**Optimization Letters manuscript No.**
(will be inserted by the editor)

# The Hop-constrained Minimum Cost Flow Spanning Tree Problem with Nonlinear Costs: an Ant Colony Optimization Approach

**Marta S.R. Monteiro · Dalila B.M.M. Fontes · Fernando A.C.C. Fontes**

Received: date / Accepted: date

**Abstract** In this work we address the Hop-Constrained Minimum cost Flow Spanning Tree (HMFST) problem with nonlinear costs. The HMFST problem is an extension of the Hop-Constrained Minimum Spanning Tree problem since it considers flow requirements other than unit flows. We propose a hybrid heuristic, based on Ant Colony Optimization and on Local Search, to solve this class of problems given its combinatorial nature and also that the total costs are nonlinearly flow dependent with a fixed-charge component. We solve a set of benchmark problems available online and compare the results obtained with the ones reported in the literature for a Multi-Population hybrid biased random key Genetic Algorithm (MPGA). Our algorithm proved to be able to find an optimum solution in more than 75% of the runs, for each problem instance solved, and was also able to improve on many results reported for the MPGA. Furthermore, for every single problem instance we were able to find a feasible solution, which was not the case for the MPGA. Regarding running times, our algorithm improves upon the computational time used by CPLEX and was always lower than that of the MPGA.

**Keywords** Minimum Spanning Tree · Network Flows · Hop-constraints · Nonlinear Costs · Ant Colony Optimization · Local Search

**Mathematics Subject Classification (2000)** MSC 90C27 · MSC 90C59

Marta S. R. Monteiro
Faculdade de Economia and LIAAD-INESC TEC, Universidade do Porto, Rua Dr. Roberto Frias, 4200-464 Porto, Portugal, E-mail: martam@fep.up.pt

Dalila B. M. M. Fontes
Faculdade de Economia and LIAAD-INESC TEC, Universidade do Porto, Tel.: +351-22-0426240, Rua Dr. Roberto Frias, 4200-464 Porto, Portugal, E-mail: fontes@fep.up.pt

Fernando A. C. C. Fontes
Faculdade de Engenharia and ISR-Porto, Universidade do Porto, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal, E-mail: faf@fe.up.pt

## 1 Introduction

The Minimum Spanning Tree (MST) problem is a very well-known combinatorial optimization problem where the objective is to find a tree spanning all nodes in a network while minimizing the total costs incurred. An extension to this problem, that has been considered in the literature, is the Hop-constrained MST (HMST) where a limit on the number of arcs allowed on each path from the root node to any other node is imposed. Here, we consider a further extension to this problem that considers flow requirements, other than unit flows, on each node and nonlinear flow costs on each arc. We term this problem Hop-constrained Minimum cost Flow Spanning Tree problem (HMFST). Different flow requirements at client nodes is a characteristic of several service and distribution networks, such as telecommunications, water, gas, and electrical power. The hop-constraints are used in these networks to guarantee certain levels of reliability or limits on the delays, for example in networks with multidrop lines (where data packets may have to queue before being sent) or in air-mail distribution problems. The nonlinear costs on the arcs, in particular concave costs, are more realistic in the many situations where we have economies of scale.

In this work, we propose a Hybrid Ant Colony Optimization (hybrid ACO, HACO) algorithm to solve the HMFST problem. The use of an heuristic method is adequate, since exact methods can be very expensive in terms of memory requirements and also of computational effort, leading to large running times whenever the problem is solvable. In particular, ACO algorithms were firstly developed to solve hard combinatorial optimization problems [8] and have been successfully applied to solve flow problems with concave cost functions, such as the transportation problem [2] and the minimum cost network flow problem [22]. In addition, many other optimization problems have been solved by ACO algorithms with improved results when compared with other heuristics, such as Genetic Algorithms (GAs), see e.g. [29, 4, 25, 22]. Thus, a priori, we expected ACO to have a competitive performance, in comparison to other heuristic methods already used to solve the HMFST problem.

Works considering HMFST problems are scarce. In [9] dynamic programming is used to solve the HMFST problem. The author considers three distinct nonlinear cost functions with discontinuities other than at the origin and solves 4050 problem instances to optimality. It is also demonstrated that the computational performance is independent of cost function type. Fontes and Gonçalves [11] use a Hybrid Biased Random Key Genetic Algorithm with 3 randomly generated populations evolving separately to solve the same problems. Random keys are used to encode the solution tree. The hop-constraint is handled a posteriori, by penalizing infeasible solutions. Local search is performed by replacing a node in the tree with another node not in the tree, given that the new solution is still an extreme flow. The results were obtained for problems considering spanning trees with up to 50 nodes. We refer the reader to [21, 23] for a literature review on the hop-constrained MST (e.g. [16, 9, 15, 17, 1]) and related problems. See also the works of Tseng et al. [27] and Wang et al. [28] proposing ACO approaches to problems in communication networks which, although related, are different since neither arc flows nor flow requirements are considered.

Our contributions are as follows. Firstly, the application of an ant based algorithm to solve the HMFST problem is, to the best of our knowledge, here proposed for the first time. Secondly, we provide a mathematical programming formulation for the HMFST, which is an adaptation of that of Gouveia and Martins [14]. Thirdly, we use general nonlinear and concave cost functions comprising fixed-charges and variable marginal routing costs. In general, works on HMST use linear costs, for example, [16, 15]. Apart from [9] and [11], this type of functions has never been used before with such problems. We compare the results obtained with the ones provided by CPLEX and with the ones reported in literature [11].

