Noname manuscript No.
(will be inserted by the editor)

A Multi-Population Hybrid Biased Random Key Genetic Algorithm for Hop-Constrained Trees in Nonlinear Cost Flow Networks

Dalila B.M.M. Fontes · José Fernando Gonçalves

Received: date / Accepted: date

Abstract Genetic algorithms and other evolutionary algorithms have been successfully applied to solve constrained minimum spanning tree problems in a variety of communication network design problems. In this paper we enlarge the application of these types of algorithms by presenting a multi-population hybrid genetic algorithm to another communication design problem. This new problem is modeled through a hop-constrained minimum spanning tree also exhibiting the characteristic of flows. All nodes, except for the root node, have a nonnegative flow requirement. In addition to the fixed charge costs, nonlinear flow dependent costs are also considered. This problem is an extension of the well know NP-*hard* hop-constrained Minimum Spanning Tree problem (HMST) and we have termed it hop-constrained minimum cost flow spanning tree problem (HMFST). The efficiency and effectiveness of the proposed method can be seen from the computational results reported.

Keywords Multi-population · genetic algorithms · local search · network flows · hop-constrained trees · general nonlinear costs.

1 Introduction

Communication Network Design has increased significantly in the last decade due to the dramatic growth in the use of the Internet for business and personal use. As society transforms itself to an information society the network becomes the primary source for information creation, storage, distribution and retrieval.

A cost-effective structure for a large communication network is a multilevel hierarchical structure consisting of a backbone network (high level) and local

Corresponding author: email `fontes@fep.up.pt`, phone +351934211979, fax +351225505050

Faculdade de Economia da Universidade do Porto
and LIAAD-INESC Porto L.A.
Address: Rua Dr. Roberto Frias, 4200-464 Porto, Portugal

access networks (low level) [3,15]. The Minimum Spanning Tree (MST) problem is one of the best-known network optimization problems used for designing backbone networks, which attempts to find a minimum cost tree network that connects all the nodes in the communication network. There might be other constraints imposed on the design such as the number of nodes in a subtree, the number of nodes in any path from the root node (hop-constrained), degree constraints on nodes (degree constrained), flow and capacity constraints on any arc or node, and types of services available on the arcs or nodes [29,37, 39]. Many polynomial-time algorithms have been proposed for the MST problem by, for example, Dijkstra, Kurskal, and Prim [4]. The basic MST problem can be solved in polynomial-time, however, the addition of one or more constraints often transforms it into problems, such as the Hop-constrained MST, which have been shown to be NP-*hard*, see e.g., [27]. For these problems, a heuristic must be used, at least on large problem instances, since there is no known polynomial-time algorithm to identify an optimum. Genetic algorithms (GAs) are examples of search heuristics that have been successfully applied to such problems.

We consider a new problem which is an extension of the Hop-constrained Minimum Spanning Tree problem (HMST), since in addition to the hop-constraints we also consider flows. All nodes, except for the root node, have a nonnegative flow requirement. Therefore, in addition to finding the arcs that are part of the HMST, we must also find the flow that is to be routed along each of these arcs. Thus, we have named this problem Hop-constrained Minimum cost Flow Spanning Tree, HMFST. The cost, which is to be minimized, is nonlinear and consists of two components: arc setup costs, as usual, and nonlinear flow dependent routing costs. The problem can be equivalently casted as finding a hop-constrained tree solution on minimum cost flow networks with nonlinear costs. Minimum cost network flow problems with nonlinear cost function, either concave or general have been studied in [40,43,9,12,11].

The HMST problem has numerous practical applications and is frequently encountered in network design problems, for example in computer networks we can find the multicast-routing problem (see, e.g., [6,7]), where a number of clients and a server are connected by a common communication network. The server wishes to transmit identical information to all clients, and does so by transmitting the data to the nodes it directly connects to, and these latter nodes forward incoming data to their respective children in the tree. Assuming that all arcs in the network have roughly the same transmission delay (which is a reasonable assumption in local area networks), it is not hard to see that this problem can be cast in the HMST framework. If in this problem, the server wishes to transmit different information to different clients or provide different levels of service to different clients, it no longer can be modeled as a HMST, since different flows to each cliente are now required. However, by including flows, as proposed, the multicast-routing problem with different levels of service or with different information to be transmitted can now be addressed, i.e. it becomes a Hop-constrained Minimum Cost Flow Spanning Tree (HMFST). Tree-routing schemes allow for fast data delivery

while keeping the total network load low. Kompella et al. [34] consider the problem of computing multicast-trees that minimize the overall network cost as well as the maximum transmission delay on any path in the tree connecting the server to a client.

The hop constraints are usually used to guarantee a certain quality of service with respect to availability and reliability constraints [36,45], as well as lower delays [30], since they limit the number of arcs in each path from the central service provider. A decrease in the hop parameter leads to trees with better performance both in terms of reliability and availability, as well as smaller delays. This does not come as a surprise since availability is the probability that all transmission lines, i.e arcs, in a path from the root node are operational, while reliability is the probability that during a session no arc, in the path being used, will fail. Network delays need to be kept under control, since otherwise they may lead to further problems such as the need for packet reordering. This latter problem has been addressed recently by, e.g. [38].

Woolston and Albin [45] have shown that, by including hop constraints it is possible to generate network designs with a much better quality of service and with only a marginal increase in the total cost. Gouveia [27] presented several node oriented formulations for the HMST problem based on the Miller-Tucker-Zemlin subtour elimination constraints. The author presented some lower bounding schemes based on Lagrangian relaxation and a fast heuristic for obtaining feasible solutions, which he has subsequently improved as reported in [28]. More recently Dahl et al. [5] have presented and surveyed several ways of computing lower bounds, including: Lagrangian relaxation, column generation, and model reformulation. Kawatra [33] has addressed a slightly different problem, since he also considers the downtime cost associated with each node that gets disconnected from the central service provider due to arc failure. A Lagrangian relaxation method is developed to find a lower bound to the optimal solution, which is optimized by using subgradient optimization to find Lagrangian multipliers. A branch exchange heuristic is used to obtain a feasible solution from an infeasible Lagrangian lower bound solution. The best feasible solution is retained when the heuristic method stops. The best lower bound given by the Lagrangian relaxation method is used to estimate the quality of the heuristic solution.

Since the hop constrained minimal spanning tree problem is NP-*hard* (see e.g. Gouveia [27]), much of the research on this problem has been devoted to optimization-based heuristics that provide good solutions.

Recently, Genetic Algorithm (GA) and other Evolutionary Algorithms (EAs) have been successfully applied to solve constrained spanning tree problems of the real life instances; and also have been used extensively in a wide variety of communication network design problems [15,16]. For example, some authors have proposed GAs for the capacitated MST problem [1,35], while others propose GAs for the degree-constrained MST, see for example [31,46] for a hybrid GA (with local search). Other researchers have investigated different encoding schemes, see for example [42,44]. GAs have also been proposed

for other types of trees-shaped problems, for instance in [9] the authors address problems including flows and with general nonlinear cost functions.

