FEP WORKING PAPERS

FEP WORKING PAPERS

Research Work in Progress

n. 493 May 2013

ISSN: 0870-8541

Solving Hop-constrained MST problems with ACO

Marta S.R. Monteiro [1,2]
Dalila B.M.M. Fontes [1,2]
Fernando A.C.C. Fontes [3,4]

[1] FEP-UP, School of Economics and Management, University of Porto
[2] LIAAD/INESC TEC
[3] FEUP-UP, Faculty of Engineering, University of Porto
[4] ISR-Porto

U.PORTO FEP ECONOMICS AND MANAGEMENT

Solving Hop-constrained MST problems with ACO[*]

Marta S.R. Monteiro[1][†], Dalila B.M.M. Fontes[1], Fernando A.C.C. Fontes[2]

[1] Faculdade de Economia and LIAAD-INESC TEC

Universidade do Porto

Rua Dr. Roberto Frias, 4200-464 Porto, Portugal.

E-mail: martam@fep.up.pt;fontes@fep.up.pt

Tel.: +351-22-0426240

[2] Faculdade de Engenharia da Universidade do Porto and ISR-Porto

Rua Dr. Roberto Frias, 4200-464 Porto, Portugal.

E-mail: faf@fe.up.pt

Abstract

The Hop-constrained Minimum cost Flow Spanning Tree (HMFST) problem is an extension of the Hop-Constrained Minimum Spanning Tree problem since it considers flow requirements other than unit flows. Given that we consider the total costs to be nonlinearly flow dependent with a fixed-charge component and given the combinatorial nature of this class of problems, we propose a heuristic approach to address them. The proposed approach is a hybrid metaheuristic based on Ant Colony Optimization (ACO) and on Local Search (LS). In order to test the performance of our algorithm we have solved a set of benchmark problems and compared the results obtained with the ones reported in the literature for a Multi-Population Genetic Algorithm (MPGA). We have also compared our results, regarding computational time, with those of CPLEX. 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 nor for CPLEX. Regarding running times, our algorithm improves upon the computational time used by CPLEX and was always lower than that of the MPGA.

[*]This work is funded by the ERDF through the Programme COMPETE and by the Portuguese Government through FCT - Foundation for Science and Technology, projects PTDC/EGE-GES/099741/2008 and PTDC/EEA-CRO/116014/2009.

[†]Corresponding author.

Keywords: Ant Colony Optimization, Nonlinear Costs, Hybrid, Local Search, Minimum Spanning Tree Problem, Hop-constraints
JEL Classifications: C61, C44.

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. This combinatorial problem is frequently used to model several applications, specially in the area of telecommunications, where there is a central device, for example a hub, that must be linked to a set of remote terminals. This is the case of Frey et al (2008) and Hwang et al (2007) that use the MST to model multicast networks.

An extension to the MST problem that limits the number of arcs allowed on each path from the root node to any other node is called the Hop-constrained Minimum Spanning Tree (HMST) problem. The addition of a maximum number of arcs in each path is, usually, related to reliability issues. Furthermore, these constraints, called Hop-constraints, can also be associated to lower delay times in a multi-drop lines network, where packages of information may have to queue before reaching their destination. Although in its simplest version the MST problem can be solved in polynomial time the Hop-constrained version is NP-Hard Gouveia (1995). Here, however, we look into another generalization of the MST problem since we consider that flow requirements at client nodes can have values other than the unit and be different across clients. Different flow requirements at client nodes is a characteristic of several service networks, such as telecommunications, water, gas, and electrical power. Furthermore, the cost functions considered involve two components: a setup or fixed cost incurred by using the arc and a routing cost nonlinearly dependent on the flow being routed through the arc.

In this work, we propose a Hybrid Ant Colony Optimization (HACO) algorithm to solve the Hop-constrained Minimum cost Flow Spanning Tree (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. The choice of a population based method has been motivated by an increasing interest in recent metaheuristics, such as Ant Colony Optimization, Swarm Optimization, or Genetic Algorithms, that by themselves or hybridized have been known to have the best results to the moment for some problems, Talbi (2002). In particular, ACO algorithms were firstly developed to solve hard combinatorial optimization problems Dorigo and Stützle (2004); Dorigo and Blum (2005), being initially applied to solve the well-known NP-Hard Travelling Salesman Problem. ACO algorithms have also been successfully applied to solve flow problems with concave cost functions, as is the case of the Transportation Problem Altiparmak and Karaoglan

(2007) and of the Minimum Cost Network Flow Problem Monteiro et al (2011). The HMFST problem can be viewed as the problem of finding shortest paths (in the sense of the least cost paths) between a source node and every single demand node in a network having into account the flow to be routed between pairs of nodes and the limitation on the number of allowed arcs in each path. Therefore, even before starting this research, we would expect that ACO would have a good performance also while solving the HMFST problem. This is because all these three characteristics can be found in them, flow between pairs of nodes, nonlinear costs and shortest paths[1]. In addition, we are able to cite several other optimization problems that have been solved by ACO algorithms with improved results when compared with other heuristics, such as GAs, among others, see e.g. Bui and Zrncic (2006); Yin and Wang (2006); Bin et al (2009); Faria et al (2006); Putha et al (2012); Monteiro et al (2013). Therefore, we expect ACO to have a competitive performance, in comparison with other heuristic methods already used to solve the HMFST problem.

As far as the authors are aware of, although several MST problems have been solved with ant based algorithms, the special case of the Hop-constrained MST with flow characteristics and with nonlinear costs has not yet been addressed by using ACO algorithms. Therefore, our contribution is twofold. 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, the use of general nonlinear and concave cost functions comprising fixed-charges, which we consider to be more realistic when economies of scale are available. In general, works on HMST use linear costs, for example, Gouveia and Requejo (2001); Gouveia et al (2011) and, apart from Fontes (2010) and Fontes and Gonçalves (2012), this type of functions were never used before with HMST problems. We compare the results obtained with the ones obtained with the commercial software CPLEX and with the ones reported in literature for an MPGA Fontes and Gonçalves (2012). The MPGA is based on a recently proposed framework by Gonçalves and Resende (2011), to solve combinatorial optimization problems with Genetic Algorithms using biased random-keys representation, where the authors also provide a survey of successful applications of this method reported in the literature.

The remainder of this paper is organized as follows. In Section 2, we provide a literature review on the HMFST problem and other related generalizations of the MST problem. In Section 3 we provide a formal description of the HMFST, along with its mathematical formulation and the cost functions herein considered. In Section 4 we review some work on ACO and in Section 5 we develop our approach to solve the HMFST problem. The results obtained and the subsequent analysis are reported and discussed in Section 6. Finally, Section 7 provides a summary and some conclusions of what has been done as well as a discussion of future work.

[1]The TSP can be seen as the one where the shortest circular route between a given set of nodes is to be defined, without visiting twice the same node.

2 Literature review

The MST problem and its extensions have been solved by several techniques. Genetic Algorithms (GAs) were first used to solve MST problems in the mid 90's and they are still one of the most popular heuristic methods to solve them, as well as, to solve many types of optimization problems. Zhou et al (1996) use a GA to solve degree-constrained MSTs. They compare three types of encodings for the chromosome used in the GA they propose, in order to choose the best one: the first associates an index to each arc thus constructing chromosomes with $n-1$ genes; the second, which was the selected one, is based on the Prüfer number encoding allowing the use of $n-2$ genes to encode a tree with n nodes; the third associates a bias value to each node and each arc. The degree constraint is guaranteed by replacing each violating gene with another gene that is not in violation of the degree constraint. The authors apply their GA only to a literature problem with a 9-vertex complete graph. Voß (1999) extends a previous mathematical formulation for the HMST from Gouveia (1995), which is based on the Miller-Tucker-Zemlin subtour elimination constraints originally defined for the TSP. The model is solved by a Tabu Search heuristic that uses a cheapest insertion heuristic, based on principles of the Prim's algorithm Prim (1957), in order to find an initial feasible solution. Hassin and Levin (2003) provide a polynomial algorithm to solve HMST problems for the case of a 2-vertex-connected graph and another polynomial algorithm with bounded performance guarantee for the general case. The Capacitated MST problem is solved in Reimann and Laumanns (2006) by constructing the solution based on an ant algorithm developed to solve the Capacitated Vehicle Routing Problem (CVRP). The algorithm uses a probability function based on the savings information (for adding arcs between pairs of demand nodes) in substitution for the heuristic information. Bui and Zrncic (2006) solve the degree-constrained MST problem with the use of an ACO algorithm, where each ant instead of constructing an entire solution only selects an arc. These arcs are not used directly to construct a tree but only to update the pheromone levels of each arc. Afterwards, candidate arcs are listed, based on their pheromone levels, and then a spanning tree is constructed with this information and with a modified version of Kruskal's algorithm. Only linear costs are considered. A recent work on the MST problem is that of Katagiri et al (2009), where the authors propose an algorithm hybridizing a Tabu Search (TS) and an Ant Colony to solve the K-Minimum Spanning Tree problem. More recently, Neumann and Witt (2010) use a simple ACO algorithm, 1-ANT, to solve the MST problem. They compare two different construction techniques in problems where pseudo-Boolean functions are to be optimized. One technique is a modified version of the Broder construction graph Broder (1989), and uses pheromone information to choose the next arc to be included into the solution. The other technique includes a new arc in the solution tree, one at a time, if it does not include a cycle. A polynomial upper bound is proved for the first technique. The second technique has the lowest running times.

As the HMST problem belongs to the NP-hard class of problems, the development of lower bound schemes is very popular. Gouveia and Martins (1999) introduce an extended and compact formulation for the Capacitated MST problem, by observing that the problem can be reduced to the HMST problem due to the one unit of traffic produced by every demand node that is considered by the authors. This new formulation introduces another index in the variables, the hop index, identifying the position of an arc in the solution. Furthermore, a new set of inequalities, called hop-ordering inequalities, are developed based in a former formulation of Gavish (1983) for a capacitated MST problem, to limit the flow in every arc not connected to the root. Latter on, Gouveia and Martins (2000) improve upon these lower bounds by proposing several levels of aggregation. The new model aggregates into one single variable all the variables associated to a given arc beyond a certain position P. This allows for the creation of a hierarchy of hop-indexed models and a reduction on the number of variables in the models. Lower bounds are computed by three iterative methods, based in the solution of a sequence of Lagrangean relaxations of the hop-indexed models with varying P values: the first one uses reduction tests to eliminate variables from the model; the second one adds each hop-ordering inequality violated by the solution; and the third one includes generalised subtour elimination constraints to the model. A Lagrangean Relaxation approach, based on a network flow formulation, is used in Gouveia and Requejo (2001) to solve the HMST problem. In it, the flow conservation constraints are associated to Lagrangean multipliers and dualized. The problem is further simplified and is separated into two subproblems: one involving the variable identifying the arcs that are in the solution tree and the other involving a variable indicating whether or not an arc is included in the solution in position q of the path from the root node to some node k. Gouveia et al (2007) model the HMST problem as a Steiner tree in a layered directed graph. The identification of each layer is associated to the hop constraint value h, where $h = 1, 2, \ldots, H$. The nodes respecting h will be copied into the corresponding layer h. Lower and upper bounds are computed by using a Dual Ascent Heuristic and a primal heuristic SPH-Prim, respectively, and finally a cutting plane algorithm is applied. The interested reader is referred to the comprehensive survey by Dahl et al (2006), and to the references therein, on how to compute lower bounds for the Hop-constrained MST problem, including techniques such as Lagrangian Relaxation or Column Generation.