The remainder of this paper is organized as follows. In Section 2, we provide a formal description of the HMFST, along with its mathematical formulation and the cost functions herein considered. In Section 3, we develop our approach to solve the HMFST problem. The results obtained and the subsequent analysis are reported and discussed in Section 4. Finally, in Section 5, we draw some conclusions and point out future work.

## 2 Problem definition and mathematical formulation

Formally, the HMFST problem can be defined as follows. Consider a directed network $G = (N, A)$, where $N$ is the set of $n+1$ nodes, with $n$ demand nodes and one single source node $t$, and $A(\subseteq N \times N \setminus \{t\})$ is the set of $m$ available arcs $(i, j)$. In the HMFST one wishes to minimize the total costs $f_{ij}$ incurred with the network while satisfying demands $d_j$. The total demand of the network, $D$, is given by the summation of all node demands. The commodity flows from the source node $t$ to the $n$ demand nodes $i \in N \setminus \{t\}$. In addition, the maximum number of arcs on a path from the source node to each demand node is constrained by a hop parameter, $H$. The position of an arc in the solution tree, counting from the source node $t$, is represented by an extra index $h$. The mathematical programming model that is given next for the HMFST problem is an adaptation of [14]. Considering the notation summarized below, the model can be written as:

|  |  |  |
|---|---|---|
| $t$ | - | source node, |
| $n$ | - | number of demand nodes, |
| $d_j$ | - | demand of node $j \in N \setminus \{t\}$, |
| $y_{ijh}$ | $=$ | $\begin{cases} 1, \text{ if } x_{ijh} > 0 \\ 0, \text{ if } x_{ijh} = 0, \end{cases}$ |
| $x_{ijh}$ | - | flow on arc $(i, j)$ which is in position $h$, |
| $f_{ij}$ | - | cost in arc $(i, j)$, |

$$\text{min: } \sum_{i \in N} \sum_{j \in N \setminus \{t\}} \sum_{h=1}^{H} f_{ij}(x_{ijh}, y_{ijh}), \tag{1}$$

$$\text{s.t.:} \sum_{i\in N}\sum_{h=1}^{H} y_{ijh} = 1, \quad \forall_{j\in N\setminus\{t\}}, \tag{2}$$

$$\sum_{i\in N} x_{ijh} - \sum_{i\in N\setminus\{t\}} x_{ji,h+1} = d_j \sum_{i\in N} y_{ijh}, \quad \forall_{j\in N\setminus\{t\}},\ \forall_{h\in\{1,\dots,H-1\}}, \tag{3}$$

$$y_{ijh} \leq x_{ijh} \leq D\cdot y_{ijh}, \quad \forall_{i\in N},\ \forall_{j\in N\setminus\{t\}},\ \forall_{h\in\{1,\dots,H\}}, \tag{4}$$

$$x_{ijh} \geq 0, \quad \forall_{i\in N},\ \forall_{j\in N\setminus\{t\}},\ \forall_{h\in\{1,\dots,H\}}, \tag{5}$$

$$y_{ijh} \in \{0,1\}, \quad \forall_{i\in N},\ \forall_{j\in N\setminus\{t\}},\ \forall_{h\in\{1,\dots,H\}}. \tag{6}$$

The objective is to minimize the total costs incurred, i.e. costs associated with the usage of arcs and the costs associated with routing of the flow, as given in (1). Equations (2) guarantee that every node is in the solution in exactly one position. Equations (3) are the balance equations, which in this case also state that if the flow enters a node through an arc in position $h$, then the flow leaving this same node must do it through an arc in position $h+1$. Equations (4) are the coupling constraints and constraints (5) and (6) state the nonnegative and binary nature of the decision variables. It is assumed that the commodity available at the source node $t$ matches the total demand.

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.

Regarding concave cost functions, they can take several forms but the most popular ones used in literature are $b_{ij}\cdot x_{ij} + c_{ij}$ , also known as fixed-charge functions. A square root cost function, $b_{ij}\cdot\sqrt{x_{ij}}$ , has also been considered in [2]. The cost function used in [7] is the $-a_{ij}\cdot x_{ij}^2 + b_{ij}\cdot x_{ij}$, while [10] use $-a_{ij}\cdot x_{ij}^2 + b_{ij}\cdot x_{ij} + c_{ij}$, and [5] define the cost function as $-a_{ij}\cdot x_{ij}^{\alpha} + b_{ij}\cdot x_{ij} + c_{ij}$ with $\alpha$ in [0, 1], just to mention but a few possibilities.

The cost functions considered in this work are as follows:

$$f_{ij}(x_{ij}, y_{ij}) = \begin{cases} -a_{ij}\cdot x_{ij}^2 + b_{ij}\cdot x_{ij} + c_{ij}\cdot y_{ij}, & \text{if } x_{ij} \leq D/2, \\ a_{ij}\cdot x_{ij}^2 + b_{ij}\cdot x_{ij} + (c_{ij}+k)\cdot y_{ij}, & \text{otherwise.} \end{cases} \tag{7}$$

We divide the cost functions in three types: F1, F2, and F3. The functions F1 and F2 consider linear routing costs $b_{ij}$ per unit of flow routed through arc $(i,j)$, as well as, fixed costs $c_{ij}$. Therefore, $a_{ij} = 0$ for F1 and F2. In addition, there is a discontinuity both in F1 and in F2 since $k = b_{ij}$ for F1 and $k = -b_{ij}$ for F2, when the flow passing through arc $(i,j)$ is larger than half the total demand $D$. These cost types correspond, respectively, to the so-called staircase and sawtooth cost functions, see [18], in our case with two segments. The third cost function F3 is represented by complete second-order polynomials and it is initially concave and then convex, the discontinuity point being at half of the total demand. We consider $k = 0$ for F3. The concavity of these cost functions, given by $a_{ij}$ , is defined such that it takes the maximum value guaranteeing that the cost functions are nondecreasing and that an equilibrium between all arcs costs is reached, thus increasing the number of local optima solutions and making the problem harder to solve.