In this paper, we propose a hybrid random key genetic algorithm, based on a previous version [13], to solve the hop-constrained minimum cost flow spanning tree problem, which is an extension of the HMST. For the HMST we wish to find a minimum cost tree spanning all nodes in a given network such that any path from the root node to any other node has no more than a pre-specified number of arcs. For the problem considered here however, we also have flow decisions to be made since all nodes, except for the root node, have a nonnegative flow requirement. Therefore, in addition to finding the arcs that are part of the HMST, we must also find the flow that is to be routed along each of these arcs. Furthermore, the costs to be minimized include a general nonlinearly flow dependent cost component and a fixed cost component. Nonlinear cost functions arise naturally in these types of problems as a consequence of taking into account economic considerations. Set up costs or fixed-charge costs arise, for example, due to the consideration of a new customer or a new route. Economies of scale often exist, and thus an output increase leads to a decrease in the marginal costs. On the other hand, further output increase may lead to an increase in marginal costs, e.g. by implying the need of extra resources. Therefore, discontinuities are observed. These may also arise due to price-discounting. As far as we are aware of, no previous work, except that of Fontes [8] (that uses an exact method) has considered HMST with flow requirements and/or having nonlinear costs.

2 Problem description

As aforementioned, our objective is to find a minimum cost tree spanning all nodes in a given network having general nonlinear arc costs and subject to hop constraints, i.e., a limit on the maximum number of arcs that any path from the root node to any other node may contain. Since all nodes, except for the root node, have a nonnegative flow requirement, in addition to finding the arcs that form the HMST, we must also find the flows that are to be routed along these arcs such that all demand nodes are satisfied.

Let $G = (W, A)$ denote a directed network defined by a set W of $n+1$ nodes (the source node and a set V containing n demand nodes, i.e. $W = V \cup \{t\}$) and a set A of m directed arcs. Nodes 1 to n have associated a nonnegative integer demand r_i, which must be satisfied. The total cost to be minimized, is given by the summation of all costs incurred by both using an arc (a setup cost) and routing flow through it (a flow cost), since each arc $(x, z) \in A$ has associated a general nonlinear and nonnegative cost function g_{xz}. The cost of sending r units of flow through an arc, say (x, z) is given by any function $g_{xz}(r)$ satisfying $g_{xz}(0) = 0$. (The flow that can be routed through each arc (x, z) may have upper u_{xz} and lower l_{xz} limits.) The hop parameter H forces all paths, of the minimum cost tree connecting the source node to any other node, to have no more than H arcs.

The cost function g_{xz} consists of a nonlinear routing component and a fixed cost component. Furthermore, a discontinuity point, other than at the origin is also considered, let it be represented by $\hat{R} < R$, where R is the total requirement and is, by definition, given by $R = \sum_{i \in V} r_i$..

The problem can be cast in an intuitive way in the form of a nonlinear programming model. Let us first define the decision variables and parameters.

Decision variables:

x_{ijh} - flow on arc $(i,j) \in A$, which is in position h,

$$y_{ijh} = \begin{cases} 1, \text{ if } x_{ijh} > 0, \\ 0, \text{ if } x_{ijh} = 0. \end{cases}$$

$$z_{ijh} = \begin{cases} 1, \text{ if } x_{ijh} \geq \hat{R}, \\ 0, \text{ if } x_{ijh} < \hat{R}. \end{cases}$$

$$(\mathcal{P}) \text{ Minimize} \quad \sum_{i=1}^{n+1} \sum_{j=1}^{n} \sum_{h=1}^{H} g_{ij}(x_{ijk}, y_{ijk}, z_{ijk}) \tag{1}$$

subject to

$$\sum_{i=1}^{n+1} \sum_{h=1}^{H} y_{ijh} = 1, \quad \text{for all } j \in V, \tag{2}$$

$$\sum_{i=1}^{n+1} X_{ijh} - \sum_{i=1}^{n} X_{jih+1} = r_j \sum_{i=1}^{n+1} Y_{ijh}, \quad \text{for all } j \in V, h = 1, \ldots, H-1, \tag{3}$$

$$x_{ijh} - \hat{R} \leq R\, z_{ijh}, \quad \text{for all } i \in W, j \in V, h = 1, \ldots, H, \tag{4}$$

$$x_{ijh} \geq 0, \quad \text{for all } i \in W, j \in V, h = 1, \ldots, H, \tag{5}$$

$$y_{ijh} \in \{0,1\}, \quad \text{for all } i \in W, j \in V, h = 1, \ldots, H, \tag{6}$$

$$z_{ijh} \in \{0,1\}, \quad \text{for all } i \in W, j \in V, h = 1, \ldots, H. \tag{7}$$

Equation (1) states the minimum cost nature of the problem. Note that it depends on x_{ijh} due to the routing cost, on y_{ijh} due to the fixed cost component, and on z_{ijh} due to the additional discontinuity point, other than at the origin. No cycles may exist, which is guaranteed by the constraints (2) since exactly one arc enters each node. Constraints (3) represent the flow conservation constraints, while constraints (4) guarantee that z_{ijh} is 1 if the flow is beyond the discontinuity value if there is an increase in cost (or a decrease, in this case the constraint becomes $x_{ijh} - \hat{R} \leq R(1 - z_{ijh})$). Finally, constraints (5) to (7) defined the domain of the variables.

3 Solution Approach

The new approach is based on a Multi-Population Hybrid Biased Random Key Genetic Algorithm, MPHBRKGA, that hybridizes a genetic algorithm with a local search method. A constructive heuristic algorithm, Tree-Constructor, is used to generate solution trees. Then a local search heuristic is applied to try to improve the solution obtained by the tree-constructor by searching amongst neighbor feasible solutions. A penalty cost term is included in the fitness function to drive the solutions towards solutions satisfying the Hop constraints.

The role of the MPHBRKGA is to evolve the encoded solutions, or chromosomes, which represent the input to the Tree Constructor and the Local Search Procedure. For each chromosome, the following phases are applied:

1. **Solution Construction**. In the first phase the Tree-Constructor transforms part of the chromosome supplied by the genetic algorithm into a solution tree.
2. **Solution Improvement**. In the second phase, part of the chromosome supplied by the genetic algorithm is used by a local search procedure to try to improve the solution tree obtained in the previous phase.
3. **Fitness Evaluation**. This phase computes the cost of the final solution tree. A penalty cost term is included in the cost to drive the evolutionary process to solutions satisfying the Hop constraints.

Figure 1 illustrates the sequence of steps applied to each chromosome generated by the genetic algorithm.

In the remainder of the paper we describe in detail all the components of the methodology.

4 Genetic Algorithm

Holland first proposed GAs in the early 1970s as computer programs that mimic the evolutionary processes in nature [32]. Since then GAs have been demonstrating their power by successfully being applied to many practical optimization problems in the last decade.

We use a random key representation and followed the biased random key genetic algorithm (BRKGA) framework that has been proposed by Gonçalves and Resende [25]. In [25] the authors present a tutorial on the implementation and use of biased random-key genetic algorithms for solving combinatorial optimization problems. Furthermore, they also provide a survey of recent successful applications that appeared in the literature.

To specify a biased random-key genetic algorithm, we simply need to specify how solutions are represented and decoded into solutions and how there corresponding fitness values are computed. In the nest sections we specify our algorithm next by first showing how the solutions are represented and then decoded into solutions and how their fitness evaluation is computed.

4.1 Chromosome Representation

The choice of genetic representation is usually the first and probably most important single decision, but many other decisions affect the effectiveness of the algorithm. The BRKGA described in this paper proposes a random-key alphabet, which is comprised of real-valued random numbers between 0 and 1. Random keys have been used successfully for addressing problems where the relative order of tasks is important, e.g., [17,18,19,20,21,22,23,24,25,26].

A chromosome is made of $3n$ genes, and is represented as a vector of $3n$ random keys, where n is the number of nodes (see Fig. 2). The first $2n$ genes of the chromosome are decoded into a solution tree by the Tree-Constructor procedure described in Section 5. The last n genes are used, by the local search procedure described in Section 6, to improve upon the existing solutions.

