Multiple criteria minimum spanning trees

Pedro Cardoso * Mário Jesus † Álberto Márquez ‡

Abstract

The $\mathcal{NP}$ multiple criteria minimum spanning tree as several applications into the network design problems. In this paper, we first introduce some properties than can help to characterize the problem, as well as to produce heuristics to solve it in a more efficient way. In the second part, we propose an application of the Multiple Objective Network optimization based on the Ant Colony Optimization (`MONACO`) algorithm to find out an approximation to the set of the non-dominated solutions of the problem. The `MONACO` algorithm uses as many pheromone trails as the number of criteria and some local operators to increase the speed of the process and the quality of the results.

1 Introduction

When developing a mathematical model, the majority of the real-world problems require the simultaneous achievement of several, often conflicting, goals, taking us into the area of multiple objective, or criteria, optimization. Due to does opposing objectives, improving some of them, usually, will worse the others, just as hardly ever exists a single solution that optimizes all of them. Therefore, a set of solutions that represents the best compromise between the conflicting objectives, usually called efficient set, is most suitable, being left the final verdict, of picking one of those solutions, to a Decision Maker [4]. Network design problems do not escape from the multiple criteria exigencies and network designers are challenged to give their best to maximize factors like: productivity, reliability, longevity, efficiency or quality. At the same time other conflicting factors, like the product final cost, the maintenance cost and the weights, are to be minimized.

In this paper, we centre our attention on a network design problem known as the spanning tree problem, focused in the multiple criteria minimum spanning trees. There are several geometric network design and application problems with solutions based on spanning trees, like for instance: VLSI circuits [1], telecommunications (IEEE-802.3D standard known as the spanning tree protocol, Quality of Service problems [13]), road network design and medical imaging. In [11] Eppstein says that the spanning trees and spanners problems can be very easily stated as:*Connect a collection of n sites by a "good" network.* Mathematically, a spanning tree is an acyclic complete connected graph. Several spanning tree problem variants are known depending on the metric used or on the restrictions applied to the network [9, 11]. An exhaustive list of variants for spanning tree problem, including several $\mathcal{NP}$ cases, can be found in [2].

The multiple objective minimum spanning tree, even in its unconstrained form, is $\mathcal{NP}$-hard and $\mathcal{NP}$-# (in the worst case can have as many solutions as the number of trees that goes up to $(n_n)^{n_n-2}$, Cayley's theorem, for the complete graph, K_{n_n}) [7]. Algorithms to scan all spanning trees have an high order of complexity with one of the best taking $\mathcal{O}(n_{ST} + n_n + n_e)$ time (where n_{ST} is the number of spanning trees, n_n the number of nodes and n_e the number of edges) and $\mathcal{O}(n_n + n_e)$ space [12] which, usually, became untreatable for a small networks. Brute force algorithm, like the recursion algorithm from Ramos et al. [**?**], can only solve, in reasonable time, very small problem instances.

*Dept. de Engenharia Electrotecnica, EST, Universidade do Algarve, Faro, Portugal, `pcardoso@ualg.pt`
†Dept. de Engenharia Civil, EST, Universidade do Algarve, Faro, Portugal, `mjesus@ualg.pt`
‡Dept. de Matemática Aplicada I, Universidad de Sevilla, Sevilla , España, `almar@us.es`

In the last decades, following the increasing computational capacities, a set of computationally demanding meta-heuristical techniques based in stochastic processes, were developed and had very expressive success to approximate solutions for some of the most demanding optimization problems. These global optimization methods are established over biological or physical principles and are based on the use of pseudo-random processes and specific heuristics, to establish near-optimum solutions for the proposed problems. This kind of meta-heuristics proved to be versatile and robust when well adapted. Some examples of this kind of algorithms are the Genetic Algorithms based on the Darwin's evolutionary theory of the species, the simulated annealing algorithms based on metal annealing process, the Ant Colony Optimization (ACO) algorithms based on the foraging behaviour of the ants colonies [6], and some others.

In this paper, we apply the Multiple Objective Network optimization based on the Ant Colony Optimization algorithm (`MONACO`) to the multiple criteria minimum spanning tree. The process uses several levels of pheromone trails (one for each objective) that are used to build the spanning trees in two phases: first a set of disjoint sub-trees are created and after this sub-trees are combined to obtain the final spanning tree.

Therefore, the remainder of the paper is organized as follow. In the second section, we introduce some preliminary results over multiple criteria decision and `MONACO` algorithm. In Section 3, we introduce some properties concerning the multiple objective minimum spanning trees. In the last sections, we explain the adaptation made to the `MONACO` algorithm to solve the present problem, present some results and draw conclusions.