Regarding the development of good heuristic methods to solve the HMST problem, not much has been done, apart from the works we review bellow. Fernandes et al (2007), taking advantage of the problems similarity, develop five heuristic procedures to solve the HMST problem based on the ideas previously proposed for the Capacitated Minimum Spanning Tree (CMST). Initially, a Savings Heuristic (SH) is developed mainly to generate initial solutions. The SH starts with a solution with all nodes linked to the source node and then performs swaps, with arcs not present in the solution tree, that represent the best savings. To generate different initial solutions, the savings heuristic is given a set of allowed arcs S_1, as well as a set of prohibited arcs S_2, differing from solution to solution. The five heuristics differ on the definition of S_1 and S_2. The first heuristic defines S_1 as the set of arcs of the previous solution to be excluded from the next solution, provided that they are not linked to the source node. The second heuristic considers prohibiting two arcs at a time. The third heuristic incorporates in S_2 the cheapest arcs incident to each node and the arcs with the minimum cost which are closer to the source node. The fourth heuristic uses as candidate arcs the set made of the four cheapest arcs incident to each node provided that they are not linked to the source node and they are not yet in the solution. Finally, the fifth heuristic is a modified version of the second heuristic where at each iteration a new solution is calculated for each arc to be prohibited. Then, all arcs of that solution are possible candidates for prohibition. The last heuristic, although having the higher time requirements, has the best performance. Gouveia et al (2011) were able to improve upon these results with the use of Dynamic Programming and of heuristics based on the exchange of arcs and on node-level exchanges. The level of a node is defined as the maximum number of arcs a node can have between itself and the source node and is the base used on a restricted Dynamic Programming (DP) formulation of the problem. The state-space restriction rule allows for the movement of at most d nodes, between any consecutive levels, and the state-transition rule forbids parallel shift moves starting and ending at different pairs of overlapping levels. Shift, swap, and shift or swap standard arc-exchange neighbourhoods are defined, as well as, a method combining arc-exchange and swap and shift moves. Five distinct heuristics were constructed by incorporating these neighbourhoods and the method combining arc-exchange, swap and shift moves was found to be the best one.

Works considering HMST problems with different node requirements are scarce. Fontes (2010) uses Dynamic Programming to solve the HMFST problem, i.e. an MST problem with Hop-constraints and with flow requirements other than the unit. The DP formulation is recursive which means that with the use of a backward-forward procedure the state space graph is successively expanded. The algorithm starts from the last stage and works backwards, through not yet computed states, until a computed state is reached. Then, after computing the state value the algorithm moves forward, through already computed states, until it reaches a state not yet computed. The process is repeated until the final state is reached and no better solution can be found. The author considers three distinct nonlinear cost functions with discontinuities, depending on a percentage of the total demand, other than at the origin. In total, Fontes solves 4050 problem instances to optimality being able to demonstrate that the computational performance is independent of cost function type. Fontes and Gonçalves (2012) use a Hybrid Biased Random Key Genetic Algorithm with 3 populations evolving separately to solve the same problems. These populations are randomly generated and are let to evolve independently. Then, every 15 generations the two best chromosomes are included in all other populations. The encoding of the solution tree is made by resorting to random keys. Therefore, a chromosome is made of $3n$ random numbers, where n is the number of nodes in the tree. The first $2n$ genes are used by a Tree Constructor procedure in order to decode the random keys into a solution tree. The last n genes are used by a Local Search procedure in order to improve the solution found. The hop-constraint is handled a posteriori, by penalizing infeasible solutions with more than h arcs in the path from the source node. 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.

3 Problem definition and mathematical formulation

The HMFST problem considers the selection of a tree spanning all nodes in a network in such a way that costs are minimized. In addition, the flows to be routed through each arc have to be found and the maximum number of arcs in each path between the source node and each demand node is limited. As the uncapacitated version of the problem is being considered, there are no upper or lower bounds on the arcs capacity.

Formally, the problem can be defined as follows. Consider a directed network $G = (N, A)$, where N is a set of $n + 1$ nodes, with n demand nodes and one single source node t, and $A(\subseteq N \times N \setminus \{t\})$ is a set of m available arcs (i, j). The number of available arcs is at most $(n + 1) \cdot n$ since there is only one source node. An HMFST is a problem dealing with the minimization of the total costs f_{ij} incurred with the network while satisfying the nodes demand d_j. The total demand of the network, D, is given by the summation of all node demands. The commodity flows from a single 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 mathematical programming model that is given next for the HMFST problem is an adaptation of the hop-indexed formulation for the Constrained MST problem by Gouveia and Martins (1999), which in turn is based on the one developed by Gavish (1983). An important feature has been added by them to the mathematical formulation, which is the introduction of an extra index h identifying the position of an arc in the solution tree counting from the source node t, i.e, the number of hops to the arc. Therefore, h can only take values from 1 to H. Considering the notation summarized bellow, the model can be written as:

t	-	source node,
n	-	number of demand nodes,
d_j	-	demand of demand 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 of arc (i, j),

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

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

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

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

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

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

Please note that, in this model we do not consider variables y_{0jh} and x_{0jh} for all $h \geq 2$ because any arc leaving the root node cannot have a hop value larger than 1; variables y_{ij1} and x_{ij1} (for all i, j) because in this case we are dealing with arcs not directly connected to the root, therefore they cannot be in position (hop) 1; and variables y_{iij} and x_{iij} because we are not considering loops in the tree. The objective in this problem is to minimize total costs incurred with the tree spanning all its nodes, as given in (1). Equations (2) guarantee that every node is in the solution in exactly one position. Equations (3) are called the flow conservation constraints, and they state that the difference between the flow entering a node and the flow leaving a node must be the demand of the node. Furthermore, they also state that if the flow enters a node through an arc in position h, then the flow leaves that same node through an arc in position $h+1$. Equations (4) are called the coupling constraints. Constraints (5) and (6) state the nonnegative and binary nature of the decision variables. It is assumed that the commodity produced by the source node t equals the sum of all the demands d_j, i.e.,

$$\sum_{j \in N \setminus \{t\}} d_j + d_t = 0, \tag{7}$$

where d_t is the demand of node t (represented by a negative value).

3.1 Cost Functions

Given the easiness of approximation of any cost function by the first few terms of a Taylor Series, four types of polynomial cost functions are considered in this work:

- **Type 1:**

$$f_{ij}(x_{ij}) = \begin{cases} b_{ij} \cdot x_{ij} + c_{ij}, & \text{if } x_{ij} > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

- **Type 2:**

$$f_{ij}(x_{ij}) = \begin{cases} 0, & \text{if } x_{ij} = 0, \\ b_{ij} \cdot x_{ij} + c_{ij}, & \text{if } x_{ij} \leq D/2, \\ b_{ij} \cdot x_{ij} + c_{ij} + b_{ij}, & \text{otherwise.} \end{cases} \tag{9}$$

- **Type 3:**

$$f_{ij}(x_{ij}) = \begin{cases} 0, & \text{if } x_{ij} = 0, \\ b_{ij} \cdot x_{ij} + c_{ij}, & \text{if } x_{ij} \leq D/2, \\ b_{ij} \cdot x_{ij} + c_{ij} - b_{ij}, & \text{otherwise.} \end{cases} \tag{10}$$

These first three cost functions F1, F2, and F3 consider linear routing costs b_{ij} per unit of flow routed through arc (i, j), as well as, fixed costs c_{ij}. In addition, there is a discontinuity in F2 cost functions by adding b_{ij}, and in F3 cost functions by subtracting b_{ij}, when the flow passing through an arc is higher than half the total demand D. The fourth cost function is represented by complete second-order polynomials and is initially concave and then convex, the discontinuity point being at half of the total demand.

- **Type 4:**

$$f_{ij}(x_{ij}) = \begin{cases} 0, & \text{if } x_{ij} = 0, \\ -a_{ij} \cdot x_{ij}^2 + b_{ij} \cdot x_{ij} + c_{ij}, & \text{if } x_{ij} \leq D/2, \\ a_{ij} \cdot x_{ij}^2 + b_{ij} \cdot x_{ij} + c_{ij}, & \text{otherwise.} \end{cases} \tag{11}$$

All cost functions consider a_{ij}, b_{ij}, and $c_{ij} \in \mathbb{Z}^+$. Please 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, provided that the position does not violate the hop-constraint.

The use of these cost functions, except for F1, follows the work of (Fontes and Gonçalves, 2012). Cost function F1 was herein introduced with the purpose of assessing the behaviour of the algorithm when small changes are introduced in the form of the cost function, when considered along with F2 and F3, given that the three cost functions are based on first order polynomials.

4 Ant colony optimization

In their daily life, one of the main tasks ants have to perform is to search for food, in the vicinity of their nest. While walking in such a quest, the ants deposit a chemical substance called pheromone in the ground. This is done with two objectives. On the one hand, it allows ants to find their way back to the nest, such as Hansel and Gretel in the fairytale. On the other hand, it allows other ants to know the way they have taken, so that they can follow them. Since hundreds or even thousands of ants have this behaviour, if one could see the pheromone laid in the ground as a kind of light, one could see a large network with some of the arcs brighter than the others. And within the paths created by those arcs would surely be the shortest path between the nest and the food source. What has been observed by Deneubourg et al (1990) is that every time an ant has to choose between paths to follow to reach the food source, the ant will choose with higher probability the path with the largest pheromone concentration. However, they have also observed that there are always some ants that “like” to explore new paths. These are the principles of Ant Colony Optimization. It was the observation of this sort of communication developed by the ants that inspired Dorigo and Stützle to develop the first ant based algorithm which was called *Ant System* Dorigo et al (1996), that was used to solve the Travelling Salesman Problem (TSP), a well known NP-Hard problem.

Ant based algorithms have two main phases, one is the construction of the solution and the other is the pheromone update. Let us consider a network of arcs and nodes. Ants move on the network by going from one node to another. The node where the ant moves to is probabilistically chosen based on the pheromone quantities deposited on the arcs outgoing from the node it stands. After the ants have constructed their respective solutions, the pheromone trails are updated. The update is performed in two steps: in the first step pheromone values are decreased by a constant decay so as to mimic the natural process of evaporation; in the second step, the pheromone on the arcs of the network which are present in the solution are reinforced with extra pheromone. Such reinforcement is usually proportional to the solution quality. The process of solution construction and pheromone updating is repeated until some stopping criterion has been reached.

The description of the Ant Colony Optimization Metaheuristic by Dorigo and Stützle (2004) was followed by several works that studied the introduction of modifications to the AS, such as the introduction of a *Daemon*. The daemon, which has no equivalence in nature, has a very active and important role in the algorithm because it allows for operations that use global knowledge of the ant solutions. In ACO, the daemon can control the feasibility of each solution, for example, by evaporating a percentage of the pheromone quantity in each arc as a way of penalizing such a solution. To avoid premature convergence, usually daemon actions make use of the best solution in the current iteration and/or the best solution found so far, whose arcs are the only ones to be allowed to have pheromone deposited in them. Soon it was also made clear that ant algorithms would benefit from the introduction of a Local Search feature, to cope with a thorough exploitation of the search space nearby good solutions, leading almost always to better performances of the produced hybrid ACOs.

Nevertheless, regardless of their type, ant algorithms became very popular since the beginning and have been used to solve combinatorial problems in several research areas: Image Processing Meshoul and Batouche (2002), Data Mining Parpinelli et al (2002), Protein Folding Hu et al (2008), Power Electronic Circuit Design Zhang et al (2009), Grid Workflow Scheduling Problem Chen and Zhang (2009), Transportation Problem Musa et al (2010); Santos et al (2010), just to mention but a few. With time, several modifications have been introduced and considered as the main components of ant algorithms.