4.2 Evolutionary Strategy

Given the current population and the fitness value of each chromosome, i.e. total cost incurred in constructing the tree and routing the flow plus a penalty (for Hop constraints not satisfied), the population is divided into two sets: the elite solutions subset and the remaining solutions. The elite subset is a small subset containing only very good solutions.

The population in the next generation is formed by all solutions in the elite set (TOP), a small number of new randomly generated solutions (mutants) (BOT), and solutions obtained by mating solutions of the current population (offspring), see Figure 3.

By copying the best solutions, we guarantee that the best solution is monotonically improving. However, it may lead to excessive convergence to a local optimum. To overcome this problem we use two strategies. On the one hand, we use mutation as described below, and on the other hand we use several populations, which are evolved separately.

4.2.1 Crossover

In the crossover operator two individuals are randomly chosen to act as parents. One of them is chosen amongst the elite solution set (TOP), while the other is chosen from the entire population. Genes are chosen by using a biased uniform crossover, that is, for each gene a biased coin is tossed to decide on which parent the gene is taken from. This way, the child inherits the genes from the elite parent with higher probability, see Figure 4.

4.2.2 Mutation

As mutation operator we use the so-called immigration operator. Immigration acts like a mutation operator, however, instead of performing gene-by-gene mutation with very small probability, at each generation some new individuals

are introduced into the next generation. This new individuals are randomly generated from the same distribution as the original population and thus, no genetic material of the current population is brought in.

4.3 Multi Population Strategy

As mentioned before several populations are evolved, each being initially randomly generated. The populations are left to evolve independently and after a predetermined number of generations they interact by exchanging information. The information exchanged is the chromosomes of good quality solutions. When evaluating possible interchange strategies we observed that exchanging too many chromosomes, or exchanging them too frequently, often leads to the disruption of the evolutionary process. Therefore, we choose a strategy that after a pre-determined number of generations (determined empirically) inserts only the best two chromosomes in all populations.

5 Tree-Constructor

The Tree-Constructer uses the first $2n$ genes of each chromosome supplied by the BRKGA to construct a solution tree. The first n genes are used to determine the order in which nodes are considered by the tree constructor to find a source to be supplied from. The second n genes are used to select the corresponding source node.

The decoding (mapping) of the first n genes of each chromosome into the node sequence, NS, in which the nodes are considered by the tree constructor is accomplished by sorting, in ascending order of gene values, the corresponding nodes, see Fig. 5. The source of each node i is decoded by choosing, amongst the possible source nodes for node i, the one that has the smallest gene value and does not cause a cycle in the graph.

The Tree-Constructer repeatedly performs (for $i = 1$ to $n-1$) the following three steps in turn.

1. Select for processing node $iSel = NS[i]$;
2. Search for the set of nodes S such that an arc $(isel, j)$ exists, i.e. $S = \{i \in W : (i, j) \in A\}$;
3. Chose the node $k \in S$ with the smallest value of $gene_K$ not creating a cycle (with the arcs already chosen), if one exists.

At the end of the algorithm either a tree or an infeasibility has been obtained.

To illustrate the Tree-Constructer procedure we will consider the network data depicted in Table 1 and the chromosome given in Fig. 6.

The decoding of the first five genes into the node sequence (the source node is excluded since it has no node acting as a source to it) results in the following $NS = (3, 5, 1, 4, 2)$. The pictures in Fig. 7 will be used to illustrate the various steps of the algorithm.

Initially we have a network with no connections (no acting sources assigned) (see step 0 of Fig. 7).

In step 1, and according to the vector NS, we will process node 3. For this node the potential nodes for acting as its source are (1, 2, 4) since node 2 has the smallest gene value we select node 2 as its source node (see step 1 of Fig. 7).

In step 2, and according to the vector NS, we will process node 5. For this node the potential nodes for acting as its sources are (2, 3, 4) since node 2 has the smallest gene value we select node 2 as its source node (see step 2 of Fig. 7).

In step 3, and according to the vector NS, we will process node 1. For this node the potential nodes for acting as its source are (2, 4, 6) since node 2 has the smallest gene value we select node 2 as its source node (see step 3 of Fig. 7).

In step 4, and according to the vector NS, we will process node 4. For this node the potential nodes for acting as its source are (3, 5, 6) since node 3 has the smallest gene value we select node 3 as its source node (see step 4 of Fig. 7).

In step 5.1, and according to the vector NS, we will process node 2. For this node the potential nodes for acting as its source are (1, 3, 5, 6) since node 3 has the smallest gene value we select node 3 as its source node (see step 5.1 of Fig. 7). However, since the selection causes a cycle we have to repeat step 5 and choose another node, excluding node 3 from the possible source of node 2.

In step 5.2, and according to the vector NS, we will process node 2. For this node the potential nodes for acting as its source are (1, 5, 6) since node 6 has the smallest gene value we select node 6 as its source node (see step 5.2 of Fig. 7).

6 The Local Search

The local search tries to improve on a given solution by comparing it with adjacent extreme solutions. As no transshipment node exists, adjacent extreme solutions are obtained by replacing an arc currently in the solution by an arc not in the solution such that the new solution is still an extreme flow, i.e., a tree. Such strategy is based on the generalized definition of neighborhood by Gallo and Sodini [14]. An extreme flow X' with induced tree $T_{X'} = (W_{T_{X'}}, A_{T_{X'}})$, is adjacent to X if and only if the arcs in $A_{T_{X'}} \setminus A_{T_X}$ constitute a path that connects two vertices in W_{T_X} and does not contain any other vertex in W_{T_X}, where $T_X = (W_{T_X}, A_{T_X})$ is the tree induced by X and W_{T_X} and A_{T_X} are the set of vertices and the set of arcs in the tree, respectively.

Consider the example in Figure 8. Let X' be obtained from X by removing arc (l, m) and including a arc (k, m) from some vertex k in X. By joining the two extreme flows a single undirected cycle is formed. Let us decompose it into

four parts: arcs (k, m) and (l, m) and paths P_{qk}, and P_{qm}, where q is the last common vertex to paths P_{tk} and P_{tl}.

The total resulting change in the objective function value is

$$\Delta C = \Delta C_m(k) + g_{km}(x_{lm}) - g_{lm}(x_{lm}),$$

where $\Delta C_m(k)$ is the cost variation due to redirecting flow x_{lm} from P_{ql} to P_{qk}. If ΔC is negative then X' is better and the current solution is updated; otherwise the solution X is kept.

The order by which nodes are considered for improvement by the local search, IS, is accomplished by sorting in ascending order of gene values the corresponding nodes, see Fig. 5. According to the chromosome values given in Fig. 6 $IS = (2, 4, 3, 5, 1)$. (Note that the source node, 6, is excluded since it is not to be supplied by any other node).

Figure 9 depicts a possible outcome of the local search procedure when applied to node 1. In this case node 1 is being supplied by node 2, which is acting as a source, and could have as other potential "sourcesťť nodes 4 and 6. Assume that exchanging the acting as source node, from node 2 to node 4 we get $\Delta C = 10$, while if the exchange is to node 6 we get $\Delta C = -15$. Then the local search procedure would improve upon the current solution by exchanging the node acting as a source of node 1 to node 6.

7 Fitness

As mentioned before, the Hop constraints are not considered within the Tree-Constructor procedure and are handled implicitly by a penalty cost term in the fitness. Therefore, the fitness of each chromosome is equal to the sum of two cost terms, a cost term associated with cost of the flow and another cost term associated with a penalty for not satisfying the Hop constraints. The penalty cost depends on the Hop constraints violation degree. That is, we penalize nodes having more than H arcs in their path from the source node, as follows.

$$Max\{0, patharcs_i - H\} \times M, \text{ for all nodes } i, \tag{8}$$

where $patharcs_i$ represents the number of arcs in the path from the source node to node i and M represents the penalty factor. In summary, the fitness function associated with each chromosome is given by Flow Cost + Penalty Cost.