2 Preliminaries

2.1 Multiple criteria decision making

Real world combinatorial problems require the optimization of multiple criteria. In this case, the feasible set of solutions, $\mathcal{S}$, is defined as a subset of the power of a finite set $A = \{a_1, a_2, \ldots, a_n\}$, $\mathcal{S} \subseteq 2^A$ (for example, in the spanning tree case A is the set of edges of a network) and for each solution $s \in \mathcal{S}$ the typical objective function is the sum objective $w_i(s) = \sum_{a \in s} z_i(a), i = 1, 2, \ldots, m$, where $z_i : A \to \mathbb{R}$ is the i component of the weight associated to each of elements of A.

Therefore, we can mathematically define the multiple criteria combinatorial optimization problem as $optimize_{s \in \mathcal{S}} \mathcal{W}(s)$ where $\mathcal{W}(s) = (w_1(s), w_2(s), \ldots, w_m(s))$ is the weight vector associated to solution s and *optimize* means that we either want to minimize or maximize the weights, w_i $(i = 1, 2, \ldots, m)$. In this paper we will always consider that all the objectives are to be minimized since that, if some of the objective, w_k, is to be maximized we can apply the duality principle, $\min -w_k$.

Now, for instance, we could define *optimize* as $\min_{s \in \mathcal{S}} \max_{i=1,2,\ldots,m} w_i(s)$, called the max-ordering problem (often also called min-max problem) or, as we shall follow in this work, the definition introduced by Vilfred Pareto called Pareto efficiency and that we will define next.

Let us first define the dominance relation, $\prec$, that is a strict partial order relation: anti-reflexive $(\forall_{s,t \in \mathcal{S}} : s \prec t \Rightarrow s \neq t)$, asymmetric $(s \prec t \Rightarrow t \not\prec s)$ and transitive $(\forall_{r,s,t \in \mathcal{S}} : r \prec s \prec t \Rightarrow r \prec t)$. A solution $s \in \mathcal{S}$ **dominates** $t \in \mathcal{S}$, and we write $s \prec t$, if $w_i(s) \leq w_i(t)$ for all $i \in \{1, 2, \ldots, m\}$ and for at least one $j \in \{1, 2, \ldots, m\}$ we have $w_j(s) < w_j(t)$.

Given the dominance definition, we can define the Pareto set as the best collection of solutions, in the sense that, for those solutions one objective function can only be improved at the expense of increasing at least one of the others. So, a solution $s^* \in \mathcal{S}$ is called a **Pareto optimal** if there exists no other $s \in \mathcal{S}$ such that $s \prec s^*$ and the non-dominated set of the feasible solutions is the **Pareto set, Pareto front** or **efficient set**.

Due to the complexity inherent to most of the multiple criteria problems, several techniques were developed that take use of single objective algorithms. One example, is the weighted sum where, for some pondering weights $l_i > 0$, usually such that $\sum_{i=1}^{m} l_i = 1$, we want to $\min_{s \in \mathcal{S}} \sum_{i=1}^{m} l_i w_i(s)$ which,

providing the existence of an algorithm for the associated single objective problem, returns efficient solutions, as we can see in the following theorem.

Theorem 2.1. *For each weight vector* $l = (l_1, l_2, \ldots, l_m)$ *with* $l_i > 0$*, the solution of the single objective problem defined by* $\min_{s \in \mathcal{S}} \sum_{i=1}^{m} l_i w_i(s)$ *is a Pareto solution of* $\text{optimize}_{s \in \mathcal{S}} \mathcal{W}(s)$.

To prove this result, suppose that s^* is a solution of $\min_{s \in \mathcal{S}} \sum_{i=1}^{m} l_i w_i(s)$ but is not a Pareto solution. Then, exists $s \in \mathcal{S}$ such that $s \prec s^*$, i.e., $w_i(s) \leq w_i(s^*)$ for all $i \in \{1, 2, \ldots, m\}$ and exists $j \in \{1, 2, \ldots, m\}$ such that $w_j(s) < w_j(s^*)$. Therefore, for vector $l = (l_1, l_2, \ldots, l_m)$, with $l_i > 0$, we have $\sum_{k=1}^{m} l_k w_k(s) < \sum_{k=1}^{m} l_k w_k(s^*)$ which contradicts our hypotheses that s^* is a solution of $\min_{s \in \mathcal{S}} \sum_{i=1}^{m} l_i w_i(s)$ and, therefore, s^*, solution of $\min_{s \in \mathcal{S}} \sum_{i=1}^{m} l_i w_i(s)$, must belong to the Pareto front.

Although Theorem 2.1 guarantees efficient solutions, it can not be used to solve all multiple objective problems since that, for instance, different weight vectors need not necessarily lead to distinct solutions, as well as the associated single-objective problem might not have an efficient algorithm to solve it.