Talbi et al (2001) developed an AS algorithm, that uses a set of parallel ant colonies, to solve the Quadratic Assignment Problem. In it, they use the principles of both the AS, as every ant is allowed to deposit pheromone in every component of its solutions, and the ACO heuristic, as the pheromone quantity to be deposited is dependent of daemon actions, over all best and worst solutions found. Thus, the pheromone update reinforces pheromone values of the parts of every solution S taking into account not only its value $F(S)$ but also the value of the best solution $F(S^*)$ and the value of the worst solution found $F(S^-)$, making it proportional to the ratio $\frac{F(S^-)-F(S)}{F(S^*)}$. Their purpose is to weaken the reinforcement, preventing a quick convergence, due to the unusual large number of ants depositing pheromone on their solutions. A similar approach is used by Alaya et al (2004) to solve multidimensional knapsack problems. In their case, the pheromone update is done in such a way that the quantity deposited in each component of the solution includes information about the difference between the objective function value of the best solution of the iteration $F(S^i)$ and of the global best solution $F(S^*)$, $\frac{1}{1+F(S^*)-F(S^i)}$. Therefore, the closer the solution is to the global best solution, the higher the quantity of pheromone deposited. The influence of the heuristic information in the performance of ant algorithms when solving Set Covering Problems was studied in Lessing et al (2004). Other works use different types of ants in their algorithm. Rappos and Hadjiconstantinou (2004) use ACO to solve two-edge connected network flow design problems introducing two classes of ants, flow ants that are mainly related to the construction of the network, and reliability ants that are concerned with the reliability of the network. Chen and Ting (2008) solve a Single Source Capacitated Facility Location Problem using two colonies, one with location ants to find the location of facilities, and the other with selection ants to assign customers to locations. A Terminal Assignment problem is solved in Bernardino et al (2009) by means of an ACO algorithm with a Local Search procedure embedded in it. The information about the pheromone quantity laid in each path is used to modify the solutions that were obtained previously, instead of producing new solutions. Although ACO algorithms perform well, there are some advantages in hybridizing them with other metaheuristics, thus benefiting from the joint characteristics obtained. While Crawford and Castro (2006) solve both Set Covering and Set Partitioning benchmark problems with

ACO algorithms, and with hybridizations between ACO and Constraint Programming techniques, Forward Checking, Full Lookahead, Arc Consistency, and Post Processing procedures, Bouhafs et al (2006) join Simulated Annealing and an Ant Colony System (ACS) to solve Capacitated Location-Routing problems. The SA component of the algorithm is used to locate the distribution centres (DC) and to assign customers to each DC, while the best routes are defined by the ACS. In Altiparmak and Karaoglan (2007) the authors hybridize ACO and Genetic Algorithms to solve Transportation Problems with square root concave costs. They introduce a mechanism to identify the stagnation of the algorithm in two ways: i) whenever more than 50% of the arcs in the network have reached a minimum value allowed for the pheromones, setting all the pheromones to a maximum value, and ii) whenever the global best solution has not been updated for 50 iterations, in this case replacing 10% of the worst chromosomes of the population with randomly generated ones.

The use of several colonies of ants, whether with different or with similar tasks, has also been approached in the literature. One of the first works was developed by Gambardella et al (1999). They use two different colonies to solve the Vehicle Routing Problem. One such colony is the ACS-VEI which is used to handle the minimization of the number of vehicles (routes). The other, is the ACS-TIME colony that will try to optimise the travelling time of the solutions found by the ACS-VEI. Middendorf et al (2002) present a study on four different techniques to share information between the colonies. The first identifies and sends to all the colonies the overall best solution. In the second strategy each colony will send its best solution to its neighbour colony. The third strategy identifies the best m solutions between a pair of neighbour colonies and these are the ones used to update the pheromone matrix. Finally, the fourth strategy is a combination of the previous two methods.

Many more works could be cited, but it would be beyond the objective of this work. For the interested reader, besides the works already mentioned here and the references therein, Cordon et al (2002), García-Martínez et al (2007), and Mullen et al (2009) provide excellent surveys on ant colony algorithms and their applications.

Next, we will describe the approach we have made with ACO to solve the HMFST problem.

5 Ant colony optimization approach for the nonlinear HMFST problem

The construction of a heuristic algorithm is always associated to a set of major decisions that have to be made regarding parameters values and more complex issues such as the decision of how to construct a solution. ACO algorithms are no different. Therefore, the following list of decisions, considered as their building bricks, are specified: method chosen to construct the solution, heuristic information, pheromone updating rule, probability function, parameter values, and finally, but also very important, the termination condition. In the following sections we describe each one of them.

5.1 Defining a solution to the HMFST problem

The first and most important decision to be made is the representation of the solution to the problem being solved, since a poor representation can lead to not so good solutions, and high computational running times.

The solution for the HMFST problem is a tree, i.e. a graph connecting all nodes without cycles. Therefore, between the central node, also known as root or source node, and every demand node, there can only exist a single directed path. The solution constructed by an ant must then consist on as many arcs as the number of demand nodes and include all demand nodes. Furthermore, a solution is considered feasible if the path from the source node t to every demand node has at most H arcs.

5.2 Constructing a solution

In the method used to construct solutions for this problem, all ants begin their solution construction at the source node. Initially, an ant selects an existing arc linking the source node t and one of the demand nodes $j \in N \setminus \{t\}$. Then, the ant selects another arc, from the set of available arcs linking the source or one of the demand nodes already in the partial solution to another demand node not yet considered, and adds it to the solution tree. For example, consider a fully connected network with four demand nodes $\{a, b, c, d\}$ and a source node t. Let us suppose that the first arc added to the solution is arc (t, a). The next step to be taken is to choose from the remaining demand nodes $\{b, c, d\}$ the one entering the solution tree provided that it is either linked to the source node t or to the demand node a, already in the solution. The possibilities are thus $\{(t, b), (t, c), (t, d), (a, b), (a, c), (a, d)\}$. One of these arcs is chosen not at random but rather based on the pheromone quantity present in it. These two steps are repeatedly performed until there remains no demand node outside the solution tree.

The steps described above, do not guarantee the feasibility of the solution regarding the hop-constraints. To overcome this problem each time an arc (k, j) is added to the solution tree 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. If an ant is able to construct a solution with n arcs then the solution is feasible. Otherwise, the solution is discarded. Since the problem instances do not consider a complete network at some point an ant may have no possibility of constructing a feasible solution. In this case, we could have chosen to fix unfeasible solutions or allow the ant to look for another solution. However, given that the number of discarded solutions was not significant we decided to disregard unfeasible solutions.

As said before, an arc entering the solution is not chosen completely at random. Instead, it is chosen by using a probability function incorporating information about the visibility of arcs and on the pheromone quantity associated to them, as defined bellow:

$$P_{ij} = \frac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{(i,j)\in A} [\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}, \tag{12}$$

where τ_{ij} is the pheromone present in arc (i, j) at the current iteration; η_{ij} is the visibility of arc (i, j) and is usually represented as the inverse of the cost of the arc; $\alpha, \beta > 0$ are parameters weighting the relative importance of the pheromone value and of the visibility information, respectively. The visibility of an arc is also known as *heuristic information* and is only calculated once at the beginning of the algorithm, since it depends only on the problem data. If we make an analogy between cost and distance, the visibility of an arc (i, j) is higher if node j is "close" to node i and thus we can *see* it while standing at node i, and is lower if it is difficult to *see* node j from node i.

5.3 Updating and bounding pheromone values

After all ants have constructed their solutions, the algorithm steps into the pheromone updating phase. In this phase, the best solution of the current iteration S^i is identified and the algorithm updates the pheromones. We only allow the ant that has constructed S^i to reinforce the pheromone quantity in the arcs of its solution. The update of pheromones initially simulates the natural process of evaporation by reducing the pheromone values in every existing arc $(i, j) \in A$. This is represented by the first component of Equation (13), where $\rho \in]0, 1]$ represents the pheromone evaporation rate and τ_{ij} the pheromone quantity in arc (i, j). If the evaporation rate parameter ρ takes a value near to 1, then the pheromone trail will not have a lasting influence throughout the following iterations, whereas a small value will increase the importance of the arcs a lot longer.

$$\tau_{ij} = (1 - \rho) \times \tau_{ij} + \Delta\tau_{ij}. \tag{13}$$

The second component, $\Delta\tau_{ij}$, represents the pheromone quantity to be deposited in arc (i, j), and is given by:

$$\Delta\tau_{ij} = \begin{cases} \frac{Q}{F(S^i)} & \text{if } (i,j) \text{ belongs to solution } S^i, \\ 0 & \text{otherwise,} \end{cases} \tag{14}$$

where Q is a positive proportionality parameter and $F(S^i)$ is the cost of the best solution found at the current iteration.

In the initialization phase of the algorithm, an equal amount of pheromone τ_0 is deposited in every arc of the problem, thus guaranteeing that every arc has the same chance of being chosen, as stated in Equation 15.

$$\tau_{ij} = \tau_0, \ \ \forall (i,j) \in A. \tag{15}$$

At each iteration, after the pheromone update is performed, a check is done to find out if its value is within the interval $[\tau_{min}, \tau_{max}]$, following the work of Stützle and Hoos (1997). The τ_{max} value depends on the cost of the best solution found so far F^* and on the pheromone evaporation rate ρ, and the τ_{min} value depends on the upper bound for the pheromone value τ_{max} and on a parameter p_{best}, the probability of constructing the best solution, as given in Equation (16).

$$[\tau_{min}, \tau_{max}] = \left[\frac{\tau_{max} \cdot (1 - \sqrt[n]{p_{best}})}{(\frac{n}{2} - 1) \cdot \sqrt[n]{p_{best}}}, \frac{1}{\rho \cdot F^*} \right]. \tag{16}$$

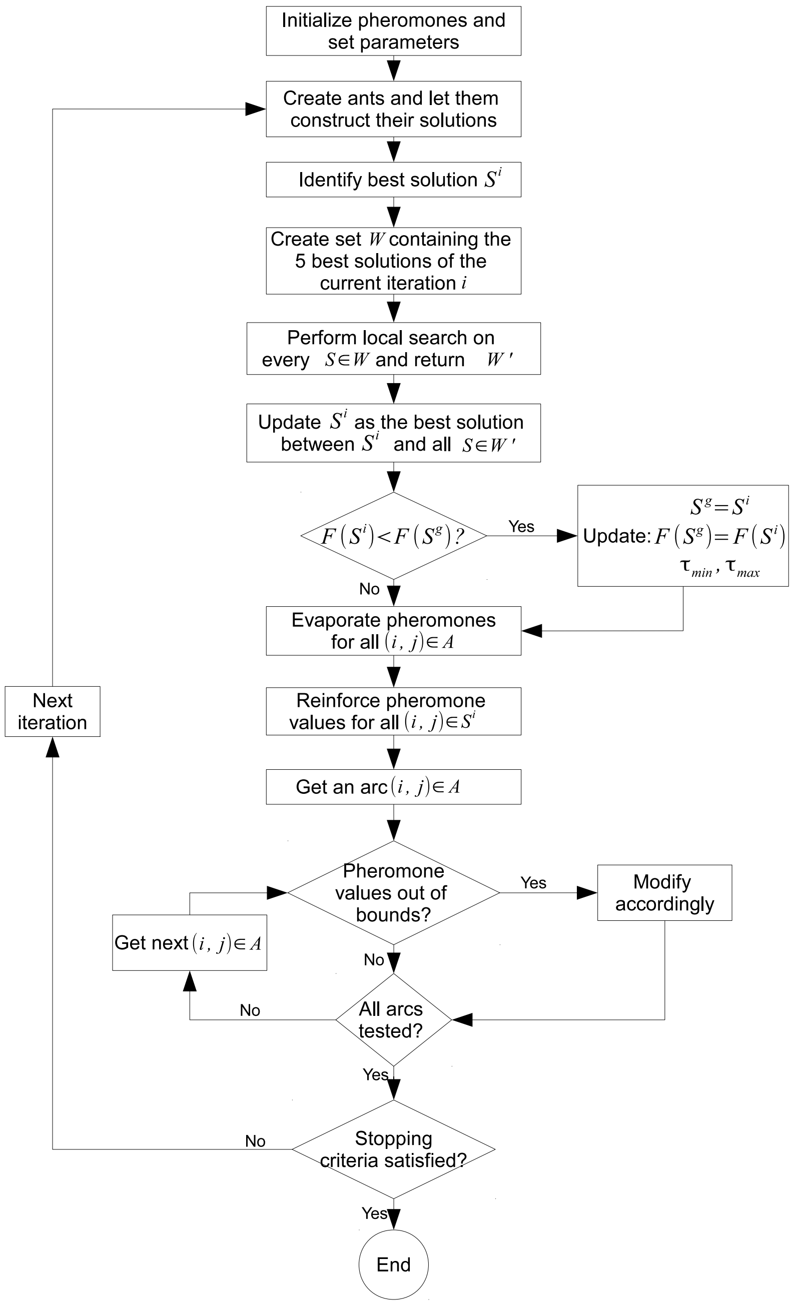
Since both τ_{min} and τ_{max} only depend on the cost of the best solution found so far, they merely have to be updated each time the best solution is improved. When checking if the pheromone values are within the limits defined, the following corrective actions are taken: if on the one hand, some pheromone value is bellow τ_{min}, it is set to τ_{min}; on the other hand, if some pheromone value is above τ_{max}, it is set to τ_{max}. By limiting the pheromone value within these bounds, the choice of the initial pheromone value τ_0 becomes less important as pheromones converge within a few iterations to reasonable values within the desired interval.