8 Computational results

The algorithm proposed in this paper was implemented in C++ and the computational experiments were performed on a personal computer with a Intel Core2 processor at 2.4 GHZ and a Linux Fedora 12 operating system. The pseudo-random generator used is based on the 1998 paper of Park and Miller [41], who wrote an excellent paper on random number generators. The random

number generator code in C++ is present in the appendix A. The proposed algorithm was computationally evaluated by solving a set of randomly generated test problems. The problems considered are amongst the most difficult nonlinear network flow problems since all arcs have nonlinear cost functions, some are neither convex nor concave. The cost functions consist of two components: a set-up cost and a routing cost. The problem data can be downloaded from the OR-Library[1] , see [2], and a thorough description of the generation procedure is provided in [10].

Three different cost function types are considered: types G1 and G2 are variations of the fixed-charge cost function where discontinuities other than at the origin are introduced; and type G3, where arc costs are initially concave and then convex, having a discontinuity at the break point. The discontinuity point was set to $\hat{R} = 0.5\ R$.

$$g_{ij}(r) = \begin{cases} 0, & \text{if } r = 0, \\ -a_{ij}r^2 + b_{ij}r + c_{ij} & \text{if } r \leq \hat{R}, \\ a_{ij}r^2 + b_{ij}r + c_{ij} + k & \text{otherwise,} \end{cases} \tag{9}$$

where $a_{ij} = 0$ for G1 and G2, $k = b_{ij}$ for G1, $k = -b_{ij}$ for G2, and $k = 0$ for G3.

A graphical representation of the fixed-charge cost function variations, G1 and G2, is given in Figure 10, while Figure 11 represents a cost function that is initially concave and then convex, as is the case for cost functions of type G3. Types G1 and G2 correspond, respectively, to the so called staircase and sawtooth cost functions with two segments.

In Table 2, we report on the parameters used in the MPHBRKGA. The seeds used to initialize the random number generator of each of the 3 populations of each of the five runs are given in Table 3.

In tables 4 to 6, we summarize the results obtained for uncapacitated problems involving cost functions of types G1, G2, and G3, respectively, with the discontinuity point occurring at 50% of the root node outflow, which is obviously the total requirement R. Four different arc limit values have been considered $H = 3, 5, 7, 10$. Nine problem sizes have been considered $n = 10, 12, 15, 17, 19, 25, 30, 40, 50$. For each size considered, each cost function type (G1, G2, and G3) and hop parameter value ($H = 3, 5, 7, 10$) value we have solved 30 problem instances for the smaller size problems (up to 30 nodes) and 15 problem instances for the remaining problem sizes. Each of the 720 ($3 \times (7 \times 30 + 2 \times 15)$) problem instances has been solved for each of the four considered hop parameter values five times. Thus, overall we have solved 2880 problem instances, five times each.

We report on the average computational time, in seconds, required by the MPHBRKGA. We also report on the average deviation from the optimal

[1] The problems ar found under "Network flow: Single commodity, concave costs, single source, uncapacitated" and are named "CCNFP10g1a.txt, CCNFP10g1b.txt, CCNFP10g1c.txt, CCNFP10g2a.txt, ... , CCNFP10g10c.txt, CCNFP12g1a.txt, ... , CCNFP50g5c.txt".

value. The percentage deviations from the optimal values (minimum, maximum, and average) have been computed using the optimal solutions obtained by CPLEX for problems of type G1 and G2 and by the dynamic programming methodology proposed in [8] for problems of type G3 with up to 19 nodes as $\frac{MPHBRKGA-Opt}{Opt} \times 100$.

As can be seen from the results reported in tables 4 to 6, the multi population genetic algorithm is capable of finding an optimal solution for all problem instances, since the minimum deviation from optimal is always zero. However, this is not the case for all solutions found. As it can be seen, the average deviation from optimal is small but positive. It can also be observed that problems are harder to solve for tighter hop constraints, i.e. smaller hop parameter values.

In order to better understand the results obtained and the conclusions drawn we provide some graphical representations of the results, given in figures 12 and 13. The computational times reported show that the time requirements of the DP algorithm grow very rapidly. The CPLEX computational times also have a rather bigger rate of increase with problem size. However, this trend is not observed in the proposed algorithm, see Figure 13. It should be noticed that each figure shown in the tables and in the graphs is an average obtained after solving 30 or 15 problem instances, depending on the size under consideration, for each combination of problem size, cost function type, and hop parameter value.

9 Conclusions

In this work the problem of finding Hop-Constrained Trees in Nonlinear Cost Flow Networks, a very recent problem first proposed by [8], is addressed. This problem is a generalization of the hop-constrained Minimum Spanning Tree problem, since it also includes the determination of the flows to be routed through the tree, and therefore it is NP-*hard*. In practice it can be used to model the multicast-routing problem, where a number of clients and a server are connected by a common communication network, where the server wishes to transmit nonidentical information to all clients. Information is transmitted by the server to the nodes it directly connects to, and these latter nodes forward incoming data to their respective children in the tree.

We have presented a Multi Population Hybrid Random Key Genetic Algorithm that on average finds nearly optimal solutions. In addition, as the results have shown, the MPHBRKGA actually has been able to find an optimal solution for all problem instances solved. Recall that CPLEX and a DP method were used to obtain an optimal solution for these problems. The proposed algorithm combines a local search algorithm with a Multi Population biased random key genetic algorithm, where several populations are evolved independently.

We have solved 240 problem instances with four different hop-parameter values and three different cost function types (2880 overall). The cost func-

tions considered are amongst the most difficult ones, since in addition to a fixed cost component they also include a nonlinear routing cost component. Furthermore, the routing cost functions are neither convex nor concave and have discontinuity points other than at origin. The results obtained have been compared with the existing literature and the comparisons have shown the proposed algorithm to improve upon the efficiency of existing methods [8], since the computational time requirements are very modest for all problem sizes (always below one minute). When the Hop-constraints are not very tight we were able to find an optimal solution for almost all runs performed. Nevertheless, when they are very tight we are still able to find very good average solutions, always below 0.425%. In addition, we were always able to find an optimal solution for all problem instances regardless of the cost function type and hop parameter value.

Thus, the Multi Population Hybrid Genetic Algorithm proposed here is capable of efficiently finding heuristic solutions, close to optimal (when not optimal), for the Hop-Constrained Minimum Cost Flow Spanning Tree Problem, which is NP-*hard*.

Acknowledgements The financial support by FCT, POCI, COMPTE, and FEDER, through projects PTDC/EGE-GES/099741/2008 and PTDC/EGE-GES/117692/2010 is gratefully acknowledged.

