AN ANT COLONY SYSTEM APPROACH FOR SOLVING THE AT-LEAST VERSION OF THE GENERALIZED MINIMUM SPANNING TREE PROBLEM

Arindam K. Das

University of Washington
arindam@ee.washington.edu

Payman Arabshahi, Andrew Gray

Jet Propulsion Laboratory
{payman,gray}@jpl.nasa.gov

ABSTRACT

We consider the "at least" version of the Generalized Minimum Spanning tree problem (L-GMST). Unlike the MST, the L-GMST is known to be NP-hard. In this paper, we propose an ant colony system based solution approach for the L-GMST. A key feature of our algorithm is its use of ants of different behavioral characteristics, which are adapted over time. Computational results on datasets used in earlier literature indicate that our algorithm provides similar or better results for most of them.

1. INTRODUCTION

Given an undirected graph $G = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is the set of nodes and $\mathcal{E}$ the set of edges, and a symmetric edge cost matrix $\mathbf{C} = [\mathbf{C}_{ij}]$, $\mathbf{C}_{ij} \in \mathbb{R}^+$, $(i,j) \in \mathcal{E}$, the *Minimum Spanning Tree* (MST) problem seeks to find a spanning tree on the nodes such that the cost of the tree is minimum. The MST problem is solvable in polynomial time and techniques, *e.g.*, via Prim's and Kruskal's algorithms [1].

An extension to the MST, the *Generalized Minimum Spanning Tree* (GMST) problem, was suggested by Myung *et al* [2]. In GMST, the nodes are divided into C clusters and a minimum weight tree is sought spanning the C clusters using *exactly* one node from each cluster. This version of the GMST is also referred to as the E-GMST (to emphasize the "exactly one node per cluster" condition). Another variant of the GMST, introduced by Ihler *et. al.* [3], is the L-GMST, where a minimum weight spanning tree is sought on the clusters using *at least* one node from each cluster. Figure 1 illustrates both these versions. The GMST problem finds applications in diverse areas including telecommunications, transportation engineering, and biology.

Similar generalizations can be made to other commonly studied network design and combinatorial optimization problems, *e.g.*, the *Generalized Steiner Tree* problem, *Generalized Traveling Salesman* problem and the *Generalized Shortest Path* problem. Discussions on these can be found in [12].

In this paper, we concentrate on the L-GMST problem, although our proposed algorithm can be extended straightforwardly to the E-GMST with only minor modifications to the edge selection rules. Unlike the MST, both versions of the GMST have been shown to be *NP*-hard [2, 3]. Specifically for the L-GMST, it is shown in [3] that no constant factor polynomial time algorithm exists unless $\mathcal{P} = \mathcal{NP}$. While most research regarding the GMST has focused on the E-GMST [2, 4, 7, 8, 11], some solution methods have also been reported in the literature for solving the L-GMST problem. These include integer linear programming (ILP), local search and metaheuristics such as tabu search and genetic algorithms. ILP formulations for the L-GMST were first given by Dror *et. al.* [5]. However, two out of their three models were subsequently shown to be invalid by Feremans *et. al.* [8]. Another branch-and-cut based exact algorithm was suggested by Feremans [9]. Transformational techniques to convert an instance of the L-GMST problem to the E-GMST are also discussed in [9]. Besides the ILP formulations, Dror *et. al.* [5] also proposed four heuristic algorithms and a genetic algorithm based optimization procedure for the L-GMST. Two other polynomial time heuristics have been suggested by Ihler *et. al.* in [3].

Our proposed algorithm is based on the *swarm intelligence* paradigm which appears in biological swarms of certain insect species and gives rise to intelligent behavior through complex interaction of thousands of autonomous swarm members. A main principle behind swarm interaction is *stigmergy*, or communication through the environment. An example is *pheromone* laying on trails followed by ants. Pheromone is a potent form of hormone that can be sensed by ants as they travel along trails. It attracts ants which therefore tend to follow trails that have high pheromone concentrations. This causes an autocatalytic reaction, *i.e.*, one that is accelerated by itself. Ants attracted by the pheromone will lay more pheromone on the same trail, causing even more ants to be attracted. Swarm intelligence paradigms thus use positive reinforcement as a search strategy. The Ant Colony System (ACS) algorithm, an optimization procedure inspired by swarm intelligence principles, was originally proposed by Dorigo and Gambardella [13] for solving the celebrated traveling salesman problem.