Although setting pheromone bounds is a good way to prevent getting trapped into local optima, the algorithm needs an extra mechanism to deal with stagnation and cycling of solutions. To solve this problem we keep track of the number of iterations an incumbent solution has not been changed. Whenever that number reaches 200 iterations, the pheromone values are reinitialized, that is, the pheromone values are set to τ_0 for all arcs. This gives the algorithm a chance to search for a better solution in another region of the search space, before the fixed number of allowed iterations is reached.

5.4 ACO algorithm to solve nonlinear HMFST problems

Now that we have described every specific characteristic of the algorithm let us present the flowchart for the HACO heuristic, which is given in Fig. 1, along with a brief description.

The algorithm starts by depositing an initial pheromone quantity τ_0, on every arc and by initializing all necessary parameters. Then, every ant constructs its solution following the procedure explained in Section 5.2. Solutions are sorted in ascending order of cost and the best solution of the iteration S^i is afterwards identified. A set W is created with the best five solutions found at the current iteration. Local search is performed on all solutions $S \in W$ and a set W' with the improved solutions, if any exist, is returned. S^i is updated by identifying the best solution between the previous S^i and the solutions returned in W'. If the cost of S^i is lower than the cost of the best solution found so far S^g, the incumbent solution S^g and the best cost $F(S^g)$ are updated.



Initialize pheromones and set parameters
Create ants and let them construct their solutions
Identify best solution S^i
Create set W containing the 5 best solutions of the current iteration i
Perform local search on every $S \in W$ and return W'
Update S^i as the best solution between S^i and all $S \in W'$
$F(S^i) < F(S^g)$?
Yes
Update: $S^g = S^i$ $F(S^g) = F(S^i)$ τ_{min}, τ_{max}
No
Evaporate pheromones for all $(i, j) \in A$
Reinforce pheromone values for all $(i, j) \in S^i$
Get an arc $(i, j) \in A$
Pheromone values out of bounds?
Yes
Modify accordingly
No
Get next $(i, j) \in A$
All arcs tested?
No
Yes
Next iteration
Stopping criteria satisfied?
No
Yes
End

Figure 1: Flowchart for the ACO algorithm

The next step is to update the pheromone values. Firstly, pheromones are evaporated in every single arc. Then, a pheromone value proportional to the cost of the best solution found at the current iteration is added to all the arcs in the solution. Afterwards, the algorithm tests all pheromone values for violations of the pheromone bounds. If the pheromone accumulated on an arc exceeds τ_{max} then its value is set to τ_{max}, and if it falls below τ_{min}, then its value is set to τ_{min}.

Continuing with the flowchart, if any of the stopping criteria is satisfied, then the algorithm stops and returns the best solution found, as well as its cost, as the final result; otherwise it continues to the next iteration. Two stopping conditions are used. The algorithm stops running if a predefined maximum number of iterations is reached, or it stops if three consecutive restarts, of the pheromone matrix, have been performed. Recall that the pheromone matrix is reinitialized every time that a number of iterations (200 in our case) has been performed without improving the best solution.

5.5 Local Search

A Local Search procedure was developed and, as said before, is applied right after the ants have constructed a feasible solution and identified S^i. The feasible solutions provided by the ants are sorted in ascending order of total costs, that is to say, sorted from the best to the worst. Then, we create a set W, containing the first five solutions, to have local search performed on. The local search, when performed in the vicinity of a particular solution S, allows the algorithm to search for a neighbour solution that might have a lower cost. 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. Therefore, after the swap takes place node j gets its demand and the demand of all nodes it supplies, if any, from node l instead of from node i. The local search algorithm, that is applied to the aforementioned five solutions, is given in Fig. 2. The algorithm starts by sorting the arcs in the selected solution S in ascending order of their pheromone value, in order to try to substitute in the first place the “worst” arcs, i.e., the ones with the least pheromone. For each one of them we try to find an alternative arc that improves the cost of the solution. In order to do so, we find all the arcs (l, j) that can replace the current one while maintaining feasibility of the solution, i.e. not forming a cycle and maintaining the constraint H for all paths. We attempt to replace the original arc (i, j), by starting with the ones with a higher pheromone value. If one of the replacements improves the cost of the solution we proceed to the next arc (i, j) in the solution without attempting the remaining options. The solution to be used in the remaining of the algorithm is either S or S^N, whichever has the lowest cost.

Let us consider a small example to clarify how the local search procedure is applied. Consider a problem defined on a fully connected graph with six demand nodes $\{a, b, c, d, e, f\}$, a root node t, and where $H = 2$. We make use of Fig. 3 in order to illustrate the evolution of a solution S with the application of local search. Assume that we have already identified the set of five solutions selected to perform local search on and that we are inspecting the neighbourhood of

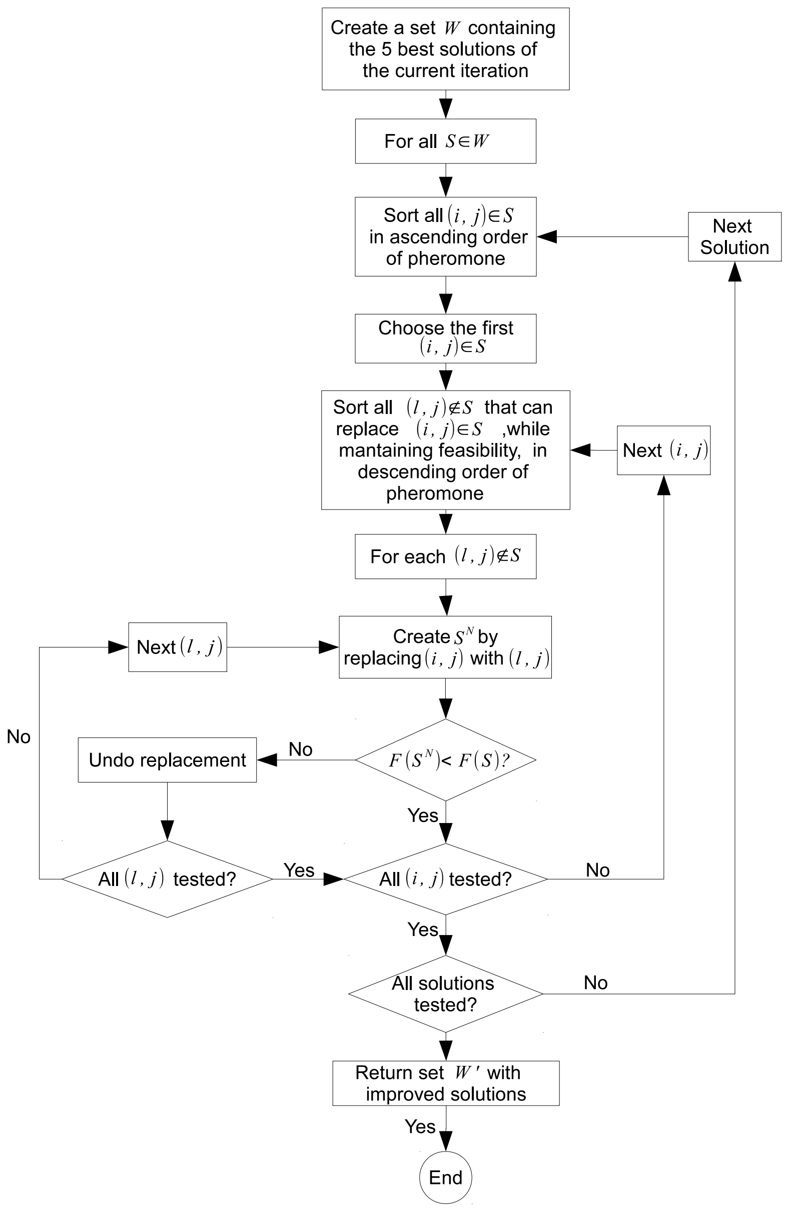

Create a set W containing the 5 best solutions of the current iteration
For all $S \in W$
Sort all $(i,j) \in S$ in ascending order of pheromone
Next Solution
Choose the first $(i,j) \in S$
Sort all $(l,j) \notin S$ that can replace $(i,j) \in S$,while mantaining feasibility, in descending order of pheromone
Next (i,j)
For each $(l,j) \notin S$
Next (l,j)
Create S^N by replacing (i,j) with (l,j)
No
Undo replacement
No
$F(S^N) < F(S)?$
Yes
All (l,j) tested?
Yes
All (i,j) tested?
No
Yes
All solutions tested?
No
Return set W' with improved solutions
Yes
End

Figure 2: Pseudo-code of the Local Search procedure that was incorporated into the ACO algorithm developed

one particular solution S, where $S = \{(f,c),(t,f),(f,e),(a,b),(t,d),(t,a)\}$, see Fig. 3 (a). In order to simplify, assume also that the arcs in S are already sorted in ascending order of arc pheromone, i.e. $\tau_{fc} \leq \tau_{tf} \leq \tau_{fe} \leq \tau_{ab} \leq \tau_{td} \leq \tau_{ta}$.

We try to improve solution S by replacing each of the six arcs in S, one at a time, with better arcs, i.e. with arcs that decrease the total cost of S.

First, we remove arc (f,c) from S, and then we identify the set P of candidate arcs $(l,c) \notin S$ to substitute arc (f,c), since this is the arc with the smallest pheromone value. The candidate arcs set is given by $P = \{(d,c),(t,c),(a,c)\}$. Suppose that P is already sorted in descending order of arc pheromone, i.e. $\tau_{dc} \geq \tau_{tc} \geq \tau_{ac}$. In Fig. 3 (b) all fine dashed lines represent arcs that can be used to reconnect node c to the solution tree S, whereas all dashed lines represent arcs that cannot be considered as candidate arcs because they would lead to unfeasible solutions. Note that arcs (b, c) and (e, c), in dashed lines, cannot be in the set of candidate arcs P because they would violate the hop-constraint $H = 2$.

Following the algorithm, we replace arc (f, c) with arc (d, c) thus obtaining a neighbour solution $S^N = \{(d, c), (t, f), (f, e), (a, b), (t, d), (t, a)\}$. Next, we calculate $F(S^N)$ and compare it with $F(S)$. Let us assume that $F(S^N) < F(S)$. Then, we accept this arc swap and continue with the local search procedure considering this new solution S^N, by making $S \leftarrow S^N$, see Fig. 3 (c). After this swap takes place node c gets its demand from node d instead of from node f.

The local search will now try to replace the next arc in line in S, which is arc (t, f). It is important to notice though that we never go backwards while trying to improve a solution, which means that once a swap has been made, the procedure will not try to improve the swaps that have already been performed. Let us go back to our example.