References

1. Ahuja, R., Orlin, J.: Multi-exchange neighborhood structures for the capacitated minimum spanning tree problem. Mathematical Programming **91**, 71–97 (2001)
2. Beasley, J.E.: Or-Library: Distributing test problems by electronic mail. Journal of The Operational Research Society **41**, 1069–1072 (1990)
3. Boorstyn, R., Frank, H.: Large-scale network topological optimization. IEEE Transactions on Communications **COM-25**, 29–47 (1977)
4. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to algorithms, 2nd edn. MIT press Cambridge, MA (2001)
5. Dahl, G., Gouveia, L., Requejo, C.: On formulations and methods for the hop-constrained minimum spanning tree problem. In: P.M. Pardalos, M. Resende (eds.) Handbooks of Telecommunications, pp. 493–515. Springer (2006)
6. Deering, S.E., Cheriton, D.R.: Multicast routing in datagram internetworks and extended lans. ACM Transactions on Computing Systems **8**, 85–110 (1990)
7. Deering, S.E., D., D.E., Farinacci: An architecture for wide-area multicast routing. Proceedings of SIGCOMM (1994)
8. Fontes, D.B.M.M.: Optimal hop-constrained trees for nonlinear cost flow networks. INFOR **48**, 13–22 (2010)
9. Fontes, D.B.M.M., Gonçalves, J.F.: Heuristic solutions for general concave minimum cost network flow problems. Networks **50**, 67–76 (2007)
10. Fontes, D.B.M.M., Hadjiconstantinou, E., Christofides, N.: Upper bounds for single source uncapacitated minimum concave-cost network flow problems. Networks **41**, 221–228 (2003)
11. Fontes, D.B.M.M., Hadjiconstantinou, E., Christofides, N.: A branch-and-bound algorithm for concave network flow problems. Journal of Global Optimization **34**, 127–155 (2006)
12. Fontes, D.B.M.M., Hadjiconstantinou, E., Christofides, N.: A dynamic programming approach for solving single-source uncapacitated concave minimum cost network flow problems. European Journal of Operational Research **174**, 1205–1219 (2006)

13. Fontes, D.B.M.M., Gonçalves, J.F.: Upper Bounds for Single Source Uncapacitated Concave Minimum Cost Network Flow Problems. Proceedings of INOC - International Network Optimization Conference (2009)
14. G. Gallo and C. Sodini, Adjacent extreme flows and application to min concave cost flow problems, Networks **9**, 95–121 (1979)
15. Gen, M., Cheng, R., Oren, S.: Network design techniques using adapted genetic algorithms. Advances in Engineering Software **32**, 731–744 (2001)
16. Gen, M., Kumar, A., Kim, R.: Recent network design techniques using evolutionary algorithms. International Journal of Production Economics **98**, 251–261 (2005)
17. Gonçalves, J.F., Mendes, J.J.M., Resende, M.: A genetic algorithm for the resource constrained multi-project scheduling problem. European Journal of Operational Research **189** pp. 1171–1190 (2009).
18. Gonçalves, J., Resende, M.: A parallel multi-population genetic algorithm for a constrained two-dimensional orthogonal packing problem. Journal of Combinatorial Optimization **22** pp. 1–22 (2010).
19. Gonçalves, J., Resende, M., Mendes, J.: A biased random-key genetic algorithm with forward-backward improvement for the resource constrained project scheduling problem. Journal of Heuristics **17** pp. 1–20 (2010).
20. Gonçalves, J., Sousa, P.: A genetic algorithm for lot sizing and scheduling under capacity constraints and allowing backorders. International Journal of Production Research **49**(9), 2683–2703 (2011).
21. Gonçalves, J.: A hybrid genetic algorithm-heuristic for a two-dimensional orthogonal packing problem. European Journal of Operational Research **183**(3), 1212 – 1229 (2007).
22. Gonçalves, J., Almeida, J.: A hybrid genetic algorithm for assembly line balancing. Journal of Heuristics **8**, 629–642 (2002).
23. Gonçalves, J., Mendes, J., Resende, M.: A hybrid genetic algorithm for the job shop scheduling problem. European Journal of Operational Research **167**(1), 77 – 95 (2005).
24. Gonçalves, J., Resende, M.: An evolutionary algorithm for manufacturing cell formation. Computers and Industrial Engineering **47**, 247–273 (2004).
25. Gonçalves, J., Resende, M.: Biased random-key genetic algorithms for combinatorial optimization. Journal of Heuristics **17**, 487–525 (2010).
26. Gonçalves, J., Resende, M.: A parallel multi-population biased randomkey genetic algorithm for a container loading problem. Computers & Operations Research **39**, 179–190 (2012).
27. Gouveia, L.: Using the miller-tucker-zemlin constraints to formulate a minimal spanning tree problem with hop constraints. Computers and Operations Research **22**, 959–970 (1995)
28. Gouveia, L.: Multicommodity flow models for spanning trees with hop constraints. European Journal of Operational Research **91**, 178–190 (1996)
29. Gouveia, L., Martins, P.: The capacitated minimum spanning tree problem: revisiting hop-indexed formulations. Computers & Operations Research **32**, 2435–2452 (2005)
30. Gouveia, L., Requejo, C.: A new lagrangean relaxation approach for the hop-constrained minimum spanning tree problem. European Journal of Operational Research **132**, 539–552 (2001)
31. Han, L., Wang, Y., Guo, F.: A new genetic algorithm for the degree-constrained minimum spanning tree problem. IEEE International Workshop on VLSI Design and Video Technology pp. 125–128 (MAY 2005)
32. Holland, J.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI (1975)
33. Kawatra, R.: A hop constrained min-sum arborescence with outage costs. Computers & Operations Research **34**, 2648–2656 (2007)
34. Kompella, V., Pasquale, J., Polyzos, G.: Multicast routing for multimedia communication. IEEE/ACM Trans. Networks **1**, 286–292 (1993)
35. Lacerda, E., Medeiros, M.: A genetic algorithm for the capacitated minimum spanning tree problem. IEEE Congress on Evolutionary Computation **1-6**, 725–729 (2006)
36. LeBlanc, L., Reddoch, R.: Reliable link topology/capacity design and routing in backbone telecommunication networks. First ORSA telecommunications SIG conference (1990)

37. Lee, Y., Atiquzzaman, M.: Least cost heuristic for the delay constrained capacitated minimum spanning tree problem. Computer Communications **28**, 1371–1379 (2005)
38. Lelarge, M.: Packet reordering in networks with heavy-tailed delays. Mathematical Methods of Operations Research **67**, 341–371 (2008)
39. Montemanni, R., Gambardella, L.: A benders decomposition approach for the robust spanning tree problem with interval data. European Journal of Operational Research **161**, 771–779 (2005)
40. Nahapetyan, A., Pardalos, P.: Adaptive dynamic cost updating procedure for solving fixed charge network flow problems. Computational Optimization and Applications **39**, 37–50 (2008)
41. Park, S.K., Miller, K.W.: Random number generators: good ones are hard to find. Communications of the ACM **31**, 1192–1201 (1998)
42. Raidl, G., Julstrom, B.: Edge sets: An effective evolutionary coding of spanning trees. IEEE Transactions on Evolutionary Computation **7**, 225–239 (2003)
43. S., R., A., N., P.M., P.: Bilinear modeling solution approach for fixed charge network flow problems. Optimization Letters **3**, 347–355 (2009)
44. Thompson, E., Paulden, T., Smith, D.: The dandelion code: A new coding of spanning trees for genetic algorithms. IEEE Transactions on Evolutionary Computation **11**, 91–100 (2007)
45. Woolston, K., Albin, S.: The design of centralized networks with reliability and availability constraints. Computers & Operations Research **15**, 207–217 (1988)
46. Zeng, Y., Wang, Y.: A new genetic algorithm with local search method for degree-constrained minimum spanning tree problems. ICCIMA - 5th International Conference on Computational Intelligence and Multimedia Applications pp. 218–222 (September 2003)

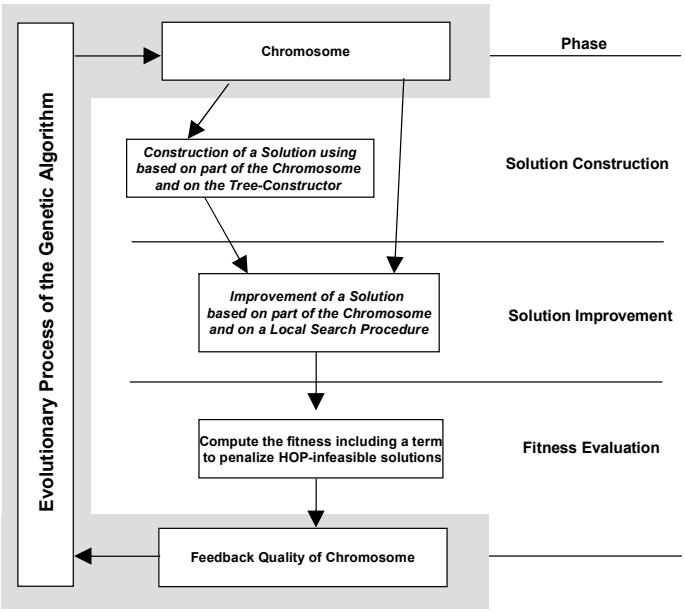


Fig. 1 The solution approach.