All cost functions consider $a_{ij}, b_{ij}$, and $c_{ij} \in \mathbb{Z}^+$. Note that we have dropped the $h$ index in the flow variables $x_{ij}$ since the cost of an arc does not depend on its position in the tree. The use of these cost functions follows the work in [11], where a more detailed description and motivation for the use of the functions can be found.

## 3 Ant colony optimization approach for the nonlinear HMFST problem

The Ant Colony Optimization metaheuristic is inspired by the foraging behaviour of real ants and was firstly introduced by Dorigo and Stützle [8]. ACO based algorithms became very popular since the beginning and have been used to solve combinatorial problems in several research areas. Although ACO algorithms perform well, there are some advantages in hybridizing them with other metaheuristics, thus benefiting from the joint characteristics obtained, see e.g. [19, 20]. For the interested reader, [6], [13], and [24] provide excellent surveys on ant colony algorithms and their applications.

The performance of an ACO algorithm mainly depends upon the specification of the method chosen to construct the solution, heuristic information, pheromone updating rule, and probability function. A flowchart describing the relation between the main components of the algorithm is given in Fig. 1. In the next sections we provide a description of each of the components.

### 3.1 Defining and constructing a solution to the HMFST problem

Each ant finds a solution for the HMFST problem, i.e. a tree such that every path from the source node $t$ to every demand node has at most $H$ arcs.

All ants begin their solution construction at the source node. Initially, an ant selects an existing arc linking the source node $t$ and a demand node. Then, at each iteration the ant selects another arc, from the set of available arcs, linking a node already in the partial solution (the source node or one of the demand nodes) to a demand node not yet considered, and adds it to the tree under construction.

Each time an arc $(k, j)$ is added to the partial solution the length $l_j$ of the path linking the demand node $j$ to the source node $t$ is computed. If $l_j = H$, the maximum length (number of hops) allowed has been reached in that path and thus the arcs considering node $j$ as a parent are excluded from the set of viable arcs. New arcs are added until all the demand nodes are in the solution, or until no viable arcs are available. Unfeasible solutions are discarded.

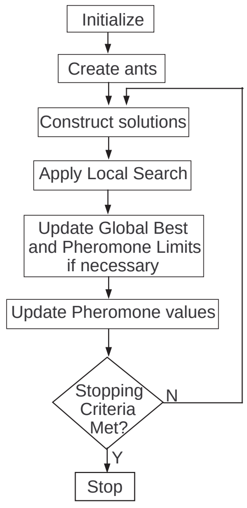

**Fig. 1** Flowchart for the proposed ACO algorithm.

Each arc entering the solution is chosen accordingly to the probability function defined by

$$P_{ij} = \frac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{(i,j)\in A} [\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}, \tag{8}$$

where $\tau_{ij}$ is the pheromone in arc $(i,j)$ at the current iteration; $\eta_{ij}$ is the visibility of arc $(i,j)$, represented as the inverse of the cost of the arc; and $\alpha, \beta > 0$ are parameters weighting the relative importance of $\tau_{ij}$ and $\eta_{ij}$, respectively.

## 3.2 Updating and bounding pheromone values

After all ants have constructed their solutions, the best solution of the current iteration $S^i$ is identified and only the ant that has constructed $S^i$ is allowed to reinforce the pheromone quantity in the arcs of its solution, using the rule

$$\tau_{ij} = (1-\rho) \times \tau_{ij} + \frac{Q}{F(S^i)}, \tag{9}$$

where $\rho \in ]0,1]$ is the pheromone evaporation rate, $\tau_{ij}$ is the pheromone quantity in arc $(i,j)$, $Q$ is a positive proportionality parameter and $F(S^i)$ is the cost of $S^i$.

In the initialization phase of the algorithm, each arc is given an equal amount of pheromone $\tau_0$.

Following the work of [26], pheromone values are bounded to be in the interval

$$[\tau_{min}, \tau_{max}] = \left[\frac{\tau_{max} \cdot (1 - \sqrt[n]{p_{best}})}{(\frac{n}{2} - 1) \cdot \sqrt[n]{p_{best}}}, \frac{1}{\rho \cdot F^*}\right], \tag{10}$$

where $F^*$ is the cost of the best solution found so far and $p_{best}$ is the probability of constructing the best solution. Whenever a pheromone value is bellow $\tau_{min}$, it is set to $\tau_{min}$ and whenever it is above $\tau_{max}$, it is set to $\tau_{max}$.

To deal with stagnation and cycling of solutions, the pheromone matrix is restarted, i.e. its values are reset to $\tau_0$, whenever the number of iterations for which the incumbent solution has not changed reaches a predefined number.

### 3.3 Local Search

Right after the ants have constructed a feasible solution, the Local Search procedure developed is applied to a set of solutions $W$, consisting of the best five solutions obtained in each iteration. For this problem, a solution $S^N$ is a neighbour solution of solution $S$ if $S^N$ is obtained from $S$ by swapping an arc $(i, j) \in S$ with another arc $(l, j) \notin S$ that does not create a cycle and still ensures no node is more than $H$ arcs away from the source node.