For arc (t, f) the new $P = \emptyset$, because none of the arcs $(l, f) \notin S$ can provide a feasible solution (they would all introduce either a cycle or violate the hop-constraint), see Fig. 3 (d). Therefore, arc (t, f) is kept in S, and the procedure continues the search with the next arc, arc (f, e), which will render $P = \{(a, e), (d, e), (t, e)\}$, as can be seen in Fig. 3 (e).

The local search procedure continues the search for better solutions until all arcs in the original solution S have been tested. This is repeated for the remaining solutions in W, until all five solutions have been inspected in order to be improved.

6 Computational Experiments

6.1 Test Problems

In order to test the algorithm that was developed we downloaded the Euclidean test set available from Beasley (2010). The set is divided in ten groups $\{g_1, g_2, \ldots, g_{10}\}$ with different ratios between variable and fixed costs, V/F. Each of these subsets has three problem instances. Furthermore, the number of nodes considered is 10, 12, 15, 17, 19, 25, 30, 40, and 50. For the problems with 40 and 50 nodes, there are only 5 groups defined. For further details on these problems please refer to Fontes et al (2003). Therefore, there is a total of 240 problem instances to be solved ten times for each of the four cost functions considered F1, F2, F3, and F4 and each of the four H values, where $H \in \{3, 5, 7, 10\}$.

It is important to report that from the 240 available problems, for $H = 3$ and $H = 5$ and for cost functions F1, F2 and F3 only 165 and 233 problems, respectively, have feasible solutions. Regarding cost function F4, neither HACO nor MPGA were able to find a feasible solution for $H = 3$ for any of the problem instances with size 40 and 50, while for $H = 5$ they were not able

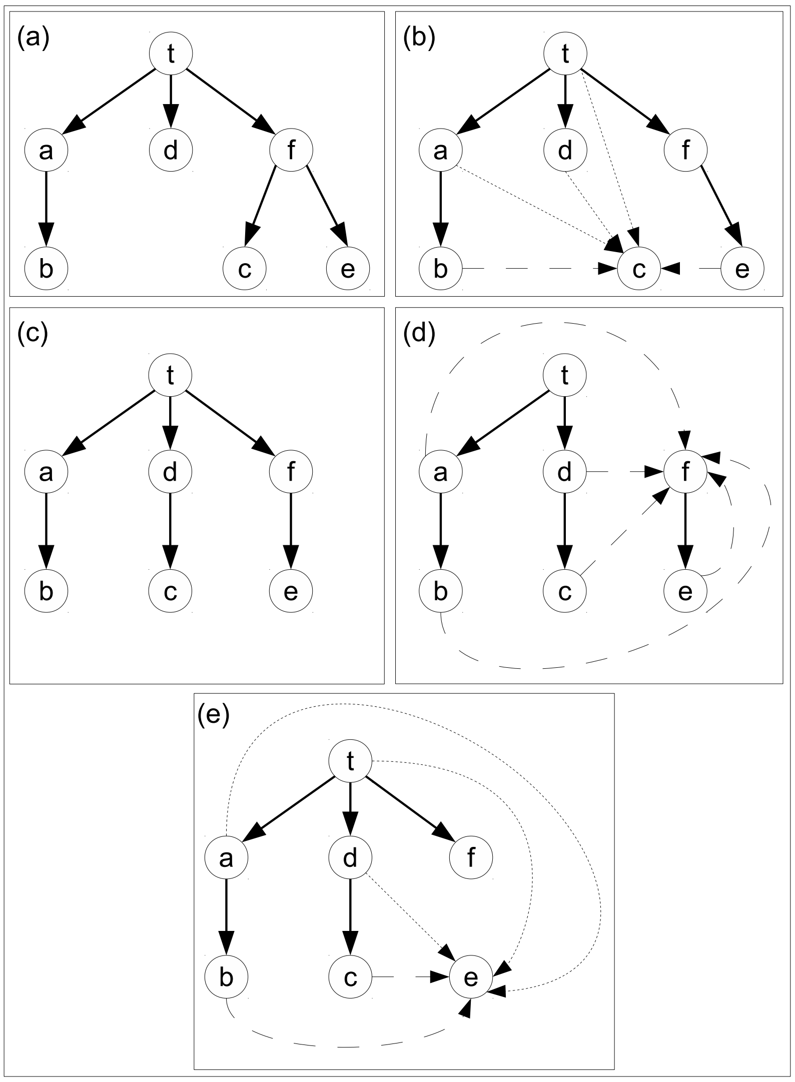


Figure 3: Graphical representation of the example given for the Local Search procedure

to solve two problem instances with size 40 and five problem instances with size 50. However, since these problem sizes have not been solved by an exact method and we cannot guarantee that they have no feasible solution. Thus, all things considered, we have solved a total of 35120 problem instances. In the following sections, we present and discuss the results obtained.

6.2 Parameters setting

There are a few decisions regarding the values to be taken by the parameters described in the previous sections. The development of our algorithm was achieved in several phases and we had to set some values for our first experiences. In an initial phase, based on the literature and previous experience, we tested the parameter values given in the second column of Table 1. The ones with the best results are summarized in the third column. After developing the last phase of the HACO algorithm, we tested the parameter values once more. The results indicated that the ones chosen in the first tests were still the ones achieving the best results. Some comments

Table 1: Parameter values for the ACO.

Parameter	Tested Values	Final Values
α	1, 3, 5	1
β	1, 2, 3, 5	2
ρ	0.05, 0.1, 0.2, 0.5	0.1
Q	1, 2, 5	2
p_{best}	0.5, 0.05, 0.01	0.5
τ_0	1, 1000000	1000000
η_{ij}	$\frac{1}{c_{ij}}, \frac{1}{b_{ij}}, \frac{1}{c_{ij}+b_{ij}}$	$\frac{1}{c_{ij}+b_{ij}}$
no. of ants	n, 2n	2n
no. of iterations	500, 1000, 2000	2000

must be made regarding the choice of the parameters.

The observation of Equation (12) allows for the conclusion that α and β are related and that the choice of their values must be made carefully. Therefore, the tests we have made in order to choose them, consider all combinations between the values of these parameters, and the best combination was the one used thereafter. As for the evaporation rate, parameter ρ, it was found that 10% was the best value, which points towards privileging the exploitation of the search space nearby the best solution of each iteration.

6.3 Comparing our results with the ones in literature

In this section, we present the computational results that were obtained with the HACO algorithm that was developed along with results obtained with the commercial software CPLEX 12.0 and also some literature results for the same problems, in order to compare the efficiency end effectiveness of our algorithm. The analysis of the robustness, the ability of the heuristic to reproduce the same solution or a very similar one in different runs, is approximately achieved by solving 10 times each problem instance and then by computing the minimum, maximum, average and standard-deviation of the solutions obtained. If the first three statistics are around the same values and the standard deviation is small, then the method may be considered robust.

The algorithm described in this paper was implemented in Java and the computational experiments were carried out on a PC with a Pentium D at 3.20GHz and 1GB of RAM. The CPLEX was run on the same PC.

To evaluate an heuristic, we characterise its solutions regarding:

1. Time, in seconds, required to perform a full run of the algorithm;

2. Optimality gap in %, which is given by:

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

where OptS stands for the optimum solution, obtained by CPLEX for cost functions F1, F2 and F3 when available, and the best known otherwise; and the HS stands for the best solution found with the heuristic in question.

The HMFST problem was solved with CPLEX for F1, F2, and F3 cost functions. CPLEX does not solve problems with functions of type F4. Nonetheless, Fontes (2010) provided times and optimal solutions obtained with a Dynamic Programming (DP) algorithm for cost functions of type F4 and problems with up to 19 nodes. For larger problems we have used the results obtained by Fontes and Gonçalves (2012), available in www.fep.up.pt/docentes/fontes (under project PTDC/EGE-GES/099741/2008), by calculating the gap between the costs obtained by our HACO and the very best ones obtained with the Multi-Population hybrid biased random key Genetic Algorithm (MPGA), since no optimal results are available to the moment.

In order to infer about the stability of our method, we present a set of statistics regarding the gap. Tables 2 to 5 summarize the gap results obtained for each cost function herein considered. Note that Table 5 refers to problems with cost function F4 only with up to 19 nodes. Each table presents Minimum (Min), Average (Avg), 3rd Quarter (3Q), Maximum (Max), and Standard Deviation (SD) values obtained with the HACO algorithm, grouped by hop value, as well as Minimum (Min), Average (Avg), and Maximum (Max) gap results obtained with the aforementioned MPGA.

Before going any further please note that as cost function F1 was not considered in Fontes and Gonçalves (2012), Table 2 does not include results for the MPGA algorithm. Furthermore, by setting the hop value to 3 problems with 40 and with 50 nodes do not have any feasible solution. Recall that the problem instances do not consider a complete network.

Let us begin by analysing the gap. First of all, we observe that the minimum gap value obtained, regardless of cost function and hop value considered, is always zero. This result is very important because it means that in the 10 runs of the algorithm we are always able to find, at least once, the optimum solution. Furthermore, the value for the third quarter of the gap distribution is also zero, meaning that at least 75% of the solutions found by the HACO are optimal.

When we analyse the results from the Hop value point of view, it is curious to notice that, although it is known that the difficulty in solving these problems increases with the decrease on the hop parameter value H, the HACO heuristic has a very good performance for $H = 3$, regardless of the cost function, and finds the optimum value for all runs of the algorithm. The same can not be said about the performance of the MPGA which presents the worst gap values precisely for $H = 3$, in particular, the largest gap value, 17%, was obtained for a problem with 19 nodes and considering cost function F2. The HACO and MPGA performance achieved for

Table 2: HACO optimality gap (%) for F1 cost function with $H = 3, 5, 7$, and 10

	H=3					H=5				
N	Min	Avg	3Q	Max	Sd	Min	Avg	3Q	Max	Sd
10	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0.002	0	0.08	0.010
40						0	0	0	0	0
50						0	0.010	0	0.49	0.069

	H=7					H=10				
N	Min	Avg	3Q	Max	Sd	Min	Avg	3Q	Max	Sd
10	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0
40	0	0.001	0	0.02	0.003	0	0	0	0	0
50	0	0.002	0	0.05	0.010	0	0	0	0	0

$H = 10$ is also of zero gap for cost functions F2, F3 and F4. Problems with $H = 5$ and $H = 7$ proved to be the most difficult ones to solve by HACO, and the largest gap value obtained is 0.51%, considering F2 and $H = 5$. Gaps larger than zero tend to happen with problems with at least 30 nodes, although one 17 node problem presented a nonzero gap.

Looking now at the results obtained by type of cost function, there seems to be no influence on the performance of our algorithm as the gap values are not that different. In general, whenever MPGA presents a positive average gap, HACO has improved it except for the particular case, already mentioned, of the problem with 17 nodes.