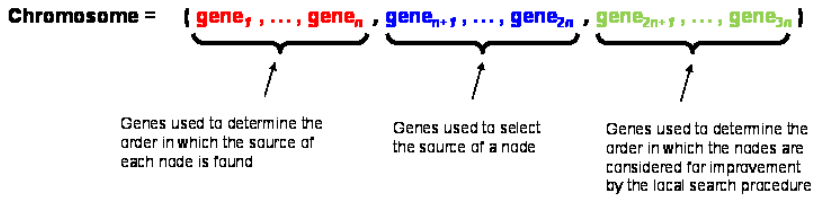


Fig. 2 Chromosome representation.

Node	Sources
1	2, 4, 6
2	1, 3, 5, 6
3	1, 2, 4
4	3, 5, 6
5	2, 3, 4

Table 1 Network data for the tree construction example.

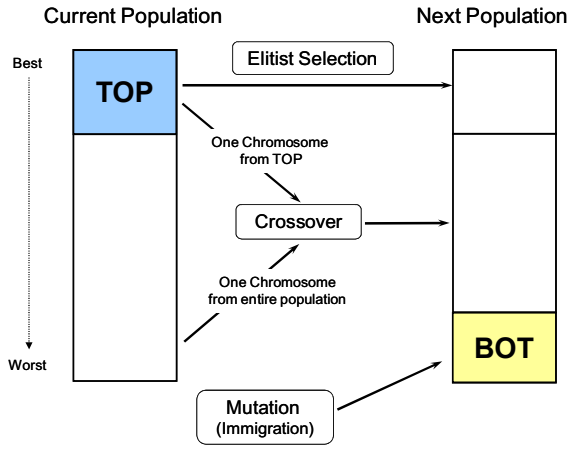


Fig. 3 Evolutionary strategy.

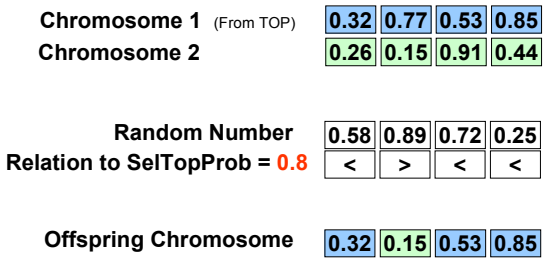


Fig. 4 Parameterized crossover example.

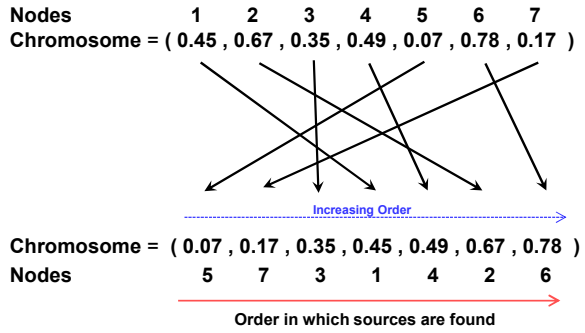


Fig. 5 Decoding procedure for the Tree-Constructor.

Nodes	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6
Chromosome	.55	.75	.35	.65	.45	.99	.9	.3	.4	.6	.5	.1	.77	.22	.57	.45	.66	.14

Fig. 6 Chromosome for the tree construction example.

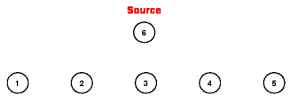
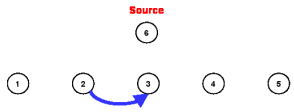
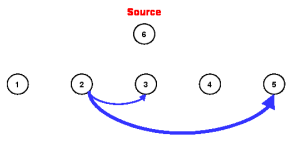
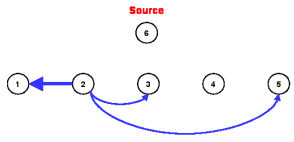
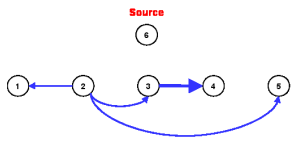
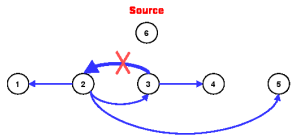
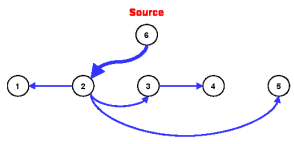
Step	Node	Sources	Source selected	Network
0				
1	3	1, 2, 4	2	
2	5	2, 3, 4	2	
3	1	2, 4, 6	2	
4	4	3, 5, 6	3	
5.1	2	1, 3, 5, 6	3	
5.2	2	1, 5, 6	6	

Fig. 7 Tree construction example.

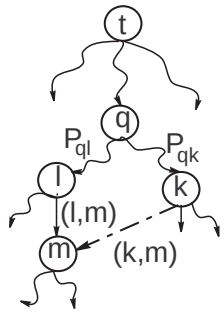
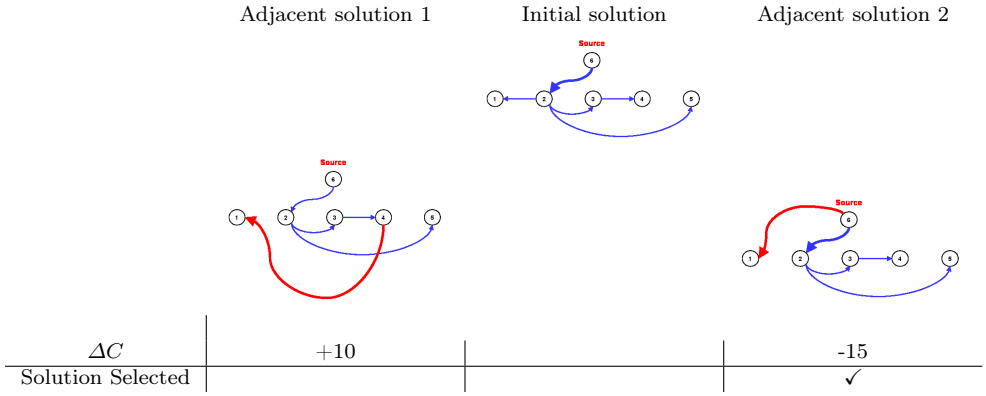


Fig. 8 Cycle decomposition.



ΔC	+10		-15
Solution Selected			✓

Fig. 9 Improvement example.