The algorithm starts by sorting the arcs in the selected solution $S \in W$ in ascending order of their pheromone value. For each arc $(i, j)$, the algorithm finds all the arcs $(l, j)$ that can replace it while maintaining feasibility of the solution. The algorithm attempts to replace the original arc $(i, j)$ by starting with the arcs with a higher pheromone value. If one of the replacements improves the cost of the solution, a new solution $S^N$ is obtained and the algorithm proceeds to the next arc $(i, j)$ in $S$ without attempting the remaining options. The local search algorithm ends when all solutions $S \in W$ have been tested for improvement.

## 4 Computational Experiments

In order to test the proposed algorithm we downloaded the Euclidean test set available from [3]. The number of nodes considered in the problems ranges from 10 up to 50. For further details on these problems we refer [12]. There is a total of 240 problems to be solved for each of the three cost functions considered F1, F2, and F3 and each of the four $H$ values, where $H \in \{3, 5, 7, 10\}$. It should be noticed that from the 240 available problems, 75 problems have no feasible solution for $H = 3$ and 7 problems have no feasible solution for $H = 5$. In total, we have 2634 problem instances to solve.

In addition, we have generated 15 larger size problems with 60 and 80 nodes, which were solved also for the 3 cost functions and the 4 $H$ values, resulting in further 360 problem instances. All 2994 problem instances were solved 10 times.

Regarding the HACO parameter values, they have been empirically chosen to be $\alpha = 1$, $\beta = 2$, $Q = 2$, $p_{best} = 0.5$, $\eta_{ij} = \frac{1}{c_{ij}+b_{ij}}$, $2n$ ants, and $\tau_0 =$

$1,000,000$. Two stopping conditions have been defined for this algorithm. The algorithm stops running when 2000 iterations are reached or three consecutive restarts of the pheromone matrix have been performed, whichever happens first.

## 4.1 Comparing our results with the ones in literature

In this section, we present the computational results that were obtained with the HACO algorithm and with the CPLEX 12.0. Also, we compare these results with the results reported in the literature for the same problems [11]. Our algorithm was implemented in Java and all our computational experiments (HACO and CPLEX) were carried out on a PC with a Pentium D at 3.20GHz and 1GB of RAM, which was also used for the results reported by Fontes and Gonçalves [11].

We characterise solutions regarding time, in seconds, required to perform a full run of the algorithm, and optimality gap in %, which is given as

$$\text{Gap}(\%) = \frac{HS - OptS}{OptS} \times 100,$$

where OptS stands for the optimum solution, when available, and the best currently known otherwise; and the HS stands for the best solution found with the heuristic in question. Each problem instance was solved 10 times. The HMFST problem was solved with CPLEX only for F1 and F2 cost functions. Nonetheless, Fontes [9] provides optimal solution results for cost functions of type F3 and problems with up to 19 nodes. For larger problems we have used the results obtained by the Multi-Population hybrid biased random key Genetic Algorithm (MPGA) developed in [11]. In this case, the optimality gap is calculated by comparing the results obtained by each of the heuristics with the currently best solutions (lowest cost solution found by HACO or by MPGA).

Table 1 presents Average (Avg) and Maximum (Max) gap values obtained with the HACO algorithm and with the aforementioned MPGA, grouped by hop value and cost function. The HACO and MPGA performance achieved for $H = 10$ is of zero gap for all cost functions considered. Therefore, we do not present gap results for $H = 10$ in Table 1. For $H = 3$, the HACO always finds an optimum solution , regardless of the cost function type and problem size. The same can not be said about the MPGA, which presents the worst gap values precisely for $H = 3$. In particular, the largest gap value, over 17%, was obtained for a problem with 19 nodes and considering cost function F1. Problems with $H = 5$ and $H = 7$ proved to be the most difficult ones to solve by HACO. Nevertheless, the largest gap value obtained amongst all problems is only 0.58% ($N = 50$,F3, $H = 5$).

The type of the cost function seems to bear no influence on the performance of our algorithm. The HACO always improves upon the MPGA results, except for one single problem with 17 nodes. Furthermore, in every run HACO has

