (4))$ and $o_2' = ((3, 7), (2, 5), (4))$ (Fig. 3(c)). Since node 1 is duplicated in o_1', we repair it by deleting node 1 from the first ring. For o_2', the best position of customer 1 may be as the second visited node on the first ring, resulting in $o_2 = ((3, 1, 7), (2, 5), (4))$. The final offspring chromosomes are shown in Fig. 3(d).

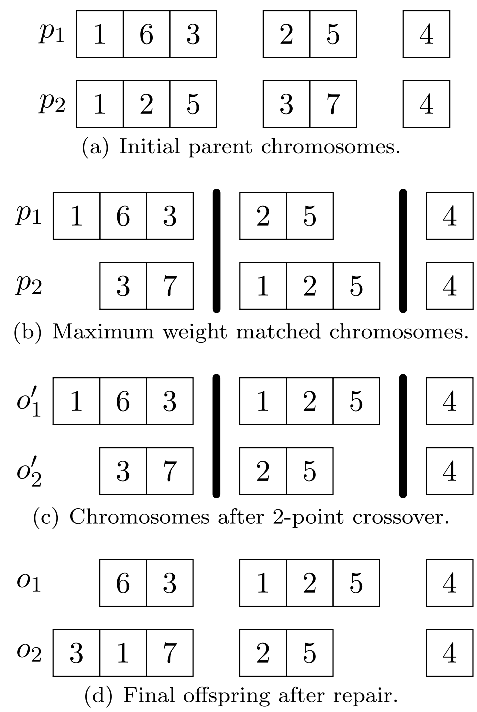


Fig. 3 An example of crossover operator

3.4 Mutation

Our mutation operation works on a CmRSP solution $S = \{\tilde{R}_1, \tilde{R}_2, \ldots, \tilde{R}_m\}$, which is a set of ring-stars, rather than a chromosome. Hence, we first convert a given chromosome $\chi \in P_O$ into a CmRSP solution S using Eqs. (1)–(4). The mutation operation is simple: for each customer in S, we swap it with a randomly selected customer with a *mutation probability* μ. This introduces a small amount of perturbation in the offspring population, which can help the process locate promising neighborhoods in a wider region of the search space.

3.5 Local refinement

After performing the mutation operation, we attempt to locally improve the solution S using a hill-climbing procedure that sequentially applies three neighborhood operators *hc_iters* times, where *hc_iters* is a user-defined parameter (lines 11–14 in Algorithm 1). Over the course of our research, we investigated several complex neighborhood operators with mixed results. After careful experimentation and analysis, we have determined that the three operators described in this section are sufficient to produce high-quality results on our test data.

3.5.1 *Extract-assign operator*

The first operator is called *extract-assign*, which works as follows. Each node $i \in S$ has an *extraction probability* ε of being selected for extraction. First, we remove all selected nodes from their current positions along with any nodes that are allocated to them (if any). Let $\bar{V}$ be the set of all nodes removed in this way. We then reassign all nodes in $\bar{V}$ in a random order to their best positions (either as allocated or visited nodes) in $S - \bar{V}$ by considering all possible positions. If the resultant solution S' is superior to S, then it replaces S; otherwise we discard S'. We perform this process $|U|$ times, where $|U|$ is the number of customers.

Note that the approach by [23] describes an extraction-assignment operator, but it differs from ours in significant ways. In particular, it only considers extracting each node once, and it only considers reassigning the node to the vicinity of its T nearest nodes.

3.5.2 *Steiner insert operator*

Given the current solution $S = \{\tilde{R}_1, \tilde{R}_2, \dots, \tilde{R}_m\}$, the *Steiner insert* operator first randomly selects a ring-star $\tilde{R}_i$. We denote the ring portion of $\tilde{R}_i$ by $R_i = (0, v_i^1, v_i^2, \dots, v_i^{q_i}, 0)$. Next, we randomly select and remove a segment $(v_i^a, v_i^{a+1}, \dots, v_i^b)$, $1 \le a \le b \le q_i$ (i.e., a consecutive sequence of visited nodes) from R_i along with all nodes allocated to this segment; let $\bar{V}$ be the set of all nodes removed in this way. We then insert the best unused Steiner node j in place of the segment such that the cost of the resultant ring-star with ring portion $(0, \dots, v_i^{a-1}, j, v_i^{b+1}, \dots, 0)$ is minimized. Finally, we reassign all nodes in $\bar{V}$ in a random order to the best positions in $S - \bar{V} + \{j\}$. We replace S with the resultant solution S' only if it is superior. We perform this process N_i times, where N_i is the number of unused Steiner nodes in the solution S.

3.5.3 *Steiner remove operator*

The *Steiner remove* operator randomly selects a used Steiner node $j \in S$ and removes it along with all customer nodes connected to j (if any), and then reassigns these customer nodes to their best positions in random order. Once again, the resultant solution S' (with one fewer Steiner node) replaces S only if it is superior, otherwise it is discarded. This is done N_r times, where N_r is the number of used Steiner nodes in the solution S.

Naji-Azimi et al. [23] describes a Steiner-node-removal operator that only considers the reassignment of the allocated customers to their T nearest neighbors. In contrast, our Steiner remove operator considers all possible positions.

3.6 Population composition strategy

Our approach uses an adaptive strategy that takes both solution quality and diversity into account to determine the composition of the chromosome population. This kind of strategy is also widely used in scatter search algorithms [16]. On the one hand, it is important to preserve the best solutions in our population so that their desirable qualities can be transmitted to their offspring and refined over future generations. On the other hand, diversity in the population is also a significant factor when attempting to escape from local optima; our preliminary experiments showed that if two chromosomes in the population are similar, they are more likely to converge to the same solution after local refinement. Hence, the population pool should be carefully considered to achieve a good balance between elite and dissimilar but promising solutions.

Algorithm 3: Population updating procedure

Input: P: population from the previous generation
Input: P_N: new population from the current generation
Output: P: updated population

$P_U \leftarrow$ sort $P \cup P_N$ in increasing order of solution cost;
Let $a = \lceil \lambda \cdot |P| \rceil$;
$P_E \leftarrow$ first a elements of P_U;
Let $P' = P_U - P_E$;
foreach $\chi \in P'$ **do**
    Calculate the similarity between χ and P_E;
Sort P' in increasing order of similarity;
$P_P \leftarrow$ first $|P| - a$ elements of P';
$P \leftarrow P_E \cup P_P$;

Our procedure for updating the population in each generation is given by Algorithm 3, which is invoked in line 17 of Algorithm 1. We divide the population P into two categories: an *elite* set P_E and a *promising* set P_P. The elite set contains the highest quality chromosomes in the entire population in terms of their cost, while the promising set contains the chromosomes that are most "unlike" the elite set. We use the parameter λ to control the sizes of the two sets: the first $a = \lceil \lambda \cdot |P| \rceil$ chromosomes correspond to the elite set and the remaining chromosomes correspond to the promising set.