Parameter	Value
Population Size	15 times the number of vertices
Crossover	The probability of tossing heads was made equal to 0.7
Selection	Copy to the next generation the top 25% chromosomes
Mutation	Substitute the bottom 15% with randomly generated chromosomes
Fitness	Cost (to minimize) and penalty function
Number of Populations	3
Exchange Frequency	Every 15 generations
Number of Seeds	1
Stopping Criterion	1000 generations

Table 2 MPHBRKGA configuration.

Run	Population 1 – Population 2 – Population 3
1	1776393743 – 1531978007 – 1772919766
2	1126905037 – 1234673966 – 30865601
3	1214597080 – 1901058825 – 833971709
4	2084232841 – 2095592470 – 1890832490
5	758651124 – 1039028829 – 1767995246

Table 3 Seeds used for the computational experiments.

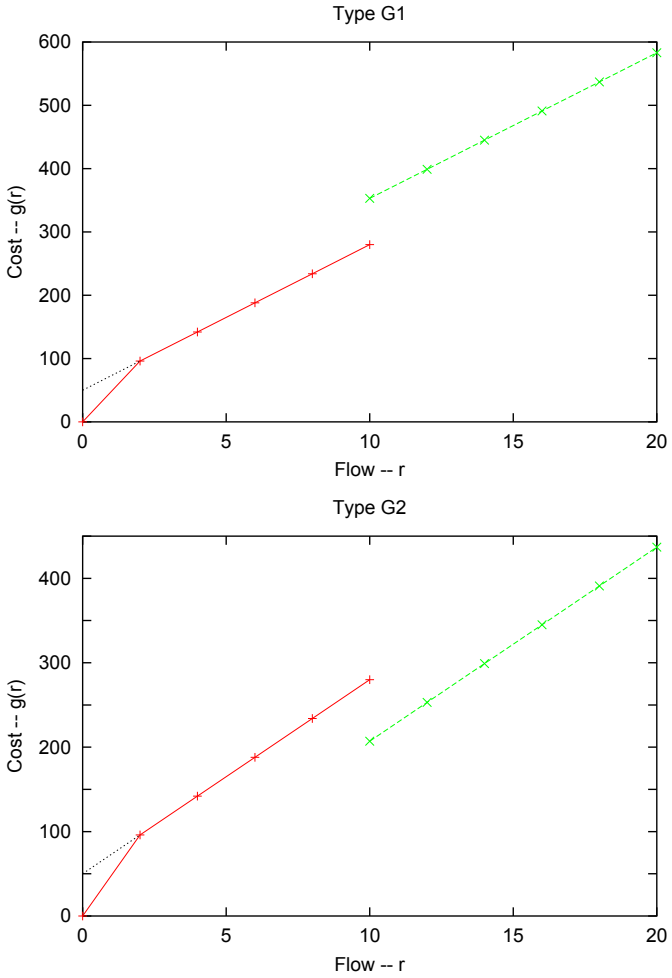


Fig. 10 Staircase and Sawtooth cost functions, respectively.

	Hop parameter values													
	3			5			7			10			CPLEX	
N	Min.	Aver.	Max.	Min.	Aver.	Max.	Min.	Aver.	Max.	Min.	Aver.	Max.	Time	Time
10	0	0.360	7.157	0	0	0	0	0	0	0	0	0	4.35	8.45
12	0	0.104	1.900	0	0	0	0	0	0	0	0	0	6.85	8.67
15	0	0.031	0.641	0	0	0	0	0.005	0.258	0	0	0	11.11	8.88
17	0	0.120	3.743	0	0	0	0	0	0	0	0	0	17.08	9.27
19	0	0.182	17.353	0	0.062	4.919	0	0	0	0	0	0	21.96	10.06
25	0	0.283	6.141	0	0.005	0.162	0	0	0	0	0	0	44.52	15.02
30	0	0	0	0	0.109	4.083	0	0	0	0	0	0	70.21	21.55
40				0	0.139	7.501	0	0.074	1.945	0	0	0	119.13	33.00
50				0	0.076	0.991	0	0.047	1.426	0	0	0	232.84	51.02

Table 4 Solution quality (% deviation from optimal) and time (s) for cost function type G1.

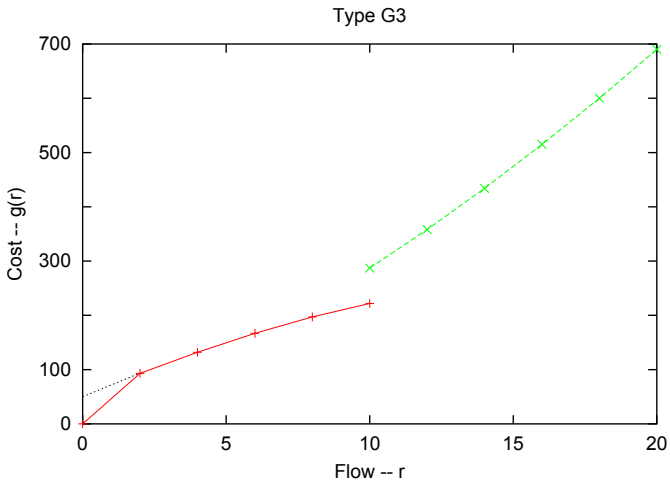


Fig. 11 Concave cost function up to the discontinuity point, which then becomes convex.

	Hop parameter values													
	3			5			7			10|			CPLEX	
N	Min.	Aver.	Max.	Min.	Aver.	Max.	Min.	Aver.	Max.	Min.	Aver.	Max.	Time	Time
10	0	0.389	7.162	0	0	0	0	0	0	0	0	0	4.906	8.555
12	0	0.110	1.914	0	0	0	0	0	0	0	0	0	7.879	8.682
15	0	0.075	2.396	0	0	0	0	0.006	0.228	0	0	0	12.201	8.888
17	0	0.120	3.743	0	0	0	0	0	0	0	0	0	19.877	9.449
19	0	0.041	1.787	0	0.095	4.922	0	0	0	0	0	0	23.305	10.205
25	0	0.251	6.993	0	0.006	0.177	0	0	0	0	0	0	50.687	14.942
30	0	0	0	0	0.071	4.083	0	0	0	0	0	0	112.754	21.264
40				0	0.425	13.909	0	0.048	1.945	0	0	0	121.342	33.110
50				0	0.084	0.992	0	0.004	0.243	0	0	0	273.912	51.089

Table 5 Solution quality (% deviation from optimal) and time (s) for cost function type G2.

	Hop parameter values													
	3			5			7			10|			DP	
N	Min.	Aver.	Max.	Min.	Aver.	Max.	Min.	Aver.	Max.	Min.	Aver.	Max.	Time	Time
10	0	0.251	7.160	0	0	0	0	0	0	0	0	0	0.04	8.69
12	0	0.157	2.453	0	0	0	0	0	0	0	0	0	0.34	8.66
15	0	0.038	2.352	0	0	0	0	0.003	0.131	0	0	0	9.74	8.93
17	0	0.093	3.858	0	0	0	0	0	0	0	0	0	107.89	9.49
19	0	0.122	9.108	0	0.042	1.615	0	0	0	0	0	0	2053.50	10.42

Table 6 Solution quality (% deviation from optimal) and time (s) for cost function type G3.