2.2 Swarm Intelligence, ACO and MONACO

Some animals, like the insects, are not considered to be intelligent. Besides, the myth of the Ant Queen, well personified for instance in the Antz film, does not have any relationship to the real ant colony conduct, since that, neither the queen nor any of the ants has a command role in the colony dynamics. However, the group behaviour of some of those social creatures, acting as a swarm, reveal aptitude to solve very complex tasks [8]. Those swarms are complex adaptive systems that display emergent behaviour and are the base idea for the called Swarm Intelligence algorithms. These algorithms mimic those group actions to solve optimization problems, becoming an area of high interest in the last few years. In general, the Swarm Intelligence algorithms employ a set of unfussy agents that react to environmental signals, to solve the proposed problems, using environmental changes introduced by other of agents, thus acting locally to produce complex global behaviour. Furthermore, the use of multiple agents as the advantage of searching for solution in multiple places at the same time, having the control distributed, as well as their robustness and flexibility, besides the fact that most of the times this algorithms can be parallelized in a very straight and efficient way.

The Ant Colony Optimization (ACO) algorithms are, probably, the more spread Swarm Intelligence algorithms [6]. Introduced by Marco Dorigo, the ACO is one of the most recent meta-heuristics that, as the name suggests, mimics the forager behaviour of the ants' colonies. To communicate the ants use a chemical substance called pheromone to guide the ants between the anthole and the food. As in the natural process, the artificial process is based in a set of artificial ants that communicate using artificial trails of pheromones. Those trails reflect the experience of the agents that have already solved the problem and favour the creation of new solutions. The method comprises a set of iterations where collections of solutions are obtained. At the end of each iteration, the pheromone trails are updated considering the known solutions as well as a certain pheromone evaporation. The ACO as been applied to some of the most demanding combinatorial optimization problems like: the TSP, the Quadratic Assignment problem, the Vehicle Routing problem [6] and logic circuit design (a more extensive list of application as well as the respective references can be found in [5]).

In [3] an multiple objective algorithm based on the ACO algorithms called MONACO was used to simulate network flows. Compared to the ACO, the main differences were the use of multiple levels of pheromones, the existence of the Pareto set of solutions and the use of explicit local optimizers to improve the ants' solutions. In Figure 1 we can see sketched a low-level description of the MONACO procedure.

In graph problems, those multiple pheromone trails present in each edge, usually represent the value of that edge in the construction of good solution. Those values are strictly connected both, to the number of times that the edge as belonged to solutions retrieved by the algorithm as well as to the quality of those solutions. The pheromone trails are updated at the final of each iteration using the formula: $\tau_k(u, v) = \rho_k \tau_k(u, v) + \Delta\tau_k(u, v)$, for $k = 1, 2, \ldots, m$ and $(u, v) \in \mathcal{E}$, where $\tau_k(u, v)$ is the

- *Initialize the pheromone trails and the Pareto set*
- *WHILE stopping criteria is not met DO*
  - *FORALL ants DO*
    - *Construct a new solution using the pheromone trails and apply local optimizer operators*
    - *Evaluate the solution and update Pareto set*
  - *Update the pheromone trails.*

Figure 1: Low-level description of the `MONACO` Algorithm

quantity of pheromone associated to the k weight on edge (u, v), $0 \leq \rho_k \leq 1$ are persistence factors for the pheromone trails ($1 - \rho_k$ is the evaporation factor) and $\Delta\tau_k(u, v)$ is the quantity of pheromone placed by the ants. This quantity is in the inverse proportion to the solution k-weight. For instance, if $\mathcal{S}$ is the set of solutions built in some iteration, $w_k(t)$ the k-weight of solution $t \in \mathcal{S}$ and $\mathcal{S}_{uv} = \{t \in \mathcal{S} : (u, v) \in t\}$ is the set of solution to which (u, v) belongs, then $\Delta\tau_k(u, v) = \sum_{t \in \mathcal{S}_{uv}} \frac{Q}{w_k(t)}, k = 1, 2, \ldots, m$, where Q is some constant related to the pheromone quantity that each ant leaves (usually a value of the same magnitude of the solution).

3 Multiple criteria minimum spanning trees

Having an undirected graph $\mathcal{N} = (\mathcal{V}, \mathcal{E}, \mathcal{Z})$, where $\mathcal{V} = \{1, 2, \ldots, n\}$ is the set of nodes, $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is the set of edges and $\mathcal{Z} : \mathcal{E} \to \mathbb{R}^m$ such that $\mathcal{Z}(e) = (z_1(e), z_2(e), \ldots, z_m(e))$ is the weight vector associated to the edges, we can define a spanning tree as any connected acyclic subgraph of $\mathcal{N}$. Let $\mathcal{T}_\mathcal{N} \subseteq 2^\mathcal{E}$ be the set of all spanning trees over $\mathcal{N}$.