Given the population P from the previous generation and the new population P_N from the current generation, we sort their union P_U in decreasing order of solution cost. The first a elements of this set is marked as the elite set P_E. For the remaining elements, we calculate their similarity from the elite set and retain the most dissimilar $|P| - a$ chromosomes. Given a population P and a chromosome χ, the similarity between χ and P is defined as the minimum similarity between χ and all the elements in P; we use the same notion of similarity as our crossover operator described in Sect. 3.3.

The parameter λ decreases linearly over the generations. It starts from $\lambda = 1$ and decreases to $\lambda = \lambda_{min}$, where λ_{min} is a user-defined parameter. Let *iter* be the number of the current iteration; the value of λ is calculated as $\lambda = ((num\_gens - iter)/num\_gens) \times (1 - \lambda_{min}) + \lambda_{min}$. This strategy increases the diversity of the population as the number of consecutive non-improving generations increase, which helps the algorithm escape from local optima.

4 Experiments and analysis

Our MA approach was implemented in C++ and compiled by GCC 4.1.2. The experimental results reported in this paper were obtained on a PC with a 2.27 GHz Xeon processor and 4GB of RAM under the Linux operating system. Computation times reported are in CPU seconds on this machine.

We first conducted experiments over four classes of C*m*RSP test instances proposed by [2] and [23]. In [2], the authors generated two classes of 45 instances (Classes A and B) that contain either 26, 51, 76 or 101 nodes based on three TSPLIB instances *eil51, eil76* and *eil101* [28]. The underlying graph of the instances with 26 nodes consists of the first 26 nodes of *eil51*, and other instances were derived from their corresponding TSPLIB instances. Class A instances assume that the routing costs and allocation costs between two nodes i and j are identical and equal to their EUC_2D distance $e(i,j)$, i.e., $c(i,j)=d(i,j)=e(i,j)$; this models a small-scale network where all connections consist mainly of cheaper cables. EUC_2D distance of two nodes is the integer closest to their real Euclidean distance; specifically, half-integers are always rounded to even numbers. Class B instances have $c(i,j)=7\times e(i,j)$ and $d(i,j)=3\times e(i,j)$, which simulates a large-scale network spanning a large geographical area, where the connections in a ring might represent high-quality fiber optic cables while the allocated nodes are connected to the ring using relatively cheaper cables. Recently, [23] generated another two classes (Classes C and D) of 24 instances using two TSPLIB instances *kroA150 and kroA200* with the instance generation procedure of [2], where Classes C and D correspond to Classes A and B, respectively. Due to the rounding issue, some instances in these four classes do not satisfy the triangle inequality. More precisely, there exist three-edge cycles in which the sum of the lengths of two edges is less than the length of the remaining edge by 1 unit in some instances of Classes A and C and by 7 units in some instances of Classes B and D. We refer the reader to [2] for the details of generating these instances.

Next, we randomly generated two new classes of larger test instances (called Classes E and F) for the C*m*RSP. These new instances use a pricing structure for digital data service networks [34], and also address some possible shortcomings of the existing instances. We describe the full details of our new instances along with the computational results obtained by our MA approach on these instances in Sect. 4.4.

All instances as well as their detailed computational results are available in the online supplement to this paper at www.computational-logistics.org/orlib/cmrs.

Table 1 Parameters for MA approach

Symbol	Description
hc_iters	Number of iterations of hill-climbing local refinement procedure
$\|P\|$	Retained population size
$\|P_O\|$	Number of offspring per generation
ε	Extraction rate for extract-assign operator
μ	Mutation rate for mutation operator
λ_{min}	Minimum proportion of elite vs. diverse chromosomes

4.1 Parameter tuning

Our proposed MA approach contains a number of parameters as given in Table 1. In order to determine appropriate values for these parameters, we selected 10 instances from Classes C and D for our preliminary testing, namely *C01, C06, C11, C16, C21, D01, D06, D11, D16* and *D21* (we omit the suffixes in the instance names for the sake of brevity). We then performed a series of experiments where we tested different values for one or two parameters while keeping the others constant. After each experiment, we selected the parameter value(s) that produced the best results and used these values for subsequent experiments. This is a common method of parameter tuning that allows us to find suitable parameter values without over-fitting our algorithm to the data set. For each experiment, we executed our MA approach on each of the 10 instances 5 times with different random seeds, where a time limit of 200 seconds was imposed on each execution. For each instance, we recorded the best value (BV) and calculated the average value (AV) and standard deviation (SD) of the five solution values. The values under the columns "ABV", "AAV" and "ASD" in the following tables in this subsection represent the average values of BV, AV and SD over the 10 selected instances.

Our first experiment aims to determine the number of iterations of hill-climbing local search per generation *hc_iters* from the set $\{1,2,4,8,16\}$; the results are reported in Table 2. The remaining parameters were fixed as: $\varepsilon=0.1$, $\mu=0.1$, $|P|=10$, $|P_O|=20$ and $\lambda_{min}=1$. We eventually decided on *hc_iters* = 8 since this setting could generate the smallest "ABV" and "AAV" and the second smallest "ASD". This table also provides an indication of the effectiveness of our hill-climbing local refinement procedure. Observe that when $hc\_iters=1$, the "AAV" value is 92162.1, which is higher than the other values in the same column, namely the AAV values obtained after the same amount of computation time when $hc\_iters=2,4,8$. This suggests that the inclusion of our local refinement procedure, which is a defining characteristic of a memetic algorithm compared to a genetic algorithm, has a positive effect on algorithm performance.

Our next experiment determines the best combination of population size $|P|$ and the number of offspring $|P_O|$ in our

Table 2 Experiment to find the best value of *hc_iters*

hc_iters	$\|P\|$	$\|P_O\|$	ε	μ	λ	ABV	AAV	ASD
1	10	20	0.1	0.1	1	91853.1	92162.1	224.84
2	10	20	0.1	0.1	1	91290.5	91334.54	49.7
4	10	20	0.1	0.1	1	91286.6	91327.96	33.87
8	10	20	0.1	0.1	1	**91268.6**	**91322.12**	**48.65**
16	10	20	0.1	0.1	1	91274.8	91341.54	50.66

Table 3 Experiment to find the best values of $|P|$ and $|P_O|$

hc_iters	$\|P\|$	$\|P_O\|$	ε	μ	λ	ABV	AAV	ASD
8	5	5	0.1	0.1	1	91263.3	91307.6	54.27
8	5	10	0.1	0.1	1	91268.4	91307.7	39.45
8	**10**	**10**	0.1	0.1	1	**91244.6**	**91318.82**	**63.44**
8	10	20	0.1	0.1	1	91268.6	91322.12	48.65
8	20	10	0.1	0.1	1	91282.7	91363.78	57.75
8	20	20	0.1	0.1	1	91332.1	91385.22	58.36