In Table 6, we report on the optimality gap results for the HACO and for the MPGA for problems with 25 up to 50 nodes and cost function F4. These problems have only been solved with these two heuristics therefore, 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). As it can be seen in Table 6, the tendency for finding the optimum value that has been observed for the former three functions is confirmed for HACO when $H = 3$, and for

Table 3: HACO and MPGA optimality gap (%) for F2 cost function with $H = 3, 5, 7$, and 10

	H=3								H=5							
	HACO					MPGA			HACO					MPGA		
N	Min	Avg	3Q	Max	Sd	Min	Avg	Max	Min	Avg	3Q	Max	Sd	Min	Avg	Max
10	0	0	0	0	0	0	0.360	7.157	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0.104	1.900	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0.031	0.641	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0.120	3.743	0	0.008	0	0.46	0.059	0	0	0
19	0	0	0	0	0	0	0.182	17.353	0	0	0	0	0	0	0.062	4.919
25	0	0	0	0	0	0	0.283	6.141	0	0	0	0	0	0	0.005	0.162
30	0	0	0	0	0	0	0	0	0	0.002	0	0.08	0.013	0	0.109	4.083
40									0	0	0	0	0	0	0.139	7.501
50									0	0.005	0	0.51	0.051	0	0.076	0.991

	H=7								H=10							
	HACO					MPGA			HACO					MPGA		
N	Min	Avg	3Q	Max	Sd	Min	Avg	Max	Min	Avg	3Q	Max	Sd	Min	Avg	Max
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0.005	0.258	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	0	0.0004	0	0.02	0.003	0	0.074	1.945	0	0	0	0	0	0	0	0
50	0	0.001	0	0.05	0.007	0	0.047	1.426	0	0	0	0	0	0	0	0

HACO and MPGA when $H = 10$. Nonetheless, there are two important advantages regarding the HACO algorithm. Firstly, HACO is always able to find a feasible solution whereas the MPGA is not. The MPGA was not able to find a feasible solution for 19 problem instances, see the results reported in Fontes and Gonçalves (2012). Secondly, the stability of HACO is confirmed by observing the maximum gap values obtained. Although each problem instance was solved 10 times by the HACO algorithm and only 5 times by the MPGA, the maximum gap values observed for HACO are much lower than the ones observed for MPGA, except for the aforementioned problem with 17 nodes. Nevertheless, even in such case the maximum gap is below 0.5%.

Running time results are presented in Table 7, for CPLEX and HACO and in figures 4, 5, and 6 for MPGA and HACO.

The time spent by CPLEX to solve the HFMST problem clearly increases both with size and with hop value. This behaviour was expected since the number of feasible solutions increases

Table 4: HACO and MPGA optimality gap (%) for F3 cost functions with $H = 3, 5, 7$, and 10

	H=3								H=5							
	HACO					MPGA			HACO					MPGA		
N	Min	Avg	3Q	Max	Sd	Min	Avg	Max	Min	Avg	3Q	Max	Sd	Min	Avg	Max
10	0	0	0	0	0	0	0.389	7.162	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0.110	1.914	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0.075	2.396	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0.120	3.743	0	0.005	0	0.46	0.046	0	0	0
19	0	0	0	0	0	0	0.041	1.787	0	0	0	0	0	0	0.095	4.922
25	0	0	0	0	0	0	0.251	6.993	0	0	0	0	0	0	0.006	0.177
30	0	0	0	0	0	0	0	0	0	0.002	0	0.08	0.010	0	0.071	4.083
40									0	0	0	0	0	0	0.425	13.909
50									0	0	0	0	0	0	0.084	0.992

	H=7								H=10							
	HACO					MPGA			HACO					MPGA		
N	Min	Avg	3Q	Max	Sd	Min	Avg	Max	Min	Avg	3Q	Max	Sd	Min	Avg	Max
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0.006	0.228	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	0	0.001	0	0.02	0.004	0	0.048	1.945	0	0	0	0	0	0	0	0
50	0	0.004	0	0.05	0.011	0	0.004	0.243	0	0	0	0	0	0	0	0

rapidly with problem size due to the combinatorial nature of the problem.

As we have already said, one of our objectives was to infer on the behaviour of the HACO algorithm as small changes were introduced in cost function F1, which in this work is represented by F2 and F3. It is curious to notice that even though CPLEX is much influenced with the form of the cost function, with average running time increasing, in general, from F1 to F2 and then to F3, the HACO performance is almost the same for the three functions. Furthermore, the HACO heuristic can be up to 7 times faster than CPLEX. Not even F4 cost function seems to influence HACO average running times.

This means that one way or the other, with more realistic industrial problems, the HACO heuristic will be able to give a very good answer within a reasonable amount of time, whereas exact methods will not. Furthermore, our algorithm does not seem to be influenced by the type of cost function taken into account (whereas CPLEX seems to find F2 and F3 more challenging), which allows for concluding it to be very robust.

Table 5: HACO and MPGA optimality gap (%) for F4 cost functions with $H = 3, 5, 7$, and 10

	H=3								H=5							
	HACO					MPGA			HACO					MPGA		
N	Min	Avg	3Q	Max	Sd	Min	Avg	Max	Min	Avg	3Q	Max	Sd	Min	Avg	Max
10	0	0	0	0	0	0	0.251	7.160	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0.157	2.453	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0.038	2.352	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0.093	3.858	0	0.01	0	0.48	0.067	0	0	0
19	0	0	0	0	0	0	0.122	9.108	0	0	0	0	0	0	0.042	1.615

	H=7								H=10							
	HACO					MPGA			HACO					MPGA		
N	Min	Avg	3Q	Max	Sd	Min	Avg	Max	Min	Avg	3Q	Max	Sd	Min	Avg	Max
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0.003	0.131	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

In addition, we present graphical representations of the average running times for both the HACO and the MPGA, by cost function, in figures 4, 5 and 6. The results show no big difference between the MPGA and the HACO time performance. However, it should be noticed though that the MPGA was run on a much faster PC, an Intel Core2 processor at 2.4 GHZ.

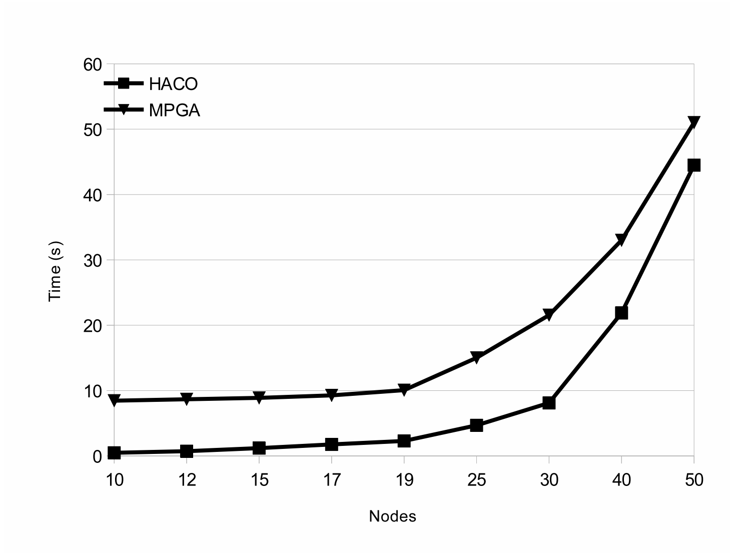


Figure 4: Computational time results obtained with HACO and MPGA for F2 cost functions

In order to infer about the evolution of CPLEX and HACO computational times, as well as about the HACO gap performance, we have also generated larger size problem instances with 60 and 80 nodes. For these sizes, we only consider five different groups, i.e. five different ratios

Table 6: Optimality gap (%) obtained by HACO and MPGA for F4 cost functions with $H = 3, 5, 7$, and 10 when compared with the currently best known solutions (obtained either by HACO or by MPGA)

	H=3								H=5							
	HACO					MPGA			HACO					MPGA		
N	Min	Avg	3Q	Max	Sd	Min	Avg	Max	Min	Avg	3Q	Max	Sd	Min	Avg	Max
25	0	0	0	0	0	0	0.287	6.961	0	0	0	0	0	0	0.004	0.148
30	0	0	0	0	0	0	0.113	2.664	0	0.012	0	0.329	0.055	0	0.152	4.087
40									0	0	0	0	0	0	0.118	7.646
50									0	0.023	0	0.579	0.115	0	0.077	0.579

	H=3								H=5							
	HACO					MPGA			HACO					MPGA		
N	Min	Avg	3Q	Max	Sd	Min	Avg	Max	Min	Avg	3Q	Max	Sd	Min	Avg	Max
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	0	0.001	0	0.078	0.009	0	0.084	1.984	0	0	0	0	0	0	0	0
50	0	0.008	0	0.221	0.038	0	0.048	1.225	0	0	0	0	0	0	0	0

between the variable and the fixed cost components. We have also considered a larger number of arcs in the networks that we have generated, since otherwise there would not be any feasible solutions for small hop values. The larger number of arcs in the networks for problems with 60 and with 80 nodes is the reason why they have feasible solutions even in cases where $H = 3$, as opposed to what happened with the set of problem instances that we have downloaded. Since these larger problems have not been solved with the MPGA, we only report on the results obtained by CPLEX and by HACO.

Although HACO was able to solve all problem instances with 60 and with 80 nodes, it is important to refer that CPLEX was not, due to memory failure. Since the behaviour of CPLEX was not uniform, regarding the number of problem instances solved without memory problems, we report in Table 8 the number of problem instances solved by CPLEX. Please recall that considering five groups and three problem instances in each of them, we have 15 problem instances to be solved per size and hop value, each of which is to be solved 10 times. For both F2 and F3 cost functions CPLEX was not able to solve all problem instances, having once more problems with shortage of memory.

In Table 9, we have the average optimality gap obtained by cost function and hop value. As it can be seen, HACO maintains its lower average gaps and we were always able to find an optimum solution within the 10 runs each problem instance was solved, for the problems CPLEX was able to solve.

Regarding computational running times, see Table 10, besides the already observed increasing behaviour with problem size, for both CPLEX and HACO, there is now stronger evidence of the

Table 7: Computational time results, for CPLEX and HACO, obtained for cost functions F1, F2, F3 and F4 and $H = 3, 5, 7,$ and 10

		Time (s)							
		CPLEX				HACO			
Function	N	3	5	7	10	3	5	7	10
F1	10	0.7	1.3	1.4	2.2	0.3	0.5	0.5	0.5
	12	0.9	1.6	2.2	3.4	0.4	0.7	0.8	0.7
	15	1.6	2.6	3.9	5.7	0.7	1.2	1.4	1.2
	17	2.0	3.7	5.5	10.1	1.0	1.7	2.0	1.9
	19	2.8	4.7	8.6	11.1	1.4	2.1	2.6	3.0
	25	5.1	9.5	18.9	26.0	2.7	4.3	5.4	5.9
	30	7.8	16.2	30.3	54.5	4.7	8.4	8.9	10.1
	40		37.2	71.2	117.8		15.1	27.7	24.6
	50		98.6	138.5	252.2		35.6	68.8	44.8
F2	10	1.2	1.9	3.4	4.3	0.3	0.5	0.5	0.5
	12	1.9	2.9	5.6	6.8	0.5	0.7	0.9	0.8
	15	2.8	5.2	8.8	11.1	0.7	1.2	1.5	1.3
	17	3.8	7.8	11.8	16.9	1.0	2.0	2.1	1.9
	19	5.3	8.3	13.9	21.8	1.3	2.2	2.7	3.0
	25	9.7	17.6	28.9	44.1	2.8	4.7	5.4	5.8
	30	14.3	29.1	47.2	70.2	4.6	9.1	8.9	9.8
	40		65.2	113.1	172.4		15.0	26.3	24.3
	50		161.0	225.9	288.5		35.5	50.7	47.3
F3	10	2.2	2.0	2.9	3.3	0.3	0.5	0.6	0.5
	12	2.6	2.6	4.7	5.5	0.5	0.7	0.9	0.9
	15	3.4	4.3	7.2	8.5	0.8	1.2	1.5	1.5
	17	4.1	6.4	10.1	21.7	1.2	1.9	2.1	2.3
	19	5.6	7.4	15.7	24.1	1.4	2.2	2.7	2.9
	25	11.0	13.5	33.3	52.2	2.9	4.4	5.3	5.7
	30	16.5	29.4	55.7	70.9	4.6	8.7	8.6	9.5
	40		60.5	139.4	239.3		15.0	25.3	23.4
	50		155.5	275.3	367.4		32.4	68.0	48.0
F4	10					0.3	0.7	0.9	0.9
	12					0.8	1.3	1.6	1.8
	15					1.3	1.5	2.5	2.5
	17					1.0	1.8	2.2	2.3
	19					2.9	3.7	3.7	4.2
	25					2.6	4.1	4.9	4.8
	30					3.9	6.7	8.6	8.4
	40						14.2	19.6	20.3
	50						27.8	48.9	47.1