For any spanning tree, T, the sum objective function also called weight or cost vector of T, is equal to $\mathcal{W}(T) = (w_1(T), w_2(T), \ldots, w_m(T)) = \left(\sum_{e \in T} z_1(e), \ldots, \sum_{e \in T} z_m(e)\right) = \sum_{e \in T} \mathcal{Z}(e)$. For $m = 1$ a spanning tree of $\mathcal{N}$ is called a minimum spanning tree if its weight is minimal over all spanning trees weights. For the multiple objective case, $m > 1$, we use the definitions of dominance and Pareto's optimality (Section 2.1) to define the efficient set of the multiple criteria optimization problem, mathematically defined as $\min^*_{T \in \mathcal{T}_\mathcal{N}} \ \mathcal{W}(T)$ where $\min^*$ indicates that all objectives are to be minimized.

3.1 Some results on multiple objective minimum spanning trees

In this section we introduce some results to characterize the problem. Let us consider that $e_{uv} \in \mathcal{E}$ is the edge defined by nodes u and v of $\mathcal{V}$, $\mathcal{E}_u$ the set of edges with origin u, $M = \{1, 2, \ldots, m\}$, $\mathcal{V}_T$ is the set of nodes of subgraph T, $\mathcal{E}_T$ the set of edges of T, $\mathcal{E}|_T$ is the set of edges defined in $\mathcal{N}$ by $\mathcal{V}_T$ and $\mathcal{N}|_T = (\mathcal{V}_T, \mathcal{E}|_T, \mathcal{Z})$. To simplify our notation, let $T' = T - \{e\} \cup \{f\}$ represent the graph obtained by removing edge e and adding edge f to a graph T, i.e., $T' = (\mathcal{V}, \mathcal{E}_T - \{e\} \cup \{f\}, \mathcal{Z})$.

Definition 3.1. A subtree T is efficient if T is efficient over $\mathcal{N}|_T$.

By Definition 3.1 a tree $T \in \mathcal{T}_\mathcal{N}$ is efficient if there is no other solution $S \in \mathcal{T}_\mathcal{N}$ such that S dominates T.

Lemma 3.2. *A spanning tree is efficient if and only if all its subtrees are efficient.*

Proof. Suppose that T has a subtree T' that is not efficient over $\mathcal{N}|_{T'} = (\mathcal{V}_{T'}, \mathcal{E}|_{T'}, \mathcal{Z})$. Then, exists $S' \in \mathcal{N}|_{T'}$ such that $S' \prec T'$, that is, $w_i(S') \leq w_i(T')$ for all $i \in M$ and exists $j \in M$ such that $w_j(S') < w_j(T')$. Now, for $S = T - T' \cup S'$ we have $w_i(S) = w_i(T) - w_i(T') + w_i(S') \leq w_i(T)$ for all $i \in M$ and $w_j(S) = w_j(T) - w_j(T') + w_j(S') < w_j(T)$ for some $j \in M$ which means that S dominates T and consequently we have a contradiction.

To prove the reciprocal, suppose that all subtrees of a tree are efficient and T is not efficient. Then exists S that dominates T and, since T is a subtree of it self, we have a contradiction which means that T must be efficient. □

Lemma 3.3. *Let e be an edge of a non empty subgraph T of $\mathcal{N}$. If f is an edge of $\mathcal{E} - \mathcal{E}_T$ and f dominates e, $f \prec e$, then the graph T' obtained by adding f to T and removing e from T, $T' = T - \{e\} \cup \{f\}$, dominates T.*

Proof. Since f dominates e then $z_i(f) \leq z_i(e)$ for all $i \in M$ and $z_j(f) < z_j(e)$ for some $j \in M$. Therefore, from the first equation for all $i \in M$ we have $w_i(T') = w_i(T) + z_i(f) - z_i(e) \leq w_i(T)$ and from the second there is the guarantee that exists $j \in M$ such that $w_j(T') = w_j(T) + z_j(f) - z_j(e) < w_j(T)$, which proves that T' dominates T. □

A **cut** of $\mathcal{V}$ is a partition of $\mathcal{V}$ in two non empty sets $\mathcal{V}_1$ and $\mathcal{V}_2$, i.e., $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$ and $\mathcal{V}_1 \cap \mathcal{V}_2 = \emptyset$. Let us also denote the set of the edges incident with one node in $\mathcal{V}_1$ and a node in $\mathcal{V}_2$ by $\mathcal{E}(\mathcal{V}_1, \mathcal{V}_2)$.

Lemma 3.4. *Let T be a tree, $e_{uv} \in \mathcal{E}_T$ and $\mathcal{V}_u, \mathcal{V}_v$ the cut induced by the remotion of e_{uv} from T. If $e_{wz} \in \mathcal{E}(\mathcal{V}_u, \mathcal{V}_v)$ then $T' = T - \{e_{uv}\} \cup \{e_{wz}\}$ is a tree.*

Proof. Let T_u and T_v be the subtrees of T defined by $\mathcal{V}_u$ and $\mathcal{V}_v$, respectively. Since T is a tree, T_u and T_v are also trees (they are subtrees of T) and by hypothesis they are node-disjoint. Therefore, for $e_{wz} \in \mathcal{E}(\mathcal{V}_u, \mathcal{V}_v)$, $T' = (\mathcal{V}, \mathcal{E}_{T_u} \cup \mathcal{E}_{T_v} \cup \{e_{wz}\}, \mathcal{Z})$ can not contain any cycles and is connected which proves that T' is a tree. □

If $\mathcal{N}$ is a connected network we say that an edge $e_{uv} \in \mathcal{E}$ is a **bridge** if by removing it the network becomes disconnected.