Table 4 Experiment to find the best values of ε and μ

hc_iters	$\|P\|$	$\|P_O\|$	ε	μ	λ	ABV	AAV	ASD
8	10	10	0.05	0.05	1	91277.9	91253.5	53.88
8	10	10	0.05	0.1	1	91327.6	91414.7	64.37
8	10	10	0.05	0.2	1	91379.8	91465.9	84.35
8	10	10	**0.1**	**0.05**	1	**91232.8**	**91283.1**	**36.67**
8	10	10	0.1	0.1	1	91244.6	91318.82	63.44
8	10	10	0.1	0.2	1	91377.8	91429.66	41.52
8	10	10	0.2	0.05	1	91348.9	91383.82	29.22
8	10	10	0.2	0.1	1	91386.1	91454.48	60.80
8	10	10	0.2	0.2	1	91419.2	91456.22	39.81

parameter setting. The value of $|P_O|$ has a significant effect on the running time of the algorithm since it determines the number of chromosomes that require mutation and local refinement in each generation. A full 3^2 experimental design process was conducted for the values of $|P|$ and $|P_O|$. After some preliminary experiments, the levels of both parameters were taken from $\{5, 10, 20\}$. The results of this experiment are presented in Table 3. We can see that when $|P| = 10$ and $|P_O| = 10$, the algorithm generated the smallest "ABV" with the value 91244.6. However, this setting did not generate the smallest values for "AAV" and "ASD". Since our aim is to find the solution with smallest objective value, the parameter setting resulting the best "ABV" should have the highest priority to be selected. As a result, we fixed the values of parameters $|P|$ and $|P_O|$ at 10.

Subsequently, we tested various combinations of extraction rate ε and mutation rate μ by conducting another full 3^2 experimental design, where the levels of both parameters were taken from $\{0.05, 0.10, 0.20\}$. The results of this experiment are shown in Table 4. Note that a higher extraction rate requires the extract-assign operator to reassign more nodes per operation, increasing the running time of the algorithm. The remaining parameter values remain unchanged, i.e., $hc\_iters = 8$, $|P| = 10$, $|P_O| = 20$, and $\lambda_{min} = 1$. We find that the combination of $\varepsilon = 0.10$ and $\mu = 0.05$ produced the smallest "ABV" and the second smallest "AAV" and "ASD". Thus, we fixed the values of ε and μ at 0.10 and 0.05, respectively.

Finally, we performed an experiment to determine the value of λ_{min}. We tested our approach with $\lambda_{min} = \{0.2, 0.5, 0.8, 1.0\}$ on the selected 10 instances and report the results in Table 5. We find the algorithm with $\lambda_{min} = 0.5$ was able to find the smallest "ABV", "AAV" and "ASD"; this motivated us to fix the value of λ_{min} at 0.5.

In summary, the values for our various parameters were fixed as follows: $hc\_iters = 8$, $\varepsilon = 0.10$, $\mu = 0.05$, $|P| = 10$, $|P_O| = 10$, and $\lambda_{min} = 0.5$. These are the settings used for the remainder of this study. As the sizes of the selected 10 instances are close, we imposed a time limit of 200 seconds on each execution in the parameter tuning stage. However, we need to perform this algorithm to solve all test instances whose sizes are considerably different. So it is not appro-

Table 5 Experiment to find the best value of λ_{min}

hc_iters	$\|P\|$	$\|P_O\|$	ε	μ	λ	ABV	AAV	ASD
8	10	10	0.1	0.05	0.2	91247.8	91290.02	45.21
8	10	10	0.1	0.05	**0.5**	**91219.3**	**91252.4**	**34.87**
8	10	10	0.1	0.05	0.8	91230.9	91288.6	39.01
8	10	10	0.1	0.05	1	91232.8	91283.1	36.67

priate to fix computation time at a constant value. Instead, in the following experiments we fixed the number of generations at 500 for each execution and report the average computation time consumed in the tables in the subsequent subsections.

4.2 Results on Classes A and B

Our MA approach does not assume that the routing costs satisfy the triangle inequality, and therefore it allows non-connecting Steiner nodes. However, we can convert our solution into one that contains no non-connecting Steiner nodes using a "repair" procedure, which simply removes each such Steiner node j and then connects the two nodes adjacent to j. For ease of discussion, we will refer to our MA approach with the repair procedure as *MA+R*.

Tables 6 and 7 show the results obtained by our MA and MA+R approaches compared to the branch-and-cut approach by [2] and two heuristic methods, called HP and NST, by [22, 23] on the Class A and B instances; note that all of these approaches upon which we make our comparisons can only produce solutions that contain no non-connecting Steiner nodes. All computational results of HP and NST were taken from [23] which were obtained on a machine that is superior to ours. However, we believe that there is no dramatic difference between the speeds of these machines, so it is reasonable to directly compare the computational times of the HP, NST and MA+R approaches.

The instance names are listed in column *Instance*. The best costs of solutions without the non-connecting Steiner nodes are highlighted in bold. For the branch-and-cut approach by [2], we give the cost of the solution obtained and the time taken in columns *Cost* and *Time*, respectively; all solutions that were obtained within 7200 seconds are optimal if non-connecting Steiner nodes are not allowed. For the heuristic methods by [22, 23] and our MA+R approach that were all executed 5 times using different random seeds, columns *Best*, *Average* and *A.T.* give for each instance the best cost, the average cost and the average time of the five runs. The last column *Best_nr* gives the cost of the best solutions found by our MA approach, where the values marked with an asterisk (*) indicate solutions containing non-connecting Steiner nodes.

When non-connecting Steiner nodes are not allowed, the branch-and-cut, HP, NST and MA+R approaches achieved the best solutions for 65, 85, 89, and 89 out of the 90 Class A and B instances, respectively. In particular, our MA+R approach found a solution better than all other algorithms for instance *B30* and a solution worse than the best known solution for instance *B22*. The NST and MA+R approaches are both capable of producing the best solutions for almost all instances. However, the running time required by our MA+R approach is much larger than that required by NST. Hence, we suggest that the best existing algorithm for solving the Class A and B instances is the NST heuristic.

The column *Best_nr* shows that our MA approach can further reduce the costs for 31 out of 90 instances when taking non-connecting Steiner nodes into account. Moreover, for each instance the value in the column *Best_nr* is not larger than the value in each of the four *Best* columns. To the best of our knowledge, all algorithms in existing literature assume that these instances abide by the triangle inequality rule. It may be possible to achieve good results on these instances by adapting the NST heuristic to allow non-connecting Steiner nodes, failing which our MA approach could be employed instead.

4.3 Results on Classes C and D

For each of the Class C and D instances, the HP and NST heuristics were executed 20 times while our MA+R approach was only executed 10 times. The superiority of our MA approach over the branch-and-cut, HP and NST approaches is clearly exhibited by the results shown in Tables 8 and 9: these four approaches achieved the best solutions that contain no non-connecting Steiner nodes for 6, 16, 24 and 42 out of the 48 Class C and D instances, respectively.