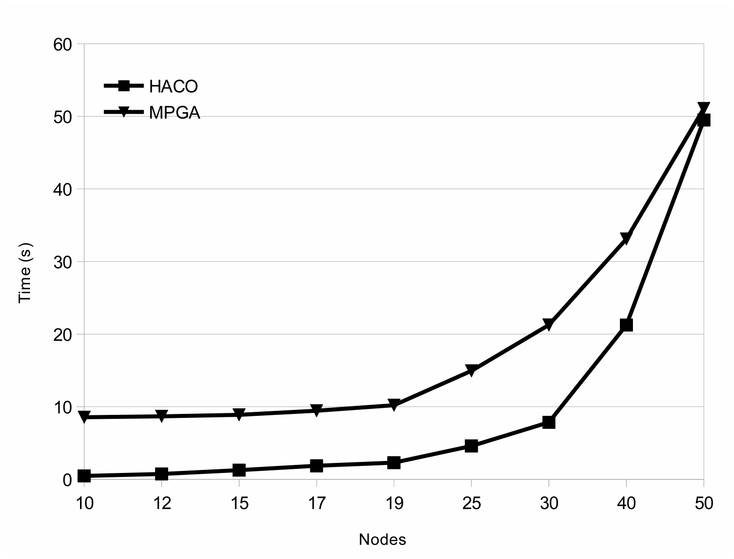


Figure 5: Computational time results obtained with HACO and MPGA for F3 cost functions

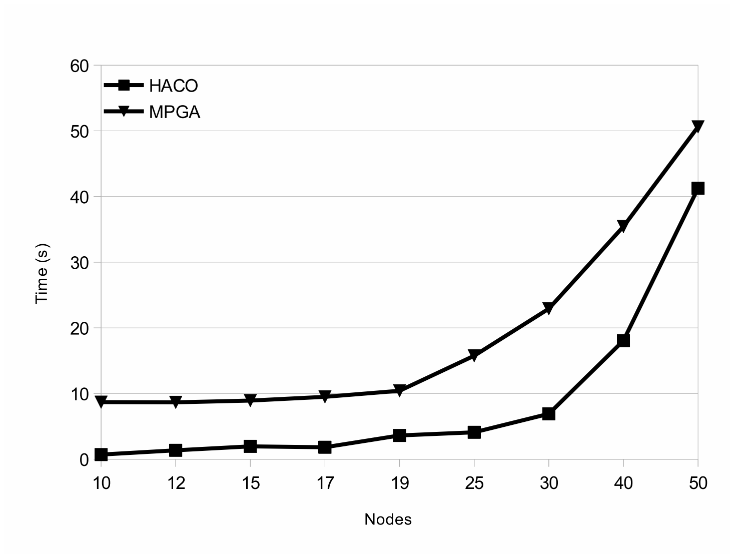


Figure 6: Computational time results obtained with HACO and MPGA for F4 cost functions

influence of the hop values. It is notorious, by solving larger size problems, that running times increase with the hop parameter value, although for CPLEX the rate of influence is much larger. For instance, considering problems with 80 nodes and cost function F1, there is an increase of more than 75% of CPLEX running times from $H = 7$ to $H = 10$. Finally, these new problem instances confirm the conclusion already stated that CPLEX is influenced by the type of cost function whereas HACO is not.

In order to better understand the results obtained with CPLEX, regarding running times, we have performed two extra experiences considering F2 and F3 cost functions. In the first experience we allowed 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 gave CPLEX a time bound above the maximum average running times obtained with HACO. The

Table 8: Number of problem instances with 60 and with 80 nodes solved by CPLEX

		No. of problems			
		HOP			
Function	N	3	5	7	10
F1	60	3	12	11	11
	80	3	9	9	9
F2	60	3	10	10	–
	80	3	8	–	–
F3	60	3	12	9	9
	80	6	9	9	–

Table 9: Average optimality gap (%) obtained with HACO for F1, F2, and F3 cost functions for $H = 3, 5, 7$, and 10 for problems with 60 and with 80 nodes

		GAP			
		HOP			
Function	N	3	5	7	10
F1	60	0	0	0	0
	80	0.001	0	0	0
F2	60	0	0	0	–
	80	0.001	0	–	–
F3	60	0.001	0	0	0
	80	0.001	0	0	–

Table 10: Computational time results, for CPLEX and HACO, obtained for problems with 60 and with 80 nodes considering cost functions F1, F2, and F3 and $H = 3, 5, 7$, and 10

		Time (s)							
		CPLEX				HACO			
Function	N	3	5	7	10	3	5	7	10
F1	60	65,9	246,8	297,6	517,5	16,6	31,5	36,7	40,8
	80	235,8	483,1	845,5	1511,8	56,1	89,7	105,8	116,7
F2	60	67,8	289,6	368,3	–	16,2	29,5	35,4	42,8
	80	197,7	541,1	–	–	52,1	88,5	109,1	117,0
F3	60	103,2	635,6	336,8	596,7	17,2	29,7	35,8	49,1
	80	607,3	549,4	943,3	–	53,4	87,2	131,7	122,0

results obtained can be seen in Table 11, where columns Avg and Max present the average and maximum gaps observed, respectively, and CT presents the percentage of problem instances with a feasible solution solved by CPLEX within the time limit. In the second experience we gave an optimality gap tolerance of 1% to CPLEX and calculated average gap and running times needed. The results obtained can be seen in Table 12, where columns Avg and Max present the average and maximum gaps observed, respectively, and T (in %) gives the proportion of the computational time used to reach a 1% optimality gap, when compared with the time needed to find the optimal solution.

From the results in Table 11, we can observe, from column CT, CPLEX is unable to provide a feasible solution for a considerable number of problem instances. Furthermore, the optimality gap for CPLEX is larger than the one observed for HACO. Regarding the results in Table 11, we can conclude that in order to be within 1% of the optimality gap CPLEX takes about half the time needed to prove that the solution is an optimum. However, HACO continues to have a better performance regarding time and optimality gap.

Table 11: Results obtained by bounding CPLEX in time, and considering F2 and F3 cost functions

	H=3			H=5			H=7			H=10		
N	Avg	Max	CT(%)	Avg	Max	CT(%)	Avg	Max	CT(%)	Avg	Max	CT(%)
F2												
10	0	0	100	0	0	100	0	0	100	0	0	100
12	0	0	100	0	0	100	0	0	100	0	0	100
15	0.0001	0.003	100	0	0	100	0	0	100	0.096	2.872	100
17	0	0	100	0	0	100	0	0	100	0.851	5.609	100
19	0	0	100	0	0	100	0.718	8.124	97	0.457	3.333	67
25	0	0	100	1.012	6.713	97	x	x	0	x	x	0
30	0	0	100	x	x	0	x	x	0	x	x	0
40	-	-	-	0	0	100	0.009	0.129	100	0.115	1.718	100
50	-	-	-	1.918	16.198	90	0	0	53	0.340	2.063	53
60	0	0	100	0	0	9	x	x	0	x	x	x
80	0	0	100	0.002	0.016	78	x	x	x	x	x	x
F3												
10	0	0	100	0	0	100	0	0	100	0	0	100
12	0	0	100	0	0	100	0	0	100	0	0	100
15	0	0	100	0	0	100	0	0	100	0.129	3.867	100
17	0	0	100	0	0	100	0.005	0.160	100	0.994	6.213	100
19	0	0	100	0.097	2.799	97	0.534	5.426	100	0.667	5.657	50
25	0	0	100	1.029	15.269	90	x	x	0	x	x	0
30	0	0	100	x	x	0	x	x	0	x	x	0
40	-	-	-	0	0	100	0	0	100	0.034	0.509	100
50	-	-	-	0	0	100	0.049	0.733	100	1.125	5.667	93
60	0	0	100	1.389	10.963	67	x	x	0	3.560	6.664	22
80	0	0	100	0	0	100	0.237	0.945	44	x	x	x

Table 12: Results obtained by CPLEX with an 1% gap tolerance, and considering F2 and F3 cost functions

	H=3			H=5			H=7			H=10		
N	Avg	Max	T(%)	Avg	Max	T(%)	Avg	Max	T(%)	Avg	Max	T(%)
F2												
10	0.036	0.637	66	0.012	0.189	61	0.005	0.159	43	0.001	0.033	46
12	0.001	0.039	53	0.001	0.039	54	0.032	0.764	41	0.037	0.602	47
15	0.045	0.861	54	0.051	0.989	48	0.057	0.969	43	0.023	0.313	43
17	0.045	0.535	56	0.039	0.535	44	0.034	0.480	44	0.084	0.962	40
19	0.003	0.079	49	0.037	0.588	55	0.096	0.749	48	0.025	0.339	38
25	0.070	0.438	49	0.090	0.726	46	0.047	0.377	46	0.076	0.857	38
30	0.009	0.110	51	0.085	0.625	44	0.086	0.656	37	0.129	0.656	41
40	-	-	-	0.023	0.300	72	0.195	0.927	45	0.100	0.686	38
50	-	-	-	0.029	0.147	45	0.016	0.234	39	0.021	0.234	37
60	0.012	0.023	63	0.074	0.375	51	0.030	0.107	36	x	x	x
80	0.330	0.916	53	0.232	0.991	34	x	x	x	x	x	x
F3												
10	0	0	78	0.011	0.184	49	0.000	0.000	49	0.006	0.184	58
12	0.009	0.252	45	0.053	0.794	62	0.036	0.809	47	0.063	0.679	60
15	0.008	0.115	45	0.003	0.091	54	0.000	0.000	49	0.034	0.789	61
17	0.000	0.000	40	0.000	0.000	52	0.028	0.293	55	0.026	0.561	39
19	0.000	0.000	36	0.012	0.232	55	0.012	0.201	39	0.033	0.403	36
25	0.049	0.644	40	0.054	0.428	57	0.041	0.720	35	0.070	0.720	32
30	0	0	27	0.050	0.673	40	0.038	0.822	29	0.018	0.258	35
40	-	-	-	0.034	0.382	50	0.124	0.442	35	0.101	0.525	29
50	-	-	-	0.007	0.066	39	0.047	0.658	27	0.008	0.058	28
60	0.005	0.014	29	0.054	0.337	29	0.030	0.078	32	0.089	0.493	16
80	0.082	0.360	21	0.072	0.250	27	0.057	0.250	18	x	x	x

7 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 four types, and they comprise both fixed-charge and routing costs, which are very difficult to solve even for problems with a small number of nodes. We have solved problems with four different values of the hop-parameter and with a number of nodes ranging from 10 to 50. We compared our results with the ones reported in the literature and our algorithm proved to be both very effective and very efficient. The solutions obtained were always better or as good as the ones provided by current literature, except for 13 problem instances out of the 2798 solved. It should be noticed that for the remaining 82 problem instances there are no feasible solutions. Furthermore, our algorithm was always able to find a feasible solution, when there was one, whereas the MPGA was 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. Although several cost functions with different complexity have been used, the algorithm performance has not been affected. Both solution quality and computational time remain of the same magnitude. Furthermore, the results obtained over 10 runs of the proposed algorithm are within 0.01% of each other proving the method to be very robust. 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 Fontes and Gonçalves (2012) 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.