2. THE ACS ALGORITHM FOR L-GMST

2.1. Notation

The following notation will be used in this paper:

t	= time index
t_{MAX}	= maximum time index
$N_{ants(A)}(t)$	= number of Type-A ants at time t
$N_{ants(B)}(t)$	= number of Type-B ants at time t
$\mathcal{N}$	= set of all nodes in the input graph
$\lvert\mathcal{N}\rvert$	= cardinality of $\mathcal{N}$ = N
$\mathcal{E}$	= set of all bidirected edges in the input graph
$\lvert\mathcal{E}\rvert$	= cardinality of $\mathcal{E}$ = E
$\mathcal{C}$	= set of all clusters in the input graph
$\lvert\mathcal{C}\rvert$	= cardinality of $\mathcal{C}$ = C
$\mathcal{N}(\mathcal{C}_c)$	= set of all nodes in cluster $\mathcal{C}_c$

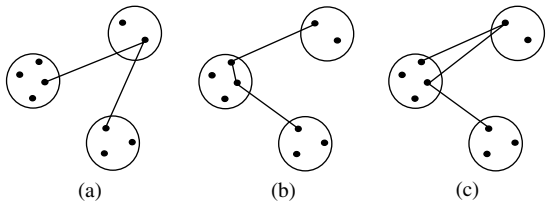

Figure 1. (a) A feasible solution for the E-GMST problem, where *exactly* one node is used from each cluster. (b) A feasible solution for the L-GMST problem, where *at least* one node is used from each cluster. (c) Another feasible solution for the L-GMST. If two nodes are chosen from a cluster, it is possible for them to be joined by inter-cluster edges, as shown here. In many applications, however, the weights of inter-cluster edges are much greater than the weights of the intra-cluster edges, and a solution such as this will not be the optimal L-GMST.

$\lvert\mathcal{N}(\mathcal{C}_c)\rvert$	= number of nodes in cluster $\mathcal{C}_c$
$\alpha(\mathcal{C}_c)$	= node density of cluster $\mathcal{C}_c := \lvert\mathcal{N}(\mathcal{C}_c)\rvert/N$
$\mathcal{E}(\mathcal{C}_c)$	= set of edges whose end nodes are in cluster $\mathcal{C}_c$ (also referred to as *intra-cluster* edges)
$\bar{\mathcal{E}}$	= set of all *inter-cluster* edges $:= \mathcal{E} \setminus \bigcup_c \mathcal{C}_c$
$F(i, \mathcal{C}_c)$	= largest forest in cluster $\mathcal{C}_c$, rooted at node i ($i \in \mathcal{N}(\mathcal{C}_c)$, all edges in $F(i, \mathcal{C}_c)$ are in $\mathcal{E}(\mathcal{C}_c)$)
$\mathcal{N}(F(i, \mathcal{C}_c))$	= set of all nodes in $F(i, \mathcal{C}_c)$
$\tau_{ij}(t)$	= pheromone level on the edge $i \rightarrow j$ at time t
$\mathbf{W}_{ij}$	= weight of the link between nodes i and j
η_{ij}	= visibility of node j from node $i := 1/\mathbf{W}_{ij}$
$\beta_A(t)$	= tunable parameter to control η_{ij} for Type-A ants, possibly time-varying, $0 < \beta_A \leq 1$
$\beta_B(t)$	= tunable parameter to control η_{ij} for Type-B ants, possibly time-varying, $0 < \beta_B < \beta_A \leq 1$
$T_m(t)$	= tree developed by ant m at time t
$Y_m(t)$	= cost of $T_m(t)$
ρ	= pheromone decay coefficient, $\rho \in (0, 1]$
γ	= pheromone delivery coefficient (> 0)
q	= uniformly distributed random variable over the interval [0,1]
$q_0(t)$	= tunable parameter, $q_0(t) \in [0, 1], \forall t$