Our MA+R approach achieved better solutions than the other approaches for 23 instances: 5 Class C instances (*C12, C21–C24*) and 18 Class D instances (*D01, D04, D06–D13, D16–D17, D19–D24*). This shows that MA+R has a greater advantage over existing algorithms when solving the Class D instances. The NST heuristic found better solutions than the other approaches for only three instances (*C17, D15, D18*); HP found such a solution for only instance *C18*; and the branch-and-cut approach found no such solution. The high quality of our MA+R solutions came at a cost of added computational time. From the last rows of Tables 8 and 9, we can see that on average our MA+R approach consumed about 4 to 10 times the computational time required by the

Table 6 Computational results for Class A instances

Instance	Branch-and-cut		HP			NST			MA + R				MA
	Cost	Time	Best	Average	A.T.	Best	Average	A.T.	Best	Average	s	A.T.	Best_nr
A01	**242**	0.10	**242**	242.0	0.10	**242**	242.0	0.18	**242**	242.0	0	15.63	242
A02	**261**	0.00	**261**	261.0	0.10	**261**	261.0	0.16	**261**	261.0	0	17.69	261
A03	**292**	0.00	**292**	292.0	0.08	**292**	292.0	0.14	**292**	292.0	0	25.97	292
A04	**301**	0.50	**301**	301.0	0.16	**301**	301.0	0.24	**301**	301.0	0	11.28	301
A05	**339**	0.30	**339**	339.0	0.20	**339**	339.0	0.38	**339**	339.0	0	13.75	339
A06	**375**	0.70	**375**	375.0	0.32	**375**	375.0	0.26	**375**	375.0	0	25.58	375
A07	**325**	3.80	**325**	325.0	0.30	**325**	325.0	0.28	**325**	325.0	0	0.84	325
A08	**362**	0.30	**362**	362.0	0.24	**362**	362.0	0.22	**362**	362.0	0	0.87	362
A09	**382**	0.20	**382**	382.0	0.46	**382**	382.0	0.40	**382**	382.0	0	0.87	382
A10	**242**	0.20	**242**	242.0	0.12	**242**	242.0	0.14	**242**	242.0	0	52.29	242
A11	**261**	0.40	**261**	261.0	0.14	**261**	261.0	0.14	**261**	261.0	0	38.45	261
A12	**286**	0.10	**286**	286.0	0.12	**286**	286.0	0.12	**286**	286.0	0	39.52	286
A13	**322**	2.10	**322**	322.0	0.34	**322**	322.0	0.40	**322**	322.0	0	86.58	322
A14	**360**	2.10	**360**	360.0	0.38	**360**	360.0	0.44	**360**	360.0	0	71.56	360
A15	**379**	2.30	**379**	379.0	0.48	**379**	379.0	0.52	**379**	379.0	0	65.04	379
A16	**373**	8.40	**373**	373.0	0.64	**373**	373.0	0.50	**373**	373.0	0	37.81	373
A17	**405**	41.70	**405**	405.0	0.70	**405**	405.0	0.72	**405**	405.0	0	54.07	404*
A18	**432**	52.20	**432**	432.8	0.76	**432**	432.0	0.94	**432**	432.0	0	45.90	432
A19	**458**	182.80	**458**	458.2	1.00	**458**	458.0	1.62	**458**	458.0	0	4.13	458
A20	**490**	220.40	**490**	490.0	1.08	**490**	490.0	1.44	**490**	490.0	0	4.24	490
A21	**520**	6334.20	**520**	520.8	1.24	**520**	520.8	1.64	**520**	520.1	0.3	4.42	520
A22	**330**	48.30	**330**	330.0	0.34	**330**	330.0	0.26	**330**	330.3	0.5	133.53	328*
A23	**385**	30.60	**385**	385.0	0.32	**385**	385.0	0.14	**385**	385.0	0	161.55	383*
A24	**448**	63.70	**448**	448.0	0.50	**448**	448.0	0.42	**448**	448.0	0	169.51	445*
A25	**402**	567.70	**402**	402.0	0.92	**402**	402.0	0.94	**402**	402.0	0	90.95	401*
A26	460	7200.00	**457**	457.8	0.96	**457**	458.0	1.12	**457**	457.3	0	85.03	456*
A27	**479**	509.30	**479**	479.0	1.04	**479**	479.0	1.28	**479**	479.0	0	78.51	477*
A28	**471**	1584.40	**471**	471.0	1.60	**471**	471.0	2.88	**471**	471.0	0	51.66	470*
A29	523	7200.00	**519**	519.8	1.60	**519**	519.6	1.96	**519**	519.0	0	59.45	518*
A30	**545**	3221.30	**545**	548.0	1.76	**545**	547.4	2.30	**545**	545.0	0	58.64	544*
A31	**564**	479.50	**564**	565.0	2.50	**564**	566.2	3.76	**564**	564.2	0.4	12.29	564
A32	606	7200.00	**602**	604.2	2.40	**602**	602.5	4.72	**602**	602.4	0.8	12.12	602
A33	654	7200.00	**640**	648.8	2.50	**640**	642.0	6.60	**640**	640.0	0	10.99	640
A34	**363**	8.70	**363**	363.0	0.58	**363**	363.0	0.40	**363**	363.0	0	147.93	361*
A35	**415**	91.80	**415**	415.0	0.60	**415**	415.0	0.32	**415**	415.0	0	152.45	414*
A36	**448**	680.40	**448**	448.0	0.90	**448**	448.0	0.98	**448**	448.2	0.4	176.61	446*
A37	**500**	7200.00	**500**	500.0	1.40	**500**	500.0	1.48	**500**	500.0	0	113.15	499*
A38	532	7200.00	**528**	528.0	1.66	**528**	528.0	2.10	**528**	528.3	0.5	87.96	528
A39	568	7200.00	**567**	567.0	1.54	**567**	567.0	1.76	**567**	567.0	0	122.78	566*
A40	**595**	6690.10	**595**	595.0	2.86	**595**	595.2	4.50	**595**	595.0	0	83.63	595
A41	625	7200.00	**623**	623.2	3.16	**623**	623.6	6.42	**623**	623.4	0.5	92.69	623
A42	662	7200.00	**657**	658.6	2.74	**657**	657.8	4.80	**657**	657.3	0.5	87.09	657
A43	**646**	283.00	648	651.0	5.30	**646**	649.8	10.52	**646**	646.8	1.5	23.26	646
A44	680	7200.00	**679**	680.2	5.18	**679**	679.8	10.06	**679**	679.6	0.7	26.49	679
A45	**700**	1310.80	**700**	700.0	4.76	**700**	700.4	10.18	**700**	700.0	0	26.14	700