Definition 3.5. Let $\mathcal{N} = (\mathcal{V}, \mathcal{E}, \mathcal{Z})$ be a network. An edge $e_{uv} \in \mathcal{E}$ is **local** or **node efficient** if e_{uv} is not dominated by any edge in $\mathcal{E}_u$ ($\forall_{e_{uw} \in \mathcal{E}_u} : e_{uw} \not\prec e_{uv}$) and is **local** or **node dominant** if e_{uv} dominates all edges in $\mathcal{E}_u - \{e_{uv}\}$ ($\forall_{e_{uw} \in \mathcal{E}_u - \{e_{uv}\}} : e_{uv} \prec e_{uw}$).

The following two theorems show that in some conditions an edge can be present in all the efficient spanning trees.

Theorem 3.6. *If e_{uv} is local dominant or e_{uv} is a bridge then e_{uv} belongs to all the efficient trees.*

Proof. The case in which e_{uv} is a bridge is trivial. Suppose now, that e_{uv} is node dominant, T is an efficient tree from $\mathcal{T}_{\mathcal{N}}$, $(u, u_1, u_2, \ldots, u_p, v)$ is the path from u to v in T, e_{uv} does not belong to T and S is the tree (Lemma 3.4) obtained by replacing edge e_{uu_1} by edge e_{uv} in T. Now, using Lemma 3.3 and since e_{uv} dominates e_{uu_1}, we can conclude that S dominates T. Therefore, T is not an efficient tree, which contradicts our hypothesis and therefore e_{uv} must belong to all efficient trees. □

Theorem 3.7. *If for some cut $\mathcal{V}_1$, $\mathcal{V}_2$ of $\mathcal{V}$ there is an edge e_{uv} in $\mathcal{E}(\mathcal{V}_1, \mathcal{V}_2)$ such that e_{uv} dominates all edges in $\mathcal{E}(\mathcal{V}_1, \mathcal{V}_2) - \{e_{uv}\}$ then e_{uv} belongs to all efficient spanning trees.*

Proof. The case in which $\mathcal{E}(\mathcal{V}_1, \mathcal{V}_2) - \{e_{uv}\} = \emptyset$ is trivial that e_{uv} belongs to any spanning tree, since it is a bridge. Suppose now that $\mathcal{E}(\mathcal{V}_1, \mathcal{V}_2) - \{e_{uv}\} \neq \emptyset$, e_{uv} does not belong to some efficient tree T and let $e_{wz} \in \mathcal{E}(\mathcal{V}_1, \mathcal{V}_2) - \{e_{uv}\}$ be an edge in T that connects $\mathcal{V}_1$ to $\mathcal{V}_2$. Then, by Lemma 3.4, $S = T - \{e_{wz}\} \cup \{e_{uv}\}$ is a spanning tree and, since $e_{uv} \prec e_{wz}$, from Lemma 3.3 can conclude that S dominates T which contradicts our hypothesis that T is an efficient spanning tree. Thus e_{uv} must belong to all spanning trees. □

If T is a tree and e an edge not contained in T then $T \cup \{e\}$ contains an unique cycle denoted by $C_T(e)$.

Theorem 3.8. *If a spanning tree T is efficient and e belongs to $\mathcal{E} - \mathcal{E}_T$ then e does not dominate any edge in the cycle $C_T(e)$, i.e., $\forall_{e \in \mathcal{E} - \mathcal{E}_T} \forall_{f \in C_T(e)} : e \not\prec f$.*

Proof. Suppose that T is an efficient spanning tree and that exists an $e \in \mathcal{E} - \mathcal{E}_T$ and a $f \in C_T(e)$ such that $e \prec f$. Now, by Lemma 3.3, the tree $S = T - \{f\} \cup \{e\} \prec T$, since $e \prec f$. This contradicts our hypothesis so, T is an efficient spanning tree. □

Theorem 3.9. *If T is a spanning tree such that for all edges e of $\mathcal{E} - \mathcal{E}_T$ and for all edges f of $C_T(e) - \{e\}$ we have $f \prec e$ then T is the unique efficient spanning tree.*