In the following section, we discuss the tree building mechanism for each ant, given an input graph $G = (\mathcal{N}(\mathcal{C}_c), \mathcal{E} : c \in \mathcal{C})$ and a symmetric edge weight matrix $\mathbf{W}$. We assume that there is no overlap between the clusters, or, any node can be a member of only one cluster. All elements of $\mathbf{W}$ are assumed to be real and non-negative. If an edge does not exist between nodes i and j, $\mathbf{W}_{ij} = \infty$. We also assume that the elements of the weight matrix satisfy the triangle inequality; *i.e.*, if there are edges between the node pairs (i, j), (j, k) and (i, k), then we have:

$$\mathbf{W}_{ij} + \mathbf{W}_{jk} \geq \mathbf{W}_{ik} \tag{1}$$

Else, all $\mathbf{W}_{ij}$'s for which an edge exists between i and j are replaced by the cost of the shortest path between i and j.

$$\mathbf{W}_{ij} = \begin{cases} \min(cost(SP(i,j)), \mathbf{W}_{ij}), & \text{if } (i \leftrightarrow j) \in \mathcal{E} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where $cost(SP(i, j))$ denotes the cost of the shortest path between nodes i and j. Note that the above transformation is not equivalent to computing the shortest path matrix (or *closure matrix*) since only those elements of $\mathbf{W}_{ij}$ for which an edge exists are modified.

We will use the notation $(i \leftrightarrow j)$ to denote a bidirectional edge between nodes i and j while a directed edge from i to j is represented by $(i \rightarrow j)$. The notation (i, j) is used to refer to the node pair. With a slight abuse of notation, we also use the set $\mathcal{E}$ to refer to all directed edges, $\{i \rightarrow j\}$, in the graph.

$$(i \leftrightarrow j) \in \mathcal{E} \Rightarrow (i \rightarrow j) \in \mathcal{E} \text{ and } (j \rightarrow i) \in \mathcal{E} \tag{3}$$

2.2. Tree building by an ant

Tree building in our ACS algorithm is an iterative procedure where a population of ants (or agents) work in parallel to generate feasible solutions. For an L-MST instance with C clusters, each ant is allowed $C - 1$ iterations to generate a feasible solution. Starting from a randomly chosen initial cluster, each ant chooses an inter-cluster edge at iteration k from a set of candidate edges $\{(i \rightarrow j) \in \bar{\mathcal{E}}\}$, where i belongs to the set of nodes in clusters which have been spanned till iteration $k - 1$, denoted by $\mathbf{CS}^{k-1}$, and j belongs to the set of nodes in clusters which have not been spanned till iteration $k - 1$, denoted by $\mathbf{CNS}^{k-1}$. If the set of candidate edges is empty, the partial solution is labeled *invalid* and assigned a high cost. See Fig. 2. In our simulations, we have observed that the frequency of occurrence of invalid solutions is significantly reduced if the tail nodes in the candidate edge list are chosen from certain rooted forests within the clusters which have been spanned. This improvement is most pronounced for problem instances which are sparsely connected and have one or more clusters with a high number of connected components. We discuss this concept and the tree construction algorithm in detail below. See Fig. 3 for an illustration of the tree-building algorithm.

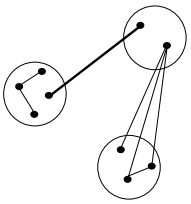

Figure 2. Generation of an invalid solution. The set of all inter-cluster and intra-cluster edges is shown. Even though a GMST clearly exists, if the first edge chosen is the one marked by a thick line, cluster 3 cannot be spanned in.

1. Initially, we mark all nodes as *unshaded* and all clusters as *not spanned*. Next, we choose an initial cluster, either randomly, or probabilistically according to the node densities of the clusters. For example, if the input graph has 3 clusters, with 1, 3 and 6 nodes, the node densities of the clusters are 0.1, 0.3 and 0.6 respectively. If the initial cluster is chosen probabilistically, cluster 3 will be chosen about 60% of the time.