Table 7 Computational results for Class B instances

Instance	Branch-and-cut		HP			NST			MA + R				MA
	Cost	Time	Best	Average	A.T.	Best	Average	A.T.	Best	Average	s	A.T.	Best_nr
B01	**1684**	0.10	**1684**	1684.0	0.12	**1684**	1684.0	0.06	**1684**	1684.0	0	17.02	1684
B02	**1827**	0.10	**1827**	1827.0	0.10	**1827**	1827.0	0.16	**1827**	1827.0	0	24.10	1827
B03	**2041**	0.00	**2041**	2041.0	0.10	**2041**	2041.0	0.12	**2041**	2041.0	0	39.27	2041
B04	**2104**	0.50	**2104**	2104.0	0.18	**2104**	2104.0	0.22	**2104**	2104.0	0	23.33	2104
B05	**2370**	0.50	**2370**	2370.0	0.26	**2370**	2370.0	0.40	**2370**	2370.0	0	31.35	2370
B06	**2615**	0.70	**2615**	2615.0	0.46	**2615**	2615.0	0.24	**2615**	2615.0	0	28.56	2615
B07	**2251**	0.40	**2251**	2251.0	0.24	**2251**	2251.0	0.48	**2251**	2251.0	0	0.85	2251
B08	**2510**	0.50	**2510**	2510.0	0.28	**2510**	2510.0	0.28	**2510**	2510.0	0	0.87	2510
B09	**2674**	0.80	**2674**	2674.0	0.38	**2674**	2674.0	0.42	**2674**	2674.0	0	0.88	2674
B10	**1681**	0.80	**1681**	1681.0	0.18	**1681**	1681.0	0.14	**1681**	1681.0	0	89.30	1674*
B11	**1821**	1.50	**1821**	1821.0	0.18	**1821**	1821.0	0.16	**1821**	1821.0	0	71.54	1821
B12	**1972**	0.30	**1972**	1972.0	0.20	**1972**	1972.0	0.18	**1972**	1972.0	0	79.51	1972
B13	**2176**	1.10	**2176**	2176.0	0.36	**2176**	2176.0	0.30	**2176**	2176.0	0	92.55	2176
B14	**2470**	7.20	**2470**	2470.0	0.42	**2470**	2470.0	0.42	**2470**	2470.7	2.2	90.62	2470
B15	**2579**	4.10	**2579**	2579.0	0.52	**2579**	2579.0	0.52	**2579**	2579.0	0	95.26	2579
B16	**2490**	17.90	**2490**	2490.0	0.84	**2490**	2496.8	0.58	**2490**	2490.0	0	64.30	2490
B17	**2721**	74.90	**2721**	2721.0	0.78	**2721**	2721.0	0.82	**2721**	2721.0	0	78.48	2714*
B18	**2908**	145.00	**2908**	2914.6	0.96	**2908**	2908.0	0.98	**2908**	2908.0	0	49.67	2908
B19	**3015**	296.70	**3015**	3015.0	1.80	**3015**	3015.0	1.72	**3015**	3015.0	0	4.39	3015
B20	**3260**	336.60	**3260**	3260.0	1.68	**3260**	3260.0	1.40	**3260**	3260.0	0	4.52	3260
B21	**3404**	6470.70	**3404**	3404.0	1.82	**3404**	3420.6	2.22	**3404**	3404.0	0	4.47	3404
B22	**2253**	105.50	**2253**	2253.0	0.44	**2253**	2256.6	0.36	2259	2259.0	0	214.12	2245*
B23	**2620**	29.50	**2620**	2620.0	0.42	**2620**	2620.0	0.24	**2620**	2620.0	0	194.12	2613*
B24	**3059**	85.30	**3059**	3059.0	0.48	**3059**	3059.0	0.38	**3059**	3060.3	4.1	212.98	3045*
B25	**2720**	1897.60	**2720**	2720.0	0.98	**2720**	2720.0	1.06	**2720**	2720.0	0	159.74	2713*
B26	3138	7200.00	**3100**	3115.2	1.36	**3100**	3113.8	1.34	**3100**	3110.6	9.4	108.23	3093*
B27	3311	7200.00	**3284**	3284.0	1.16	**3284**	3284.0	1.06	**3284**	3284.0	0	112.39	3277*
B28	3088	7200.00	**3044**	3060.0	2.96	**3044**	3049.4	2.82	**3044**	3044.0	0	80.09	3044
B29	3447	7200.00	**3415**	3438.6	3.24	**3415**	3440.8	2.40	**3415**	3422.1	8	51.26	3408*
B30	3648	7200.00	3636	3642.2	3.00	3632	3643.2	3.04	**3631**	3631.2	0.4	47.40	3624*
B31	3740	7200.00	**3652**	3687.2	5.38	**3652**	3670.2	5.46	**3652**	3652.0	0	12.60	3652
B32	4026	7200.00	4003	4006.4	4.78	**3964**	4002.8	6.32	**3964**	3982.7	18.4	12.41	3964
B33	4288	7200.00	**4217**	4217.0	4.70	**4217**	4217.0	6.28	**4217**	4217.0	0	11.79	4217
B34	**2434**	24.20	**2434**	2434.0	0.70	**2434**	2434.0	0.46	**2434**	2434.0	0	255.19	2427*
B35	**2782**	115.40	**2782**	2782.0	0.68	**2782**	2782.0	0.28	**2782**	2782.0	0	240.73	2775*
B36	**3009**	862.40	**3009**	3009.0	1.02	**3009**	3009.0	1.04	**3009**	3011.8	3.6	270.49	3002*
B37	3332	7200.00	**3322**	3322.0	1.88	**3322**	3322.0	1.98	**3322**	3322.0	0	172.60	3315*
B38	**3533**	7200.00	**3533**	3533.0	2.04	**3533**	3533.0	2.14	**3533**	3535.0	6.3	156.15	3526*
B39	3872	7200.00	**3834**	3839.6	2.42	**3834**	3839.2	2.76	**3834**	3839.2	4.8	181.90	3827*
B40	3923	7200.00	**3887**	3887.8	5.18	**3887**	3888.0	7.12	**3887**	3891.2	8.2	128.97	3887
B41	4125	7200.00	**4082**	4088.4	5.04	**4082**	4091.4	6.20	**4082**	4085.2	10.1	134.33	4082
B42	4458	7200.00	**4358**	4358.0	4.38	**4358**	4358.0	7.04	**4358**	4358.0	0	121.10	4358
B43	4110	7200.00	4135	4150.4	14.94	**4110**	4126.0	10.40	**4109**	4109.6	0.5	24.23	4109
B44	4506	7200.00	4358	4377.6	11.28	**4355**	4379.8	11.60	**4355**	4374.5	10.5	26.36	4355
B45	4632	7200.00	**4565**	4568.4	10.08	**4565**	4566.4	9.28	**4565**	4565.0	0	27.77	4565