**Table 1** HACO and MPGA optimality gap (%) for F1, F2 and F3 cost functions with $H = 3, 5$, and 7.

| | H=3 | | | | H=5 | | | | H=7 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HACO | | MPGA | | HACO | | MPGA | | HACO | | MPGA | |
| N | Avg | Max | Avg | Max | Avg | Max | Avg | Max | Avg | Max | Avg | Max |
| F1 | | | | | | | | | | | | |
| 10 | 0 | 0 | 0.36 | 7.16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | 0 | 0 | 0.10 | 1.90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | 0 | 0 | 0.03 | 0.64 | 0 | 0 | 0 | 0 | 0 | 0 | 0.01 | 0.26 |
| 17 | 0 | 0 | 0.12 | 3.74 | 0.01 | 0.46 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0.18 | 17.35 | 0 | 0 | 0.06 | 4.92 | 0 | 0 | 0 | 0 |
| 25 | 0 | 0 | 0.28 | 6.14 | 0 | 0 | 0.01 | 0.16 | 0 | 0 | 0 | 0 |
| 30 | 0 | 0 | 0 | 0 | 0.002 | 0.08 | 0.11 | 4.08 | 0 | 0 | 0 | 0 |
| 40 | | | | | 0 | 0 | 0.14 | 7.50 | 0.0004 | 0.02 | 0.074 | 1.95 |
| 50 | | | | | 0.01 | 0.51 | 0.08 | 0.99 | 0.001 | 0.05 | 0.047 | 1.43 |
| F2 | | | | | | | | | | | | |
| 10 | 0 | 0 | 0.39 | 7.16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | 0 | 0 | 0.11 | 1.91 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | 0 | 0 | 0.08 | 2.40 | 0 | 0 | 0 | 0 | 0 | 0 | 0.01 | 0.23 |
| 17 | 0 | 0 | 0.12 | 3.74 | 0.01 | 0.46 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0.04 | 1.79 | 0 | 0 | 0.10 | 4.92 | 0 | 0 | 0 | 0 |
| 25 | 0 | 0 | 0.25 | 6.99 | 0 | 0 | 0.01 | 0.18 | 0 | 0 | 0 | 0 |
| 30 | 0 | 0 | 0 | 0 | 0.002 | 0.08 | 0.07 | 4.08 | 0 | 0 | 0 | 0 |
| 40 | | | | | 0 | 0 | 0.43 | 13.91 | 0.001 | 0.02 | 0.05 | 1.95 |
| 50 | | | | | 0 | 0 | 0.08 | 0.99 | 0.004 | 0.05 | 0.004 | 0.24 |
| F3 | | | | | | | | | | | | |
| 10 | 0 | 0 | 0.25 | 7.16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | 0 | 0 | 0.16 | 2.45 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | 0 | 0 | 0.04 | 2.35 | 0 | 0 | 0 | 0 | 0 | 0 | 0.003 | 0.13 |
| 17 | 0 | 0 | 0.09 | 3.86 | 0.01 | 0.48 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0.12 | 9.11 | 0 | 0 | 0.04 | 1.62 | 0 | 0 | 0 | 0 |
| 25 | 0 | 0 | 0.29 | 6.96 | 0 | 0 | 0.004 | 0.15 | 0 | 0 | 0 | 0 |
| 30 | 0 | 0 | 0.11 | 2.66 | 0.01 | 0.33 | 0.15 | 4.09 | 0 | 0 | 0 | 0 |
| 40 | | | | | 0 | 0 | 0.12 | 7.65 | 0.001 | 0.08 | 0.08 | 1.98 |
| 50 | | | | | 0.02 | 0.58 | 0.08 | 0.58 | 0.01 | 0.22 | 0.05 | 1.23 |

always been able to find a feasible solution, when one exists. This is not the case for the MPGA, since it failed to do so for every run of 19 problem instances (see the results reported in [11]). In addition, the robustness of the HACO can be inferred from the fact that the worst gaps found are very small. The worst gap found, amongst the 26340 calculated, is only 0.58%. We recall that we have solved 2634 problem instances, 10 times each.

Running time results are given in Table 2 for CPLEX and HACO, and in Table 3 for MPGA and HACO. The times reported for CPLEX, to find an optimum solution, clearly increase both with size and with hop value. As it

**Table 2** Computational time results, for CPLEX and HACO, obtained for cost functions F1, F2 and F3 and $H = 3, 5, 7$ and 10.

| | F1 | | | | F2 | | | | F3 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| N | 3 | 5 | 7 | 10 | 3 | 5 | 7 | 10 | 3 | 5 | 7 | 10 |
| HACO | | | | | | | | | | | | |
| 10 | 0.3 | 0.5 | 0.5 | 0.5 | 0.3 | 0.5 | 0.6 | 0.5 | 0.3 | 0.7 | 0.9 | 0.9 |
| 12 | 0.5 | 0.7 | 0.9 | 0.8 | 0.5 | 0.7 | 0.9 | 0.9 | 0.8 | 1.3 | 1.6 | 1.8 |
| 15 | 0.7 | 1.2 | 1.5 | 1.3 | 0.8 | 1.2 | 1.5 | 1.5 | 1.3 | 1.5 | 2.5 | 2.5 |
| 17 | 1.0 | 2.0 | 2.1 | 1.9 | 1.2 | 1.9 | 2.1 | 2.3 | 1.0 | 1.8 | 2.2 | 2.3 |
| 19 | 1.3 | 2.2 | 2.7 | 3.0 | 1.4 | 2.2 | 2.7 | 2.9 | 2.9 | 3.7 | 3.7 | 4.2 |
| 25 | 2.8 | 4.7 | 5.4 | 5.8 | 2.9 | 4.4 | 5.3 | 5.7 | 2.6 | 4.1 | 4.9 | 4.8 |
| 30 | 4.6 | 9.1 | 8.9 | 9.8 | 4.6 | 8.7 | 8.6 | 9.5 | 3.9 | 6.7 | 8.6 | 8.4 |
| 40 | | 15.0 | 26.3 | 24.3 | | 15.0 | 25.3 | 23.4 | | 14.2 | 19.6 | 20.3 |
| 50 | | 35.5 | 50.7 | 47.3 | | 32.4 | 68.0 | 48.0 | | 27.8 | 48.9 | 47.1 |
| CPLEX | | | | | | | | | | | | |
| 10 | 1.2 | 1.9 | 3.4 | 4.3 | 2.2 | 2.0 | 2.9 | 3.3 | - | - | - | - |
| 12 | 1.9 | 2.9 | 5.6 | 6.8 | 2.6 | 2.6 | 4.7 | 5.5 | - | - | - | - |
| 15 | 2.8 | 5.2 | 8.8 | 11.1 | 3.4 | 4.3 | 7.2 | 8.5 | - | - | - | - |
| 17 | 3.8 | 7.8 | 11.8 | 16.9 | 4.1 | 6.4 | 10.1 | 21.7 | - | - | - | - |
| 19 | 5.3 | 8.3 | 13.9 | 21.8 | 5.6 | 7.4 | 15.7 | 24.1 | - | - | - | - |
| 25 | 9.7 | 17.6 | 28.9 | 44.1 | 11.0 | 13.5 | 33.3 | 52.2 | - | - | - | - |
| 30 | 14.3 | 29.1 | 47.2 | 70.2 | 16.5 | 29.4 | 55.7 | 70.9 | - | - | - | - |
| 40 | | 65.2 | 113.1 | 172.4 | | 60.5 | 139.4 | 239.3 | - | - | - | - |
| 50 | | 161.0 | 225.9 | 288.5 | | 155.5 | 275.3 | 367.4 | - | - | - | - |