Suppose the initial cluster chosen is $\mathcal{C}_a$. Label $\mathcal{C}_a$ as *spanned* and define:

$$\mathbf{CS}^0 = \mathcal{C}_a \text{ and } \mathbf{CNS}^0 = \mathbf{C} \setminus \mathbf{CS}^0$$

2. For the first iteration, we prepare a list of candidate edges $\{m \rightarrow n\}$, such that (a) $m \in \mathcal{N}(\mathbf{CS}^0)$, (b) $n \in \mathcal{N}(\mathbf{CNS}^0)$ and (c) $(m \rightarrow n) \in \mathcal{E}$. The notation $\mathcal{N}(\mathbf{CS}^0)$ denotes the set of nodes in $\mathbf{CS}^0$. Next, we choose an edge $(i_1 \rightarrow j_1)$. The criterion based on which the edge is chosen will be discussed subsequently (see Fig. 4). Suppose $j_1 \in \mathcal{C}_b$. Cluster $\mathcal{C}_b$ is then labeled *spanned* and nodes i_1 and j_1 are marked *shaded*. Note that a cluster which has at least one shaded node must be spanned, and conversely, a cluster with no shaded node must not be spanned.

Next, we grow the largest forests rooted at nodes i_1 and j_1, $F(i_1, \mathcal{C}_a)$ and $F(j_1, \mathcal{C}_b)$, and mark the nodes in the largest forests as *shaded*. Note that these forests are local; $i.e.$, they are composed of only intra-cluster edges. A greedy algorithm, such as the one discussed in Section 12.3 of [15], may be used to compute the largest forests, with the adjacency matrix of the nodes within the cluster as the weight function. Now define

$$\mathbf{CS}^1 = \mathbf{CS}^0 \cup \mathcal{C}_b \text{ and } \mathbf{CNS}^1 = \mathbf{C} \setminus \mathbf{CS}^1$$

3. If all clusters have been spanned, we stop. Otherwise, we prepare a set of candidate edges $\{m \rightarrow n\}$, such that (a) $m \in \mathcal{N}(\mathbf{CS}^1)$, (b) $n \in \mathcal{N}(\mathbf{CNS}^1)$, (c) $(m \rightarrow n) \in \mathcal{E}$ and (d) node m is *shaded*. Note that, because of the last condition, only those directed edges whose tail nodes belong to the largest rooted forests in the spanned clusters are eligible for consideration as a candidate edge. Another implication of condition (d) is that it will preclude solutions of the type shown in Fig. 1(c), where two nodes located in the same cluster, with no direct path between them, are connected by one or more inter-cluster edges. This wont be a problem, however, if any of the following conditions are satisfied:

- the problem instance is such that a local spanning tree exists for all clusters, $i.e.$, for any cluster $\mathcal{C}_a$, it is possible to reach all other nodes in $\mathcal{C}_a$ from any node $i \in \mathcal{C}_a$, and, the weights of all intra-cluster edges are smaller than the weight of any inter-cluster edge,

  or

- the weight matrix is replaced by a shortest path matrix, $i.e.$, all $\mathbf{W}_{ij}$'s are transformed as:

  $$\mathbf{W}_{ij} \leftarrow min(cost(SP(i,j)), \mathbf{W}_{ij}) : \forall (i,j)$$

  where $cost(SP(i,j))$ is the cost of the shortest path between nodes i and j. The shortest path matrix can be calculated in $O(N^3)$ time using the Floyd-Warshall algorithm (see Section 5.6 of [1]). With the above transformation, the total number of edges in the underlying graph is effectively equal to $N(N-1)/2$, which may be significantly higher than the original number of edges E.

On the positive side, imposing condition (d) ensures that a valid solution can be found after $C-1$ inter-cluster edges have been chosen during the construction phase.

From a modeling aspect, condition (d) implies an added constraint on the ILP model for the L-GMST; that, if more than one node is chosen from a cluster, they must be connected in the GMST by intra-cluster edges.

If the set of candidate edges is empty but not all clusters have been spanned, we set the cost of the partial solution to ∞ and break out of the iterative process. Otherwise, we choose an edge, say $(i_2 \rightarrow j_2)$, $j_2 \in \mathcal{C}_c$, label $\mathcal{C}_c$ as *spanned*, grow the largest forest in $\mathcal{C}_c$ rooted at node j_2, $F(j_2, \mathcal{C}_c)$, and mark the nodes in $F(j_2, \mathcal{C}_c)$ as *shaded*. Note that the cluster to which node i_2 belongs must be spanned and therefore a rooted forest already exists for that cluster. Finally, we define:

$$\mathbf{CS}^2 = \mathbf{CS}^1 \cup \mathcal{C}_c \text{ and } \mathbf{CNS}^2 = \mathbf{C} \setminus \mathbf{CS}^2$$