Table 8 Computational results for Class C instances

Instance	Branch-and-cut		HP			NST			MA + R				MA
	Cost	Time	Best	Average	A.T.	Best	Average	A.T.	Best	Average	s	A.T.	Best_nr
C01	17163	7200.00	**17138**	17138.0	2.27	**17138**	17138.0	1.06	**17138**	17138.0	0	50.83	17138
C02	**18782**	7200.00	**18782**	18782.0	2.28	**18782**	18782.0	0.87	**18782**	18782.0	0	138.20	18782
C03	**20135**	6534.50	**20135**	20186.3	3.20	**20135**	20237.6	1.49	**20135**	20135.0	0	66.38	20134*
C04	**20741**	7200.00	**20741**	20741.0	6.72	**20741**	20741.0	6.43	**20741**	20741.0	0	150.99	20741
C05	22810	7200.00	**22525**	22566.3	7.34	**22525**	22525.0	9.82	**22525**	22525.0	0	93.99	22525
C06	24955	7200.00	**24949**	24954.5	6.96	**24949**	24953.2	6.62	**24949**	24949.0	0	168.17	24948*
C07	**23259**	2914.70	**23259**	23259.0	14.65	**23259**	23314.8	15.01	**23259**	23259.0	0	81.91	23258*
C08	25121	7200.00	**25006**	25006.0	12.31	**25006**	25006.0	14.29	**25006**	25006.0	0	155.51	25005*
C09	27605	7200.00	**27277**	27288.8	13.43	**27277**	27284.2	17.37	**27277**	27277.0	0	85.58	27276*
C10	27250	7200.00	**27233**	27312.0	31.25	**27273**	27326.6	36.26	**27223**	27268.2	15.9	139.41	27223
C11	**28536**	2400.20	**28536**	28573.6	30.68	**28536**	28551.6	30.12	**28536**	28545.7	13	90.75	28536
C12	31286	7200.00	30811	30857.2	28.16	30669	30833.0	38.00	**30667**	30768.9	143.1	115.20	30667
C13	18614	7200.00	**18567**	18567.0	5.56	**18567**	18567.0	3.23	**18567**	18567.0	0	122.81	18567
C14	20834	7200.00	**20650**	20687.5	6.69	**20650**	20709.3	4.70	**20650**	20650.0	0	135.16	20650
C15	23510	7200.00	**23496**	23502.3	7.30	**23496**	23501.7	5.23	23503	23506.4	1.9	107.16	23503
C16	22919	7200.00	**22882**	22882.0	15.57	**22882**	22882.0	26.25	**22882**	22882.0	0	137.57	22882
C17	25660	7200.00	25485	25627.2	17.05	**25472**	25559.4	27.62	25473	25481.5	6.6	104.33	25473
C18	28413	7200.00	**28300**	28365.2	18.36	28333	28364.7	24.55	28334	28348.0	5.1	152.52	28334
C19	27325	7200.00	26987	27054.7	38.34	**26971**	26999.9	62.04	**26971**	26982.6	16.4	107.97	26970*
C20	29778	7200.00	29333	29649.5	36.89	**29268**	29586.9	75.16	**29268**	29341.0	99.3	126.31	29267*
C21	32243	7200.00	31944	32054.9	35.33	31946	31993.7	73.16	**31915**	31927.3	4.7	119.25	31914*
C22	30462	7200.00	30256	30591.8	67.27	30181	30351.0	130.69	**30010**	30165.4	121	136.34	30010
C23	32463	7200.00	32233	32404.2	64.26	32152	32362.3	109.76	**32074**	32181.5	118.4	115.91	32074
C24	34969	7200.00	34502	34590.2	56.80	34455	34524.2	109.58	**34427**	34477.1	35.5	143.21	34427
Average		6793.73			22.03			34.55				118.56	

NST heuristic for these two classes of instances. This is reasonable since it is easy for the NST heuristic to get trapped in local optima, which makes it terminate quickly. Our MA+R approach explores a broader solution region, so it requires a longer computational time.

The construction of a telecommunication network commonly involves a large amount of monetary investment, and the optimization of the network design must usually be completed prior to construction. Hence, telecommunication companies are willing to take a large amount of time in order to find the most cost-efficient network design possible, e.g., as much as hundreds of hours, so computational time is not a critical factor for the CmRSP. In this practical context, it is reasonable to claim that the MA+R approach outperformed the other three approaches when solving these two classes of instances.

From the last columns of these two tables, we also find that the MA approach produced better solutions for 19 out of 48 instances than the MA+R approach. Since the NST heuristic is inferior to our MA+R approach, we do not expect that its modification that allows non-connecting Steiner nodes would be able to find better solutions than our MA approach for the Class C and D instances. Thus, we can conclude that our MA approach is the best choice for larger instances of the CmRSP at the time of this writing.

4.4 New instances and results

An inspection of the results in Tables 6, 7, 8, and 9 leads us to two hypotheses. Firstly, the large number of instances where both NST and MA found solutions with identical costs suggest that most of the Class A and B instances are either optimally or close to optimally solved. Secondly, our MA approach might scale better than the existing heuristic methods as the number of nodes and customers increase.

In order to address these possibilities, we generated two new classes of larger test instances to supplement the existing instances for the CmRSP. The underlying graphs for our instances are also obtained from TSPLIB, namely the *gr202* and *gr431* instances containing $n = 202$ and $n = 431$ nodes, respectively. The first node is selected as the depot. We then randomly ordered the remaining nodes, set the

Table 9 Computational results for Class D instances

Instance	Branch-and-cut		HP			NST			MA + R				MA
	Cost	Time	Best	Average	A.T.	Best	Average	A.T.	Best	Average	s	A.T.	Best_nr
D01	110350	3193.50	111185	111360.8	3.50	110607	110618.0	1.06	**110350**	110350.0	0	61.12	110343*
D02	121569	7200.00	122415	122855.0	4.47	**122066**	123211.9	1.17	**122066**	122271.0	171.2	132.38	122066
D03	**129540**	7200.00	129882	130224.7	5.43	**129540**	130045.9	2.11	129840	130402.5	302.6	68.02	129833*
D04	130349	7200.00	130117	130832.9	11.30	28736	130106.5	8.14	**128475**	128475.0	0	151.83	128475
D05	144646	7200.00	142675	142922.1	11.96	**141680**	141796.8	7.65	**141680**	141680.0	0	71.71	141673*
D06	161128	7200.00	160988	161570.0	10.95	159938	161178.4	5.91	**159690**	159884.8	101.9	135.97	159683*
D07	144756	7200.00	146479	147509.6	28.67	145257	145787.0	16.59	**144519**	144519.0	0	136.16	144512*
D08	159197	7200.00	159368	159951.2	24.56	157193	158128.8	15.27	**156085**	156085.0	0	183.03	156078*
D09	179727	7200.00	178790	179511.8	21.12	176635	177704.4	12.76	**172722**	172722.0	0	157.37	172715*
D10	163932	7200.00	167825	169422.2	66.67	164864	167790.3	18.86	**162539**	162548.6	12.4	197.26	162539
D11	174667	7200.00	175777	176683.3	51.63	172716	175427.1	24.37	**171957**	171957.0	0	152.55	171957
D12	195838	7200.00	196133	197087.5	45.17	192298	194374.9	21.45	**190646**	191027.2	317.8	220.69	190646
D13	120704	7200.00	121684	121812.2	7.90	120913	121306.1	3.82	**120527**	120557.7	61.1	233.46	120527
D14	134630	7200.00	135276	135754.8	8.59	**134215**	134947.7	5.36	**134215**	134225.7	16.8	308.29	134215
D15	151439	7200.00	152012	152779.4	9.79	**151125**	151772.2	5.18	152306	152399.0	49.1	256.75	152306
D16	145308	7200.00	145241	145764.5	29.40	144895	145513.4	25.84	**144813**	144816.1	9.8	333.51	144813
D17	163581	7200.00	163935	165012.7	27.83	162363	164571.5	24.40	**162352**	162582.8	164.9	253.25	162345*
D18	183284	7200.00	183190	185295.2	26.97	**181182**	184612.6	26.79	182181	183559.5	766.9	355.18	182181
D19	165666	7200.00	165878	166905.5	96.14	164306	165591.6	53.55	**164243**	164289.5	17.1	273.57	164236*
D20	185886	7200.00	185855	188008.5	71.09	182707	185268.9	50.08	**182092**	182244.3	417.3	356.89	182085*
D21	201848	7200.00	203294	204684.2	62.93	201134	203505.0	59.69	**199760**	200021.6	158.8	287.72	199753*
D22	183547	7200.00	186031	188502.5	186.28	181049	183388.3	100.35	**180156**	180662.7	475.9	384.66	180156
D23	199621	7200.00	202332	205043.7	146.23	197673	201010.7	116.07	**196546**	197030.8	445.6	292.78	196546
D24	218610	7200.00	221917	223632.5	100.00	216993	220696.7	63.28	**214576**	214918.1	256.8	431.83	214576
Average		7033.06			44.11			27.90				226.50	

first $\lfloor \alpha(n-1) \rfloor$ nodes where $\alpha \in \{0.5, 0.7, 0.9\}$ to be customers, and set the remaining nodes to be Steiner nodes. The number of ring-stars for each instance is set to be $m \in \{3, 5, 7\}$. The capacity of each ring-star is given the value $Q = \lceil |U|/(0.9m) \rceil$, which implies that each ring-star can handle about 90 % of all customers; this is the method used by [2] to determine the ring-star capacities.