can be seen, CPLEX requires a computational time which is 4 to 5 times that of HACO.

The results in Table 3 show no significant difference between the MPGA and the HACO time performance. However, HACO is slightly faster than MPGA and also finds better solutions than those of MPGA.

**Table 3** Average computational times obtained with HACO and MPGA for F1, F2 and F3 cost functions.

| | F1 | | F2 | | F3 | |
|---|---|---|---|---|---|---|
| N | HACO | MPGA | HACO | MPGA | HACO | MPGA |
| 10 | 0.5 | 8.5 | 0.5 | 8.6 | 0.7 | 8.7 |
| 12 | 0.7 | 8.7 | 0.7 | 8.7 | 1.3 | 8.7 |
| 15 | 1.2 | 8.9 | 1.3 | 8.9 | 2.0 | 8.9 |
| 17 | 1.8 | 9.3 | 1.9 | 9.4 | 1.8 | 9.5 |
| 19 | 2.3 | 10.1 | 2.3 | 10.2 | 3.6 | 10.4 |
| 25 | 4.7 | 15.0 | 4.6 | 14.9 | 4.1 | 15.7 |
| 30 | 8.1 | 21.5 | 7.9 | 21.3 | 6.9 | 22.9 |
| 40 | 21.9 | 33.0 | 21.2 | 33.1 | 18.1 | 35.4 |
| 50 | 44.5 | 51.0 | 49.5 | 51.1 | 41.3 | 50.6 |

In order to infer about the evolution of CPLEX and HACO computational times, as well as about the HACO gap performance, we have generated 15 larger size problem instances with 60 and 80 nodes, where we have considered a larger number of arcs in the networks, since otherwise there would not be any feasible solutions for small hop values. These problem instances were solved (by the HACO) 10 times each. Although HACO was able to find a feasible

**Table 4** Average optimality gap (%) and computational time results for CPLEX and HACO for problems with 60 and with 80 nodes, cost functions F1 and F2 and $H = 3, 5, 7$ and 10.

| | | | | | | Time (sec.) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | GAP | | | | CPLEX | | | | HACO | | | |
| F | N | 3 | 5 | 7 | 10 | 3 | 5 | 7 | 10 | 3 | 5 | 7 | 10 |
| F1 | 60 | 0 | 0 | 0 | - | 67.8 | 289.6 | 368.3 | - | 16.2 | 29.5 | 35.4 | 42.8 |
| | 80 | 0.001 | 0 | - | - | 197.7 | 541.1 | - | - | 52.1 | 88.5 | 109.1 | 117.0 |
| F2 | 60 | 0.001 | 0 | 0 | 0 | 103.2 | 635.6 | 336.8 | 596.7 | 17.2 | 29.7 | 35.8 | 49.1 |
| | 80 | 0.001 | 0 | 0 | - | 607.3 | 549.4 | 943.3 | - | 53.4 | 87.2 | 131.7 | 122.0 |

solution to all problem instances with 60 and with 80 nodes, the CPLEX was not, due to memory failure. For further details see [23]. However, as it can be seen in Table 4, for the problem instances optimally solved by the CPLEX, the HACO maintains its lower average optimality gaps.

Regarding computational running times, it is now notorious that running times increase with the hop parameter value, although for the CPLEX the rate of increase is much larger. In addition, it can be seen that the HACO can be up to 10 times faster. Finally, these new problem instances confirm the conclusion already stated that the CPLEX is influenced by the type of cost function whereas the HACO is not.

In order to better understand the results obtained with CPLEX, regarding running times, we have performed two extra experiments. CPLEX can, sometimes, obtain good or even very good results without running to optimality. Therefore, we performed 2 other computational experiments where CPLEX is interrupted after either a time limit or an optimality gap bound has been reached. In the first experiment (E1), we allow the CPLEX to run 15 seconds for problems with up to 30 nodes, 70 seconds for problems with 40, 50 and 60 nodes, and 150 seconds for problems with 80 nodes, i.e. we give CPLEX a time limit which is larger than the maximum average running times required by the HACO. In the second experiment (E2), we give to the CPLEX a bound on the optimality gap (1%), that is the CPLEX is allowed to run until it finds a solution which is within 1% of the optimal solution previously found by CPLEX when running unrestricted.