References

Alaya I, Solnon C, Ghédira K (2004) Ant algorithm for the multi-dimensional knapsack problem. In: International Conference on Bioinspired Optimization Methods and their Applications, BIOMA 2004, pp 63–72

Altiparmak F, Karaoglan I (2007) A genetic ant colony optimization approach for concave cost transportation problems. In: Evolutionary Computation, 2007. CEC 2007. IEEE Congress on, pp 1685–1692

Beasley J (2010) Or-library. http://www.brunel.ac.uk/deps/ma/research/jeb/orlib/netflowccinfo.html

Bernardino EM, Bernardino AM, Sánchez-Pérez JM, Gómez-Pulido JA, Vega-Rodríguez MA

(2009) A hybrid ant colony optimization algorithm for solving the terminal assignment problem. In: IJCCI 2009 - International Joint Conference on Computational Intelligence

Bin Y, Zhong-Zhen Y, Baozhen Y (2009) An improved ant colony optimization for vehicle routing problem. Eur J Oper Res 196:171–176

Bouhafs L, Hajjam A, Koukam A (2006) A combination of simulated annealing and ant colony system for the capacitated location-routing problem. In: KES (1), pp 409–416

Broder AZ (1989) Generating random spanning trees. In: 30th Annual Symposium on Foundations of Computer Science, 30 October-1 November 1989, Research Triangle Park, North Carolina, USA, IEEE, pp 442–447

Bui TN, Zrncic CM (2006) An ant-based algorithm for finding degree-constrained minimum spanning tree. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation, ACM, New York, USA, GECCO ’06, pp 11–18

Chen CH, Ting CJ (2008) Combining lagrangian heuristic and ant colony system to solve the single source capacitated facility location problem. Transp Res Part E: Log 44(6):1099 – 1122

Chen WN, Zhang J (2009) Ant colony optimization approach to grid workflow scheduling problem with various QoS requirement. IEEE T Sys Man Cybern C 31:29–43

Cordon O, Herrera F, Stützle T (2002) A review on the ant colony optimization metaheuristic: Basis, models and new trends. Mathw Soft Comput 9:141–175

Crawford B, Castro C (2006) Integrating lookahead and post processing procedures with aco for solving set partitioning and covering problems. In: ICAISC, pp 1082–1090

Dahl G, Gouveia L, Requejo C (2006) On formulations and methods for the hop-constrained minimum spanning tree problem. In: Resende MGC, Pardalos PM (eds) Handbook of Optimization in Telecommunications, Springer US, pp 493–515

Deneubourg JL, Aron S, Goss S, Pasteels JM (1990) The self-organizing exploratory pattern of the argentine ant. J Insect Behav 3:159–168

Dorigo M, Blum C (2005) Ant colony optimization theory: A survey. Theor Comput Sci 344:243–278

Dorigo M, Stützle T (2004) Ant Colony Optimization. MIT Press, Cambridge, MA

Dorigo M, Maniezzo V, Colorni A (1996) The ant system: Optimization by a colony of cooperating agents. IEEE T Sys Man Cybern B 26(1):29–41

Faria J, Silva C, Sousa J, Surico M, Kaymak U (2006) Distributed optimization using ant colony optimization in a concrete delivery supply chain. In: Yen GG, Lucas SM, Fogel G, Kendall G, Salomon R, Zhang BT, Coello CAC, Runarsson TP (eds) Proceedings of the 2006 IEEE Congress on Evolutionary Computation, IEEE Press, Vancouver, BC, Canada, pp 73–80

Fernandes M, Gouveia L, Voß S (2007) Determining hop-constrained spanning trees with repetitive heuristics. J Telecom Inform Technol 4:16–22

Fontes DBMM (2010) Optimal hop-constrained trees for nonlinear cost flow networks. Infor 48:13–21

Fontes DBMM, Gonçalves JF (2012) A multi-population hybrid biased random key genetic algorithm for hop-constrained trees in nonlinear cost flow networks. Optim Lett pp 1–22, DOI 10.1007/s11590-012-0505-5

Fontes DBMM, Hadjiconstantinou E, Christofides N (2003) Upper bounds for single-source uncapacitated concave minimum-cost network flow problems. Networks 41(4):221–228

Frey H, Ingelrest F, Simplot-Ryl D (2008) Localized minimum spanning tree based multicast routing with energy-efficient guaranteed delivery in ad hoc and sensor networks. In: Proceedings of the 2008 International Symposium on a World of Wireless, Mobile and Multimedia Networks, IEEE Computer Society, Washington, DC, USA, pp 1–8

Gambardella LM, Éric Taillard, Agazzi G (1999) Macs-vrptw: A multiple ant colony system for vehicle routing problems with time windows. In: New Ideas in Optimization, McGraw-Hill, pp 63–76

García-Martínez C, Cordón O, Herrera F (2007) 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

Gavish B (1983) Formulations and algorithms for the capacitated minimal directed tree problem. J ACM 30:118–132

Gonçalves JF, Resende MGC (2011) Biased random-key genetic algorithms for combinatorial optimization. J Heuristics 17(5):487–525

Gouveia L (1995) Using the Miller-Tucker-Zemlin constraints to formulate a minimal spanning tree problem with hop constraints. Comput Oper Res 22:959–970

Gouveia L, Martins P (1999) The capacitated minimal spanning tree problem: an experiment with a hop-indexed model. Ann Oper Res 86:271–294

Gouveia L, Martins P (2000) A hierarchy of hop-indexed models for the capacitated minimum spanning tree problem. Networks 35:1–16

Gouveia L, Requejo C (2001) A new lagrangean relaxation approach for the hop-constrained minimum spanning tree problem. Eur J Oper Res 132:539–552

Gouveia L, Simonetti L, Uchoa E (2007) Modelling the hop-constrained minimum spanning tree problem over a layered graph. In: In Proceedings of the International Network Optimization Conference, Spa, Belgium,

Gouveia L, Paias A, Sharma D (2011) Restricted dynamic programming based neighborhoods for the hop-constrained minimum spanning tree problem. J Heuristics 17:23–37

Hassin R, Levin A (2003) Minimum spanning tree with hop restrictions. J Algorithm 48:220–238

Hu XM, Zhang J, Xiao J, Li Y (2008) Protein folding in hydrophobic-polar lattice model: A flexible ant-colony optimization approach. Protein Peptide Lett 15:469–477

Hwang IS, Cheng RY, Tseng WD (2007) A novel dynamic multiple ring-based local restoration for point-to-multipoint multicast traffic in wdm mesh networks. Photonic Netw Commun 14:23–33

Katagiri H, Hayashida T, Nishizaki I, Ishimatsu J (2009) A hybrid algorithm based on tabu search and ant colony optimization for k-minimum spanning tree problems. In: Proceedings of the 6th International Conference on Modeling Decisions for Artificial Intelligence, Springer-Verlag, Berlin, Heidelberg, MDAI ’09, pp 315–326

Lessing L, Dumitrescu I, Stützle T (2004) A comparison between ACO algorithms for the set covering problem. In: ANTS, pp 1–12

Meshoul S, Batouche M (2002) Ant colony system with extremal dynamics for point matching and pose estimation. In: 16th International Conference on Pattern Recognition, vol 3, pp 823–826

Middendorf M, Reischle F, Schmeck H (2002) Multi colony ant algorithms. J Heuristics 8:305–320

Monteiro MSR, Fontes DBMM, Fontes FACC (2011) An ant colony optimization algorithm to solve the minimum cost network flow problem with concave cost functions. In: Krasnogor N, Lanzi PL (eds) GECCO, ACM, pp 139–146

Monteiro MSR, Fontes DBMM, Fontes FACC (2013) Concave minimum cost network flow problems solved with a colony of ants. J Heuristics 19:1–33

Mullen R, Monekosso D, Barman S, Remagnino P (2009) A review of ant algorithms. Expert Syst Appl 36:9608–9617

Musa R, Arnaout JP, Jung H (2010) Ant colony optimization algorithm to solve for the transportation problem of cross-docking network. Comput & Ind Eng 59(1):85 – 92

Neumann F, Witt C (2010) Ant colony optimization and the minimum spanning tree problem. Theor Comput Sci 411(25):2406–2413

Parpinelli RS, Lopes HS, Freitas AA (2002) Data mining with an ant colony optimization algorithm. IEEE T Evolut Comput 6:321–332

Prim R (1957) Shortest connection networks and some generalisations. AT& T Tech J 36:1389–1401

Putha R, Quadrifoglio L, Zechman E (2012) Comparing ant colony optimization and genetic algorithm approaches for solving traffic signal coordination under oversaturation conditions. Comput-Aided Civ Infrastruct Eng 27:14–28

Rappos E, Hadjiconstantinou E (2004) An ant colony heuristic for the design of two-edge connected flow networks. In: ANTS Workshop, pp 270–277

Reimann M, Laumanns M (2006) Savings based ant colony optimization for the capacitated minimum spanning tree problem. Comput Oper Res 33:1794–1822

Santos L, Coutinho-Rodrigues J, Current JR (2010) An improved ant colony optimization based algorithm for the capacitated arc routing problem. Transp Res Part B: Methodol 44(2):246–266

Stützle T, Hoos H (1997) Max-min ant system and local search for the traveling salesman problem. In: IEEE International Conference On Evolutionary Cmputation (ICEC'97), IEEE Press, Piscataway,NJ, pp 309–314

Talbi EG (2002) A taxonomy of hybrid metaheuristics. J Heuristics 8:541–564

Talbi EG, Roux O, Fonlupt C, Robillard D (2001) Parallel ant colonies for the quadratic assignment problem. Future Gener Comput Sys 17:441–449

Voß S (1999) The steiner tree problem with hop constraints. Ann Oper Res 86:321–345

Yin PY, Wang JY (2006) Ant colony optimization for the nonlinear resource allocation problem. Appl Math Comput 174:1438–1453

Zhang J, Chung H, Lo WL, Huang T (2009) Extended ant colony optimization algorithm for power electronic circuit design. IEEE T Power Electr 24:147–162

Zhou G, Gen M, Wu T (1996) A new approach to the degree-constrained minimum spanning tree problem using genetic algorithm. In: Systems, Man, and Cybernetics, 1996., IEEE International Conference on, vol 4, pp 2683–2688 vol.4

Recent FEP Working Papers

492	Mónica L. Azevedo, Óscar Afonso and Sandra T. Silva, *Endogenous growth and intellectual property rights: a North-South modelling proposal*, April 2013
491	Francisco Rebelo and Ester Gomes da Silva, *Export variety, technological content and economic performance: The case of Portugal*, April 2013
490	João Correia-da-Silva, *Impossibility of market division with two-sided private information about production costs*, April 2013
489	Meena Rambocas and João Gama, *Marketing Research: The Role of Sentiment Analysis*, April 2013
488	Liliana Araújo, Sandra Silva and Aurora A.C. Teixeira, *Knowledge Spillovers and Economic Performance of Firms Located in Depressed Areas: Does Geographical Proximity Matter?*, March 2013
487	João Correia-da-Silva, Joana Pinho and Hélder Vasconcelos, *Cartel Stability and Profits under Different Reactions to Entry in Markets with Growing Demand*, March 2013
486	Ana Pinto Borges, Didier Laussel and João Correia-da-Silva, *Multidimensional Screening with Complementary Activities: Regulating a Monopolist with Unknown Cost and Unknown Preference for Empire-Building*, February 2013
485	Carlos Seixas, António Brandão and Manuel Luís Costa, *Policy Choices by an Incumbent: A Case with Down-Up Problem, Bias Beliefs and Retrospective Voting*, February 2013
484	Pedro Mazeda Gil, Oscar Afonso and Paulo B. Vasconcelos, *Industry Dynamics and Aggregate Stability over Transition*, February 2013
483	Márcia Oliveira, Dalila B. M. M. Fontes and Teresa Pereira, *Multicriteria Decision Making: A Case Study in the Automobile Industry*, February 2013

Editorial Board (wps@fep.up.pt)
Download available at: `http://wps.fep.up.pt/wplist.php`
also in `http://ideas.repec.org/PaperSeries.html`

www.fep.up.pt

UNDERGRADUATE, MASTER AND PHD PROGRAMMES

U.PORTO
FEP FACULDADE DE ECONOMIA UNIVERSIDADE DO PORTO
EFMD

School of Economics and Management, University of Porto
Rua Dr. Roberto Frias | 4200-464 Porto | Portugal
Telephone: +351 225 571 100