Proof. Let S_0 be a spanning tree, $\mathcal{E}_{S_0} - \mathcal{E}_T \neq \emptyset$ (if it was the empty set then $S_0 = T$) and e and edge in $\mathcal{E}_{S_0} - \mathcal{E}_T$. If we remove e from S_0, we stay with two subtrees of S_0: S_0' and S_0''. On the other hand, we know that e is dominated by all edges in the path $C_T(e) - \{e\}$ and $S_0' \cup S_0'' \cup C_T(e) - \{e\}$ will be connected again (not necessarily a tree). If e' is the edge that connects S_0' with S_0'' and $S_1 = S_0 - \{e\} \cup \{e'\}$ is the tree obtained by removing e from S_0 and adding e' then $\mathcal{W}(S_1) = \mathcal{W}(S_0) - \mathcal{Z}(e) + \mathcal{Z}(e')$. Now, since by hypothesis $e' \prec e$ then for all $i \in M$ it verifies $z_i(e) \leq z_i(e')$ and exist $j \in M$ such that $z_j(e) < z_j(e')$, which implies that $w_i(S_1) \leq w_i(S_0)$ for all $i \in M$ and exist $j \in M$ such that $w_j(S_1) < w_j(S_0)$ and, therefore, $S_1 \prec S_0$. Now, we can repeat this process while $S_i - T \neq \emptyset$, obtaining a sequence of trees such that $S_k \prec \cdots \prec S_2 \prec S_1 \prec S_0$, with $k \leq n - 1$. At the end $S_k = T$, since $S_k - T = \emptyset$, which implies that any tree is dominated by T. □

4 MONACO applied to the multiple objective minimum spanning tree problem

The construction of the spanning trees by MONACO is divided in two phases: in the first phase a set of disjoint subtrees is built and in the second phase those trees are joined to form a final tree. In both phases the process uses m pheromone trails associated to the m weights, $\tau^{(k)}$ $(k = 1, 2, \ldots, m)$, and a local heuristic to improve search procedure. At the end, a local optimizer is applied to improve the solution.

Disjoint subtree In this phase the process determines a set of disjoint spanning trees. It starts by randomly choose a node, u_0, that does not belong to any subtree. From that node we create a path, which is a subtree, through the order addition of nodes using the probabilistic formula:

$$p_{u_i u_{i+1}} = \frac{\prod_{k=1}^{m} \tau_k(u_i, u_{i+1})^{\alpha_k} w_k(u_i, u_{i+1})^{-\beta_k}}{\sum_{z \in \{v:(u_i,v) \in \mathcal{E}\}} [\prod_{k=1}^{m} \tau_k(u_i, z)^{\alpha_k} w_k(u_i, z)^{-\beta_k}]}, (u_i, u_{i+1}) \in \mathcal{E}, \tag{1}$$

where $\alpha_k \in \mathbb{R}_0^+ (k = 1, 2, \ldots, m)$ and $\beta_k \in \mathbb{R}_0^+ (k = 1, 2, \ldots, m)$ are parameters related to the relative importance of cost k and of the local heuristic, $w_k(u_i, v)^{-\beta_k}$, , respectively. This local heuristic favours the addition of edges with lower weights. The spanning tree construction stops when the selected node already belongs to some subtree. When that happens, if the last selected node does not belong to the tree in construction, then we obtain a subtree by joining the subtree under construction with the intersected tree. Otherwise, the chosen node already belongs to the tree in construction that is disjoint from any other. In this case, this tree is kept and later joined with the other disjoint subtrees. The process is repeated until all nodes belong to some subtree.

Fusion of the subtrees From the previous phase we have a set of subtrees $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$. Now, we start by picking one subtree from $\mathcal{T}$, T_i, and then we look for a good connection to one of the other trees, $\mathcal{T} - \{T_i\}$. That connection is made by randomly choosing an edge e_{uv} such that $u \in T_i$ and $v \notin T_i$, i.e., $v \in t_j$, with $t_j \in \mathcal{T} - \{T_i\}$, using formula

$$p_{uv} = \frac{\prod_{k=1}^{m} \tau_k(u, v)^{\alpha_k} w_k(u, v)^{-\beta_k}}{\sum_{(x,y) \in \mathcal{Q}} [\prod_{k=1}^{m} \tau_k(x, y)^{\alpha_k} w_k(x, y)^{-\beta_k}]}, (u, v) \in \mathcal{Q}, \tag{2}$$

where $\mathcal{Q} = \{(x, y) \in \mathcal{E} : x \in T_i \wedge y \notin T_i\}$. This process is repeated until there is only on tree. In example 4.1 we can see a simulation of the process for a 8 node network.