To determine the routing costs for each pair of nodes, we make use of the price structure for digital data service networks proposed by [34]. We first scale the graph distances such that all nodes lie in a 20 mile $\times$ 20 mile square. Let $e(i, j)$ be the Euclidean distance between nodes i and j in this square. The routing cost $c(i, j)$ is calculated by:

$$c(i, j) = \begin{cases} 30 & e(i, j) < 1 \\ 125 + \lceil e(i, j) \times 1.2 \rceil & 1 \leq e(i, j) \leq 15 \\ 130 + \lceil e(i, j) \times 1.5 \rceil & e(i, j) > 15 \end{cases}$$

We separated our new instances into two classes (Classes E and F) with different allocation costs in a similar manner to the Class A–D instances. For the Class E instances, the routing and allocation costs are identical, i.e., $d(i, j) = c(i, j)$. For the Class F instances, the ratio of routing and allocation costs is 7:3, i.e., $d(i, j) = \lceil \frac{3}{7} \times c(i, j) \rceil$. For both classes, the set of possible edges for allocation C_i is found by taking the average allocation cost $AC = \sum_{(i,j) \in S} d(i, j)/|A|$ and choosing the cheapest 40 % of these edges, i.e., $C_i = \{j : d(i, j) \leq 0.4 \times AC\}$. For each class, we randomly generated a single instance for each combination of α and m for each graph, resulting in 18 instances per class (36 instances in total). These newly generated instances do not necessarily satisfy the triangle inequality.

We set a time limit of 1800 seconds for the instances with 202 nodes, and 3600 seconds for the instances with 431 nodes. Once again, we executed our approach 10 times for each new instance using different random seeds. The results obtained by our MA+R and MA approaches on these new instances are given in Table 10, where column *NNS* gives the number of the non-connecting Steiner nodes in the best MA solutions.

Table 10 Computational results for Class E and F instances

Instance	m	$\|U\|$	Q	MA+R					MA
				Best	NNS	Average	s	A.T.	Best_nr
E01	3	101	38	4958	8	4978.1	40	226.86	4430*
E02	3	141	53	6075	5	6148.3	75.3	219.04	5744*
E03	3	181	68	7285	2	7306.6	60	212.53	7153*
E04	5	101	38	5151	8	5193.4	33.2	223.79	4621*
E05	5	141	53	6324	6	6406.4	96.6	203.79	5864*
E06	5	181	68	7438	2	7440.2	1.5	227.02	7243*
E07	7	101	38	5400	9	5564.1	132.6	259.40	4806*
E08	7	141	53	6683	9	6745.6	60.3	235.45	6088*
E09	7	181	68	7507	2	7592.8	70.0	233.13	7375*
E10	3	215	80	9016	3	9158.9	113.4	618.98	8756*
E11	3	301	112	11889	3	12035.3	116.1	930.41	11688*
E12	3	387	144	14722	2	14784.1	66.8	1375.93	14590*
E13	5	215	48	9507	8	9655.6	83.1	1503.86	8977*
E14	5	301	67	11861	1	12140.9	154.7	1312.76	11753*
E15	5	387	86	14775	3	14895.4	74	1563.27	14577*
E16	7	215	35	9864	10	9926.8	73.9	1651.23	9203*
E17	7	301	48	12276	3	12507.4	124.7	1465.22	12078*
E18	7	387	62	14844	0	14986.6	98.9	1579.86	14788
F01	3	101	38	3578	2	3674.6	75.9	205.60	3445*
F02	3	141	53	4404	0	4507.3	75.9	221.32	4327
F03	3	181	68	5111	0	5206.2	69.4	163.41	5111
F04	5	101	38	3888	2	3954.8	77.0	161.61	3727*
F05	5	141	53	4619	0	4751.4	56.3	232.67	4584
F06	5	181	68	5372	0	5460.4	63.8	177.97	5372
F07	7	101	38	4084	1	4203.8	112.3	173.34	4008*
F08	7	141	53	4855	0	5005.1	105.9	227.80	4835
F09	7	181	68	5713	0	5779.4	49.0	175.60	5713
F10	3	215	80	6084	1	6184.2	73.0	424.46	6018*
F11	3	301	112	7623	0	7686.7	44.7	559.34	7623
F12	3	387	144	8980	0	9007.2	25.1	912.13	8980
F13	5	215	48	6349	0	6458.7	66.3	1046.94	6260
F14	5	301	67	7927	1	7963.2	26.0	853.66	7861*
F15	5	387	86	9218	0	9288.4	30.8	1054.87	9188
F16	7	215	35	6701	1	6792.0	59.5	1044.27	6604*
F17	7	301	48	8255	0	8358.6	59.6	836.66	8213
F18	7	387	62	9627	0	9706.1	44.3	920.20	9627

We feel that our new instances model a practical scenario that employs a digital data service network cost structure and involves more customers than the existing benchmark instances. These new instances, along with the results obtained by our MA approach, can serve as additional benchmarks for future researchers.

5 Conclusion

When non-connecting Steiner nodes are disallowed, our memetic algorithm with a simple repair procedure obtained the best known solutions for 131 out of 138 existing benchmark instances, improving on the previously best known solutions for 24 of them. It is possible that previous researchers worked under the mistaken assumption that the routing costs of the instances satisfy the triangle inequality, which adversely affected the quality of their solutions. Nonetheless, our memetic algorithm generated the best known solutions for the CmRSP under its proper definition, which does allow non-connecting Steiner nodes. We also contributed an additional 36 larger instances with routing costs based on the pricing scheme for a digital data service network, which depicts a different practical scenario from the existing instances. This new data set supplements the existing data, enabling future researchers to more thoroughly test their approaches empirically.

There is a variant of the CmRSP that deserves further study. Currently, there is no limit on the number of nodes that can be allocated to a single connecting node. However, having several nodes allocated to a single connecting node defeats the purpose of the ring-star topology, since all of these nodes will be affected if the connecting node fails. Furthermore, Steiner nodes often represent way stations owned by the telecommunications company that are not only more easily maintained by technicians, they also tend to employ more robust hardware than those found at customer locations. Hence, more nodes can be safely allocated to Steiner nodes than customer nodes. These factors can be modeled by limiting the number of nodes that can be allocated to a customer node and a Steiner node to some values a_c and a_s, respectively.