The results obtained for each problem size and each cost function type can be seen in Table 5. Columns Avg and Max provide the average and maximum optimality gaps, respectively. Column CT gives the number of problem instances for which CPLEX was able, within the time limit, to find a feasible solution, expressed as a percentage of the number of problem instances

**Table 5** Results obtained by bounding CPLEX in time or in gap tolerance, and considering F1 and F2 cost functions.

| | E1 | | | | | | E2 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F1 | | | F2 | | | F1 | | | F2 | | |
| N | Avg | Max | CT (%) | Avg | Max | CT (%) | Avg | Max | T (%) | Avg | Max | T (%) |
| 10 | 0 | 0 | 100 | 0 | 0 | 100 | 0.01 | 0.64 | 54 | 0.004 | 0.18 | 59 |
| 12 | 0 | 0 | 100 | 0 | 0 | 100 | 0.02 | 0.76 | 49 | 0.04 | 0.81 | 54 |
| 15 | 0.02 | 2.87 | 100 | 0.03 | 3.87 | 100 | 0.04 | 0.99 | 47 | 0.01 | 0.79 | 52 |
| 17 | 0.21 | 5.61 | 100 | 0.25 | 6.21 | 100 | 0.05 | 0.96 | 46 | 0.01 | 0.56 | 46 |
| 19 | 0.29 | 8.12 | 90 | 0.32 | 5.66 | 86 | 0.04 | 0.75 | 47 | 0.01 | 0.40 | 41 |
| 25 | 0.51 | 6.71 | 46 | 0.51 | 15.27 | 44 | 0.07 | 0.86 | 45 | 0.05 | 0.72 | 41 |
| 30 | 0 | 0 | 12 | 0 | 0 | 12 | 0.08 | 0.66 | 43 | 0.03 | 0.82 | 33 |
| 40 | 0.04 | 1.72 | 100 | 0.01 | 0.51 | 100 | 0.11 | 0.93 | 52 | 0.09 | 0.52 | 38 |
| 50 | 0.75 | 16.20 | 63 | 0.39 | 5.67 | 98 | 0.02 | 0.23 | 40 | 0.02 | 0.66 | 31 |
| 60 | 0 | 0 | 16 | 1.65 | 10.96 | 39 | 0.04 | 0.37 | 50 | 0.04 | 0.49 | 26 |
| 80 | 0.00 | 0.02 | 88 | 0.08 | 0.95 | 81 | 0.28 | 0.99 | 43 | 0.07 | 0.36 | 22 |

solved by the CPLEX when running unrestricted. Column T gives the average computational time required to obtain an optimality gap of 1% or less, as a percentage of the average computational time required to find an optimal solution.

From the results in E1, we can observe that CPLEX cannot find a feasible solution for a considerable number of problem instances with known feasible solutions, as high as 88%. Furthermore, the optimality gap for the time restricted CPLEX is considerably larger than that of HACO. For example, in the case of problems with 50 nodes and cost function F1 there are $15 \times 4$ problem instances out of which the unrestricted CPLEX was able to solve 40 (for the others there are no feasible solutions). From these, the time limited CPLEX was able to find a feasible solution to 25 (63%). The 250 feasible solutions obtained (each of the 25 problem instances was solved 10 times) have optimality gaps ranging from 0% to 16.20% with an average of 0.75%.

Regarding E2, we can conclude that in order to be within 1% of the optimality gap, CPLEX takes about half the time needed to find a proved optimal solution. However, HACO continues to have a better performance regarding both time and optimality gap. For example, in the case of problems with 40 nodes and cost function F1, CPLEX was able to find an optimal solution to 43 problem instances of the 60 available ones (for the others there are no feasible solutions). The average time required to do so was 119.31 seconds. However, when run with an optimality gap bound of 1%, CPLEX finds (430) solutions having an optimality gap ranging from 0% to 0.93% with an average of 0.11%. The average time required to find such gaps was 62.04 seconds, which is about 52% of the time required to find optimal solutions (119.31 seconds). For more details and results refer to [23].

## 5 Conclusions

In this work, the Hop-constrained Minimum cost Flow Spanning Tree problem with nonlinear costs is addressed by a hybrid algorithm based on Ant Colony Optimization and on Local Search. The cost functions are of three types, and they comprise both fixed-charge and routing costs. We compared our results with the ones reported in the literature and our solutions were always better or as good as the ones provided by current literature, except for 1 problem. Furthermore, our algorithm was always able to find a feasible solution, when there was one, whereas CPLEX and the MPGA were not. In fact, for 19 problem instances, the MPGA failed to find any feasible solution, while for another 13 problem instances only in some runs a feasible solution was found. Regarding CPLEX, it cannot solve problems with cost function F3 and cannot solve most of the larger problems. In addition, when a time limit similar to the time used by HACO is given, CPLEX cannot find a feasible solution for most problems. When a bound of 1% is imposed to the optimality gap, HACO is still faster and finds better solutions than CPLEX.

The algorithm has shown to be very robust since, 1) although several cost functions with different complexity have been used, its performance has not been affected. Both solution quality and computational time remain of the same magnitude. 2) the variance of the solution quality is quite low and the worst optimality gap, out of the 26340 calculated, is only 0.58%. Therefore, the proposed HACO heuristic proved to be a good alternative method to solve HMFST problems, having demonstrated a better performance over the existing heuristic [11] regarding gap values and over both this heuristic and CPLEX regarding time.

The quality of the results obtained has encouraged us to extend the scope of application of our HACO to other network flow problems in future work.

**Acknowledgments**

This work is partially funded by ERDF (FEDER), the COMPETE through the FCT as part of projects PTDC/EGE-GES/099741/2008 and PTDC/EEA-CRO/116014/2009 and the North Portugal Regional Operational Programme (ON.2O Novo Norte) under the National Strategic Reference Framework.