Example 4.1. Consider the network in Figure 2 with associated edges weights (w_1, w_2). Suppose also that the process has been running for some time, with parameters $\alpha_1 = 1$, $\alpha_2 = 2$, $\beta_1 = 2$ and $\beta_2 = 2$, and the pheromone trails have values given by (τ_1, τ_2). Those values are resumed in Table 1, with entries of the form $(w_1, w_2, \tau_1, \tau_2)$ and the process to build a tree (Figure 2) can be described as follow:(a) Randomly select node: 1; The probabilities of adding each of the edges adjacent to node 1 are $p_{1,2} = \frac{(\frac{1}{1})^2 5(\frac{1}{2})^2 4^2}{(\frac{1}{1})^2 5(\frac{1}{2})^2 4^2 + (\frac{1}{3})^2 1(\frac{1}{3})^2 2^2 + (\frac{1}{2})^2 4(\frac{1}{2})^2 4^2 + (\frac{1}{1})^2 5(\frac{1}{2})^2 4^2 + (\frac{1}{2})^2 2(\frac{1}{3})^2 2^2 + (\frac{1}{5})^2 1(\frac{1}{4})^2 1^2 + (\frac{1}{7})^2 1(\frac{1}{5})^2 2^2} = 0.452$, $p_{1,3} = 0.001$, $p_{1,4} = 0.090$, $p_{1,5} = 0.452$, $p_{1,6} = 0.005$, $p_{1,7} = 0.0$ and $p_{1,8} = 0.0$: edge $e_{1,2}$ is added; (b) From node 2 the probabilities are $p_{2,1} = 0.293$, $p_{2,4} = 0.704$ and $p_{2,5} = 0.003$: edge $e_{2,4}$ is added; (c) From node 4 the probabilities are $p_{4,1} = 0.044$, $p_{4,2} = 0.536$ and $p_{4,6} = 0.419$: edge $e_{4,2}$ is selected; Since it would create a cycle we stop the construction of this subtree and randomly choose other node to restart the process: node 5; (d) From node 5 the probabilities are $p_{5,1} = 0.389$, $p_{5,2} = 0.004$ and $p_{5,7} = 0.607$: edge $e_{5,1}$ is added; (e) Since node 1 belongs to other subtree, the two subtree (the one in construction and the one to which node 1 belongs) are merged; (f) The process in steps (a) to (e) is repeated until all nodes belong to one subtree; (g) Randomly choose a subtree, for example T_1 and connect it to other according to probabilities $p_{1,3} = 0.001$, $p_{1,6} = 0.003$, $p_{1,7} = 0.0$, $p_{1,8} = 0.0$, $p_{4,6} = 0.543$ and $p_{5,7} = 0.453$: selected edge $e_{5,7}$; (h) Repeat last step until all subtrees are connected; (i) Final tree.

	1	2	3	4	5	6	7	8
1		(1,2,5,4)	(3,3,1,2)	(2,2,4,4)	(1,2,5,4)	(2,3,2,2)	(5,4,1,1)	(7,5,1,2)
2	(1,2,5,4)			(1,1,3,4)	(3,3,2,3)			
3	(3,3,1,2)					(2,3,4,4)		(1,2,4,5)
4	(2,2,4,4)	(1,1,3,4)				(2,1,6,5)		
5	(1,2,5,4)	(3,3,2,3)					(1,2,5,5)	
6	(2,3,2,2)		(2,3,4,4)	(2,1,6,5)				
7	(5,4,1,1)				(1,2,5,5)			(1,3,2,3)
8	(7,5,1,2)		(1,2,4,5)				(1,3,2,3)	

Table 1: Network and pheromone trail values: entries of the form $(w_1, w_2, \tau_1, \tau_2)$

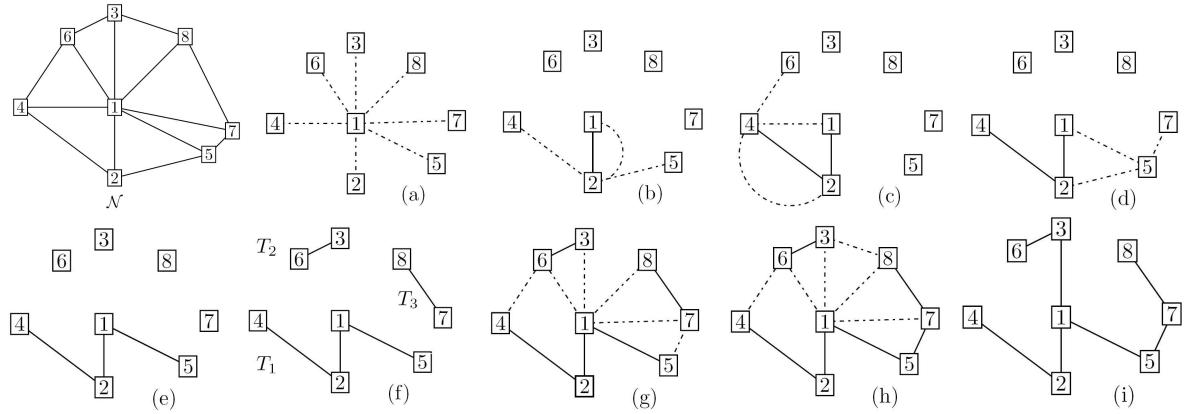
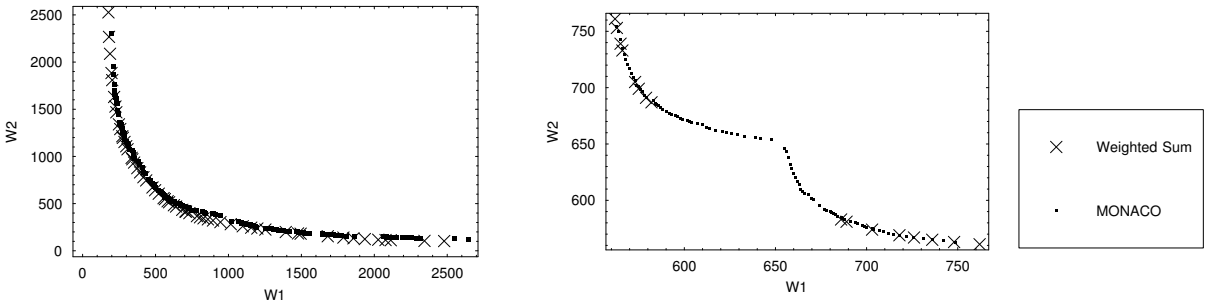