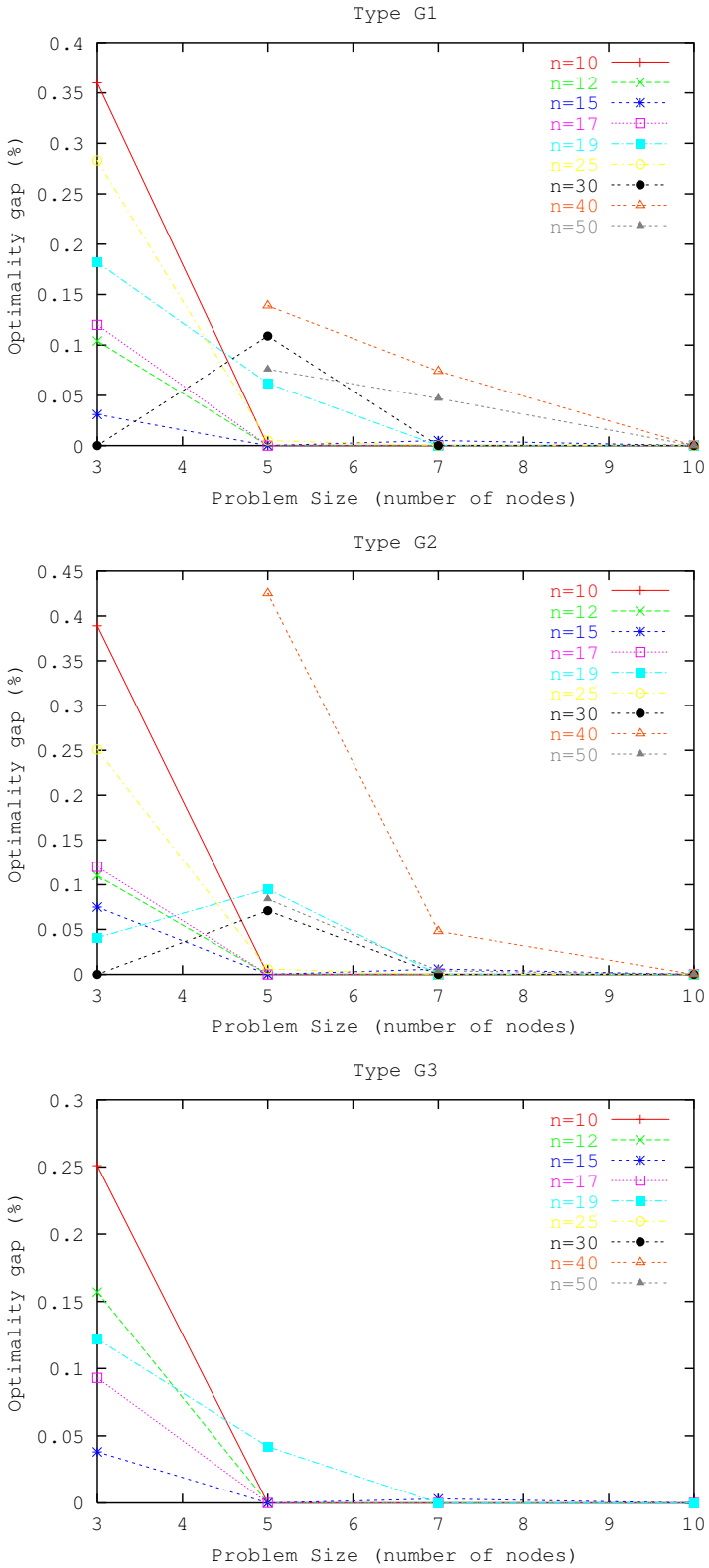


Fig. 12 The effect of the hop parameter value on solution quality, for problem types G1, G2, and G3, respectively.

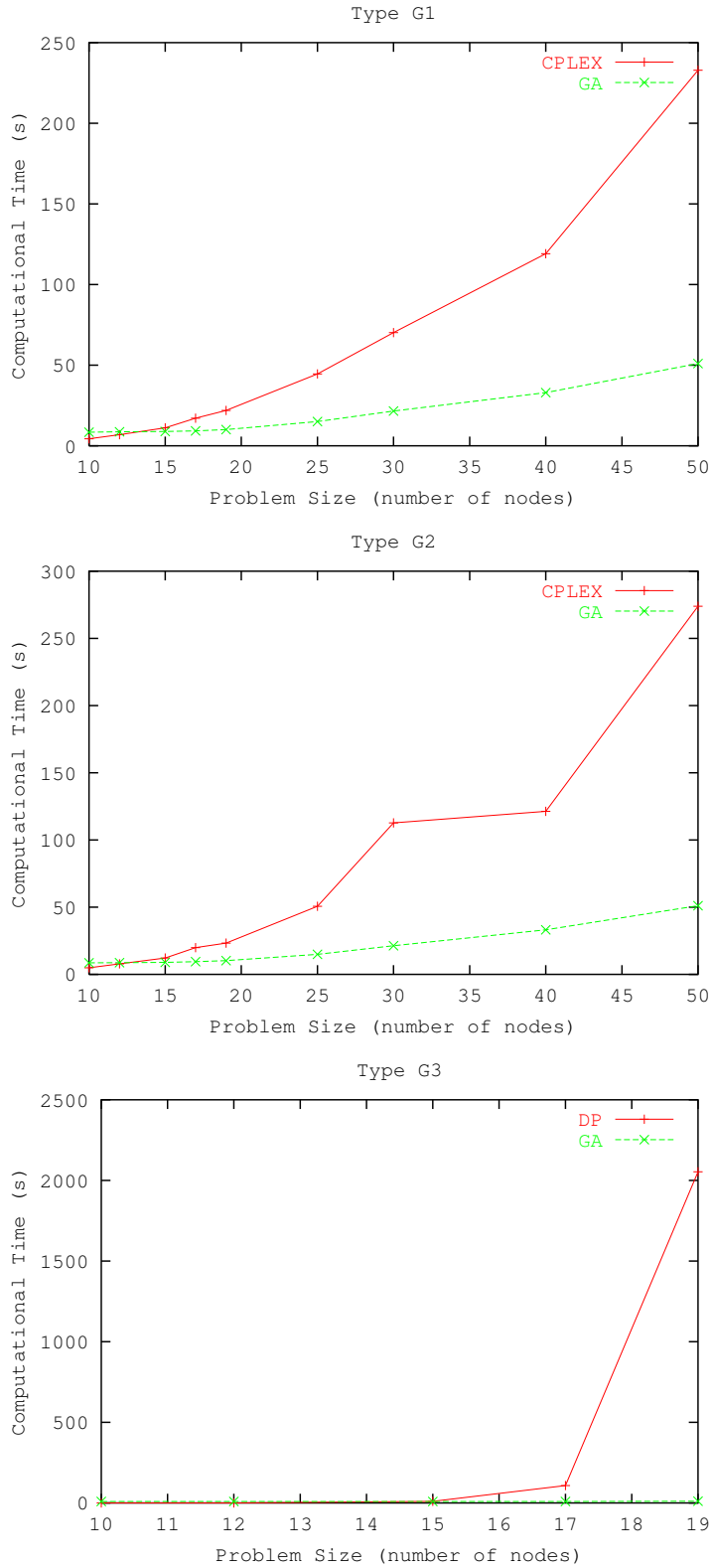


Fig. 13 The effect of problem size on computational time, for problem types G1, G2, and G3, respectively.

Appendix A

```
double NextRnd ( int & PrevSeed )

   unsigned int const a = 16807;

   unsigned int const m = 2147483647;

   unsigned int lo = a * ( PrevSeed & 0xFFFF );

   unsigned int hi = a * ( PrevSeed » 16 );

   lo += ( hi & 0x7FFF ) « 16;

   lo += hi » 15;

   if ( lo > 0x7FFFFFFF ) lo -= 0x7FFFFFFF;
   // the next line is a faster implementation of the previous line

   lo = (lo & 0x7FFFFFFF ) + ( lo » 31);

return ( PrevSeed = lo ) / ( 1.*m );
```