Acknowledgements This research was partially supported by the Fundamental Research Funds for the Central Universities, HUST (Grant No. 2012QN213) and National Natural Science Foundation of China (Grant No. 71201065).

References

1. Baldacci R, Dell'Amico M (2010) Heuristic algorithms for the multi-depot ring-star problem. Eur J Oper Res 203(1):270–281
2. Baldacci R, Dell'Amico M, Salazar González J (2007) The capacitated m-ring-star problem. Oper Res 55(6):1147–1162
3. Beasley JE, Nascimento EM (1996) The vehicle routing-allocation problem: a unifying framework. Top 4(1):65–86
4. Burkard R, Dell'Amico M, Martello S (2008) Assignment problems. Society for Industrial and Applied Mathematics, 3600 Market Street, 6th floor, Philadelphia, PA, USA (2008)
5. Current JR, Schilling DA (1994) The median tour and maximal covering tour problems: formulations and heuristics. Eur J Oper Res 73(1):114–126
6. Falkenauer E (1998) Genetic algorithms and grouping problems. Wiley, Chichester
7. Fallahi AE, Prins C, Wolfler Calvo R (2008) A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem. Comput Oper Res 35(5):1725–1741
8. Fischetti M, Salazar González JJ, Toth P (1997) A branch-and-cut algorithm for the symmetric generalized travelling salesman problem. Oper Res 45(3):378–394
9. Gillett BE, Miller LR (1974) A heuristic algorithm for the vehicle-dispatch problem. Oper Res 22(2):340–349
10. Grötschel M, Monma CL, Stoer M (1995) Design of survivable networks. In: Handbooks in operations research and management science, vol 7. Elsevier, Amsterdam, pp 617–672
11. Hoshino EA, de Souza CC (2008) Column generation algorithms for the capacitated m-ring-star problem. In: Computing and combinatorics. Lecture notes in computer science, pp 631–641
12. Hoshino EA, de Souza CC (2010) A branch-and-cut-and-price approach for the capacitated m-ring-star problem. Technical report IC-10-15, Institute of Computing, University of Campinas, Brazil
13. Labbé M, Laporte G, Rodríguez Martín I, Salazar González JJ (2004) The ring star problem: polyhedral analysis and exact algorithm. Networks 43(3):177–189
14. Labbé M, Laporte G, Rodríguez Martín I, Salazar González JJ (2005) Locating median cycles in networks. Eur J Oper Res 160(2):457–470
15. Labbé M, Rodríguez Martín I, Salazar-González JJ (2004) A branch-and-cut algorithm for the plant-cycle location problem. J Oper Res Soc 55(5):513–520
16. Laguna M, Martí R (2003) Scatter search—methodology and implementations in C. Operations research/computer science interfaces series, vol 24. Kluwer Academic, Boston
17. Liefooghe A, Jourdan L, Talbi EG (2010) Metaheuristics and cooperative approaches for the bi-objective ring star problem. Comput Oper Res 37(6):1033–1044
18. Lü Z, Hao JK (2010) A memetic algorithm for graph coloring. Eur J Oper Res 203(1):241–250
19. Mauttone A, Nesmachnow S, Olivera A, Robledo F (2007) A hybrid metaheuristic algorithm to solve the capacitated m-ring star problem. In: International network optimization conference
20. Mei Y, Tang K, Yao X (2009) A global repair operator for capacitated arc routing problem. IEEE Trans Syst Man Cybern, Part B, Cybern 39(3):723–734
21. Nagy G, Salhi S (2007) Location-routing: Issues, models and methods. Eur J Oper Res 177(2):649–672
22. Naji-Azimi Z, Salari M, Toth P (2010) A heuristic procedure for the capacitated m-ring-star problem. Eur J Oper Res 207(3):1227–1234
23. Naji-Azimi Z, Salari M, Toth P (2012) An integer linear programming based heuristic for the capacitated m-ring-star problem. Eur J Oper Res 217(1):17–25
24. Neri F, Cotta C, Moscato P (2012) Handbook of memetic algorithms, studies in computational intelligence, vol 379. Springer, Berlin
25. Ngueveu SU, Prins C, Wolfler Calvo R (2010) An effective memetic algorithm for the cumulative capacitated vehicle routing problem. Comput Oper Res 37(11):1877–1885
26. Ombuki BM, Ventresca M (2004) Local search genetic algorithms for the job shop scheduling problem. Appl Intell 21(1):99–109
27. Osiakwan CNK, Akl SG (1990) The maximum weight perfect matching problem for complete weighted graphs is in PC. In: Proceedings of the second IEEE symposium on parallel and distributed processing, pp 880–887
28. Reinelt G (1991) TSPLIB—a traveling salesman problem library. ORSA J Comput 3:376–384
29. Resende M, Ribeiro C (2003) Greedy randomized adaptive search procedures. In: Glover F, Kochenberger G (eds) Handbook of metaheuristics. Kluwer Academic, Boston, pp 219–249
30. Revelle CS, Laporte G (1996) The plant location problem: new models and research prospects. Oper Res 44(6):864–874
31. Soak SM, Lee SW (2012) A memetic algorithm for the quadratic multiple container packing problem. Appl Intell 36(1):119–135
32. West DB (2001) Introduction to graph theory. Prentice Hall, Upper Saddle River
33. Wilke P, Gröbner M, Oster N (2002) A hybrid genetic algorithm for school timetabling. In: AI 2002: advances in artificial intelligence. Lecture notes in artificial intelligence, pp 455–464
34. Xu J, Chiu SY, Glover F (1999) Optimizing a ring-based private line telecommunication network using tabu search. Manag Sci 45(3):330–345

Zizhen Zhang received the B.S. degree in the Department of Computer Science from Sun Yat-sen University, China, in 2007. He is currently working towards the Ph.D. degree at the Department of Management Sciences, City University of Hong Kong, Kowloon. His current research interests include computational intelligence and its applications in operations research.

Hu Qin received the Ph.D. degree in Management Sciences from City University of Hong Kong, Kowloon, in 2011. He is currently an Associate Professor with the School of Management, Huazhong University of Science and Technology. His current research interests are in the fields of algorithms and artificial intelligence, including various topics in operations research, such as vehicle routing problem, freight allocation problem, container loading problems and transportation problems.

Andrew Lim received the Ph.D. degree in computer science in 1992 from the University of Minnesota, Minneapolis. He is currently a Professor with the Department of Management Sciences, College of Business, City University of Hong Kong. The aim of his research is to help companies compete effectively. This includes deriving customized business value models; identifying key performance indicators, and developing optimized processes that improve these indicators. His works have been published in key journals such as Operations Research and Management Science, and disseminated via international conferences and professional seminars. More importantly, these works have contributed to substantial impacts to many renowned multinational companies leading to international awards and company-wide innovative prizes. However, among the number of activities that he has undertaken, he derives greatest satisfaction in education and mentoring which resulted in strong placements of his students at all levels.