Figure 2: Simulation of the building tree process.

Local optimization operator To accelerate the process and improve the quality of the solution we use a local search operator. The procedure consists of trying to replace each edge e_{uv} of the tree by an edge of $\mathcal{E}(\mathcal{V}_u, \mathcal{V}_v)$ such that the new tree is improved. The selected edge is the one that produces the best improvement of the tree, for some weight randomly selected.

5 Computational results, conclusions and future work

Computacional results To test our method we used two types of complete graphs with integer weights. Those weights were generated randomly (type 1) and such that the Pareto front is concave (type 2). The last case is based in the graph generator presented in [10] for the k-degree problem. To compare with our results, we used the efficient solutions obtained by the weighted sum with a single objective algorithm to the minimum spanning tree problem (see Theorem 2.1) over 100 combinations of the weight parameters, l_i. In Figure 3, we can see the Pareto fronts obtained for two networks with 50 nodes each.



(a) (b)

Figure 3: Pareto front elements for: (a) the random and (b) the concave case. In case (b) we see that the `MONACO` algorithm provides solutions of the concave region.

Conclusions and future work For the tested problems, the `MONACO` algorithm provides good approximations to the efficient solutions, as well as it can find solutions in the concave regions of the Pareto front. We also could see that for some problems, the use of the weighted sum with a single objective algorithm provides a good Pareto front in a fraction of the time needed by the `MONACO`.

In the future, we would like to improve our theoretical results and apply them to formulas (1) and (2), as well as in the local optimizer operator, in a more effective way. We would also like to produce a benchmark to the problem in study.

References

[1] M. Atallah, *Algorithms and Theory of Computation Handbook*, CRC Press, 1999.

[2] G. Ausiello, A. Marchetti-Spaccamela, G. Gambosi, M. Protasi, P. Crescenzi, Viggo Kann, *Complexity and Approximation: Combinatorial Optimization Problems and their Approximability Properties*, Springer-Verlag, Berlin, 1999.

[3] P. Cardoso, M. Jesus, A. Márquez, *MONACO - Multi-objective Network Optimisation based on an ACO*, X Encuentros de Geometria Computacional, Universidad de Sevilla, Sevilla, 2003.

[4] K. Deb, *Multi-objective Optimization using Evolutionary Algorithms*, John Wiley & Sons, 2001.

[5] M. Dorigo, E. Bonabeau, G. Theraulaz, *Ant Algorithms and Stigmergy*, 16, Future Generation Computer Systems, Elsevier, 2000.

[6] M. Dorigo, E. Bonabeau, G. Theraulaz, *Swarm Intelligence: From Natural to artificial Systems*, Oxford University Press, 1999.

[7] H. Hamacher, G. Ruhe, *On spanning tree problems with multiple objective*, Annals of operation research, Kluwer, 1994.

[8] S. Johnson, *Emergence. The connected lives of ants, brains, cities, and software*, Scribner, 2001.

[9] D. Jungnickel, *Graphs, Networks and Algorithms*, Springer-Verlag, Berlin, 1999.

[10] J. Knowles, D. Corne, *Benchmark Problems Generators and Results for the Multiobjective Degree-Constrained Minimum Spanning Tree Problem*, Genetic and Evolutionary Computation Conference (GECC-2001), 2001.

[11] J.-R. Sack, J. Urrutia *Handbook of Computational Geometry*, North-Holland, 2000.

[12] A. Shioura, A. Tamura, T. Uno, *An Optimal Algorithm for Scanning All Spanning Trees of Undirected Graphs*, Journal on Computing, SIAM, 1997.

[13] H. Yu and S. Das, Y. Lim, M. Gerla, *Efficient Building Method of Multiple Spanning Tree for QoS and Load Balancing*, GLOBECOM 2003, IEEE, 2003.