## References

1. Akgun, I., Tansel, B.C.: New formulations of the hop-constrained minimum spanning tree problem via miller-tucker-zemlin constraints. Eur J Oper Res **212**, 263–276 (2011)
2. Altiparmak, F., Karaoglan, I.: A genetic ant colony optimization approach for concave cost transportation problems. In: Evolutionary Computation, 2007. CEC 2007. IEEE Congress on, pp. 1685–1692 (2007)
3. Beasley, J.: Or-library. http://www.brunel.ac.uk/deps/ma/research/jeb/orlib/netflowccinfo.html (2010)
4. Bin, Y., Zhong-Zhen, Y., Baozhen, Y.: An improved ant colony optimization for vehicle routing problem. Eur J Oper Res **196**, 171–176 (2009)

5. Burkard, R.E., Dell'Amico, M., Martello, S.: Assignment problems. Siam (2009)
6. Cordon, O., Herrera, F., Stützle, T.: A review on the ant colony optimization metaheuristic: Basis, models and new trends. Mathw Soft Comput **9**, 141–175 (2002)
7. Dang, C., Sun, Y., Wang, Y., Yang, Y.: A deterministic annealing algorithm for the minimum concave cost network flow problem. Neural Networks **24**, 69–708 (2011)
8. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge, MA (2004)
9. Fontes, D.B.M.M.: Optimal hop-constrained trees for nonlinear cost flow networks. Infor **48**, 13–21 (2010)
10. Fontes, D.B.M.M., Gonçalves, J.F.: Heuristic solutions for general concave minimum cost network flow problems. Networks **50**, 67–76 (2007)
11. Fontes, D.B.M.M., Gonçalves, J.F.: A multi-population hybrid biased random key genetic algorithm for hop-constrained trees in nonlinear cost flow networks. Optim Lett **7**, 1303–1324 (20123)
12. Fontes, D.B.M.M., Hadjiconstantinou, E., Christofides, N.: Upper bounds for single-source uncapacitated concave minimum-cost network flow problems. Networks **41**(4), 221–228 (2003)
13. García-Martínez, C., Cordón, O., Herrera, F.: A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. Eur J Oper Res **180**(1), 116–148 (2007)
14. Gouveia, L., Martins, P.: The capacitated minimal spanning tree problem: an experiment with a hop-indexed model. Ann Oper Res **86**, 271–294 (1999)
15. Gouveia, L., Paias, A., Sharma, D.: Restricted dynamic programming based neighborhoods for the hop-constrained minimum spanning tree problem. J Heuristics **17**, 23–37 (2011)
16. Gouveia, L., Requejo, C.: A new lagrangean relaxation approach for the hop-constrained minimum spanning tree problem. Eur J Oper Res **132**, 539–552 (2001)
17. Gouveia, L., Simonetti, L., Uchoa, E.: Modeling hop-constrained and diameter-constrained minimum spanning tree problems as steiner tree problems over layered graphs. Math Prog **128**, 123–148 (2011)
18. Kim, D.: Piecewise linear network flow problems. In: Floudas, C.A., Pardalos, P.M. (eds.) Encyclopedia of Optimization. Kluwer Academic Publisher (2003)
19. Kiran, M.S., Gunduz, M., Baykan, O.K.: A novel hybrid algorithm based on particle swarm and ant colony optimization for finding the global minimum. Appl Math Comput **219**, 1515–1521 (2012)
20. Liao, T.W., Kuo, R.J., Hu, J.T.L.: Hybrid ant colony optimization algorithms for mixed discrete-continuous optimization problems. Appl Math Comput **219**, 3241–3252 (2012)
21. Monteiro, M.S.R.: Ant colony optimization algorithms to solve nonlinear network flow problems. Ph.D. thesis, Faculdade de Economia da Universidade do Porto, Porto, Portugal (2012)
22. Monteiro, M.S.R., Fontes, D.B.M.M., Fontes, F.A.C.C.: Concave minimum cost network flow problems solved with a colony of ants. J Heuristics **19**, 1–33 (2013)
23. Monteiro, M.S.R., Fontes, D.B.M.M., Fontes, F.A.C.C.: Solving hop-constrained MST problems with ACO (2013). Working paper, N493, Universidade do Porto, Faculdade de Economia
24. Mullen, R., Monekosso, D., Barman, S., Remagnino, P.: A review of ant algorithms. Expert Syst Appl **36**, 9608–9617 (2009)
25. Putha, R., Quadrifoglio, L., Zechman, E.: Comparing ant colony optimization and genetic algorithm approaches for solving traffic signal coordination under oversaturation conditions. Comput-Aided Civ Infrastruct Eng **27**, 14–28 (2012)
26. Stützle, T., Hoos, H.: Max-min ant system and local search for the traveling salesman problem. In: IEEE International Conference On Evolutionary Cmputation (ICEC'97), pp. 309–314. IEEE Press, Piscataway,NJ (1997)
27. Tseng, S., Lin, C., Huang, Y.: Ant colony-based algorithm for constructing broadcasting tree with degree and delay constraints. Expert Syst Appl **35**, 1473–1481 (2008)
28. Wang, H., Shi, Z., Li, S.: Multicast routing for delay variation bound using a modified ant colony algorithm. J Network Comput Appl **32**, 258–272 (2008)
29. Yin, P.Y., Wang, J.Y.: Ant colony optimization for the nonlinear resource allocation problem. Appl Math Comput **174**, 1438–1453 (2006)