4. The above step is repeated till all clusters have been labeled *spanned*.

5. After the tree construction algorithm terminates successfully, $i.e.$, after all clusters have been spanned in, all redundant edges which are not required to complete the GMST are pruned from the local forests grown during the construction phase. The remaining set of intra-cluster edges is appended to the set of inter-cluster edges chosen during the construction phase to complete the GMST. We now discuss the criterion by which an ant chooses an edge at any iteration of the tree-building process. Edge selection is either random or deterministic and based on selection probabilities of the constituent edges in the candidate list. The extent to which probabilistic decisions are made is controlled by a possibly time varying tunable parameter, $q_0(t)$. In our implementation, we gradually reduce $q_0(t)$ over time so that decision making is predominantly probabilistic during the initial stages of the algorithm and mostly deterministic during the latter stages. Edge selection is also dependent on the behavioral characteristics of the ant population. These characteristics, which are controlled by tunable parameters, are stochastically adapted to allow for better exploration of the search space during the initial stages of the algorithm.

The factors which determine the desirability of choosing an edge $(i \rightarrow j)$ at iteration k and time t are:

- local visibility of node j from i, η_{ij}, which is inversely proportional to the weight of the edge $(i \leftrightarrow j)$, $\mathbf{W}_{ij}$. Greater the local visibility, higher is the desirability of choosing that edge. In our algorithm, we use two types of ants, Type-A and Type-B, with different behavioral characteristics. This difference is induced by unequal exponential scaling of the local visibilities for the two types of ants. Specifically, let $\beta_A(t)$, $0 < \beta_A(t) \leq 1$, be the exponential scaling parameter for Type A ants and $\beta_B(t)$, $0 < \beta_A(t) \leq \beta_B(t) \leq 1$, the scaling parameter for type B ants. The desirability of choosing an edge $(i \leftrightarrow j)$ is proportional to $[\eta_{ij}]^{\beta_A(t)}$ for Type A ants and $[\eta_{ij}]^{\beta_B(t)}$ for Type B ants. If $\beta_A < \beta_B \leq 1$, Type A ants operate on reasonably smoothed out edge weights compared to Type B ants, and therefore distant nodes appear almost as attractive as nearby nodes. For an arbitrary 3-node network, suppose we have one Type A and one Type B ant at node 1. Assume that $\eta_{12} = 0.25$

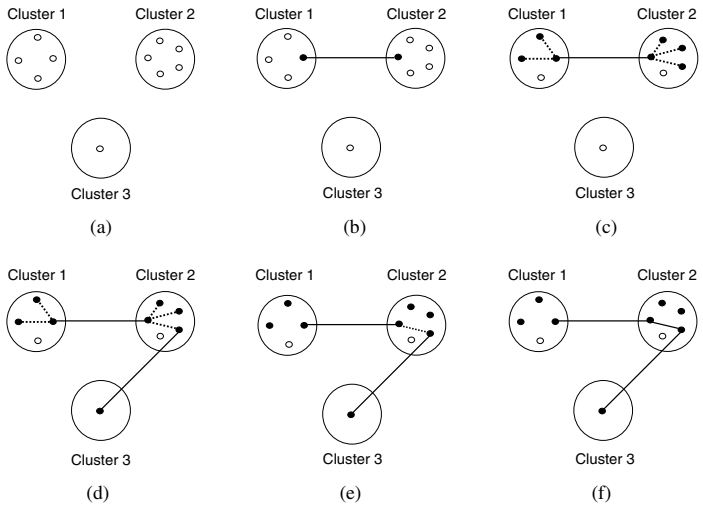

Figure 3. Illustrating tree building by an ant. (a) Distribution of nodes in the 3 clusters. Initially, all nodes are labeled *unshaded* and all clusters are labeled *not spanned*. (b) For the first iteration, an initial cluster is chosen, either randomly, or, probabilistically according to the cluster node densities, $\alpha(\mathcal{C}_c)$'s. Let the initial cluster be 1. Suppose an inter-cluster edge is chosen between clusters 1 and 2, as shown. In general, suppose that the initial edge chosen is $(i_1 \rightarrow j_1)$, $i_1 \in \mathcal{N}(\mathcal{C}_a)$ and $j_1 \in \mathcal{N}(\mathcal{C}_b)$. Mark nodes i_1 and j_1 as *shaded*, as shown in the figure. Also, mark clusters $\mathcal{C}_a$ and $\mathcal{C}_b$ as *spanned*. Note that a cluster which has at least one shaded node must be spanned, and conversely, a cluster with no shaded node must not be spanned. (c) For the first iteration only, grow the largest forests rooted at nodes i_1 and j_1, $F(i_1, \mathcal{C}_a)$ and $F(j_1, \mathcal{C}_b)$. Mark the nodes in the largest forests as *shaded*. In this figure, we show the largest rooted forests for clusters 1 and 2 and the set of all *shaded* nodes. Edges which are part of the largest rooted forests are shown dotted. (d) For all subsequent iterations, prepare a candidate edge list such that tail nodes of the edges in the list are *shaded* and the head nodes are located in clusters labeled *not spanned*. Choose an edge from the candidate edge list, say $(i_2 \rightarrow j_2)$. Suppose $j_2 \in \mathcal{N}(\mathcal{C}_c)$. Mark node j_2 as *shaded* and label cluster $\mathcal{C}_c$ as *spanned*. Note that the cluster to which node i_2 belongs is already *spanned*. If all clusters are *spanned*, the tree building algorithm terminates, as is the case in our example. Otherwise, we repeat steps (c) and (d), with the exception that, in step (c), we now grow only the largest forest rooted at j_2. (e) After the tree building algorithm terminates, we may have redundant intra-cluster edges, which are pruned. In the figure, we have pruned the redundant edges in clusters 1 and 2. (f) The remaining intra-cluster edges are appended to the set of inter-cluster edges chosen previously to complete the solution.

and $\eta_{13} = 0.125$. Choosing $\beta_A = 0.1$ and $\beta_B = 1$, the scaled local visibilities for the two types of ants are:

- Type A: $[\eta_{12}]^{\beta_A} = 0.87$, $[\eta_{13}]^{\beta_A} = 0.81$
- Type B: $[\eta_{12}]^{\beta_A} = 0.25$, $[\eta_{13}]^{\beta_A} = 0.125$

If edges are chosen probabilistically based on scaled local visibilities, the Type B ant is twice as likely to choose edge $(1 \rightarrow 2)$ over $(1 \rightarrow 3)$. The Type A ant, however, is almost equally likely to choose any of the two edges. Type A ants are therefore less greedy and better suited for random exploration of the search space.

During the initial time instants, we use an almost even mix of Type A and Type B ants. Over time, the population of Type A ants is gradually converted to Type B. This limits the exploratory regime and guides the algorithm towards convergence.

- pheromone level on the edge at time t. Since edges which are part of better solutions are positively reinforced, presence of a high pheromone level on an edge is used to boost the desirability of choosing that edge. At time t, the pheromone level on the edge $(i \rightarrow j)$ reflects the cumulative knowledge acquired by the ants till time $t-1$ on the desirability of moving to node j from node i. A very high pheromone level on any edge, therefore, makes it much more probable for that edge to be included in the final tree.

The exact formulae for computing the edge selection probabilities is shown in Fig. 4 which summarizes the edge selection mechanism discussed above. From equation (4), it is apparent that the initial stages of the algorithm (small t), when the pheromone levels on the edges are almost uniform, are conducive to the exploratory behavior of Type A ants. With gradual pheromone build-up over time, the selection probabilities (the a_{ij}'s) are dominated by the pheromone levels on the edges. Coupled with the fact that decision making is mostly deterministic as time increases (see Step 3 in Fig. 4), it is apparent that the benefit of using Type A ants reduces over time. This provides an intuitive justification for gradually converting the Type A ants to Type B as t increases.

A complete pseudocode of the tree building algorithm is shown in Fig. 5, with an optional local tree improvement step and a "shortest path replacement and edge elimination" (SPREE) step for non-Euclidean problem instances whose weight matrices are transformed as shown in (2). These are discussed below.

2.3. Non-Euclidean weight matrices: Shortest path replacement and edge elimination (SPREE)

We mentioned at the beginning of this section that the weight matrix for non-Euclidean problem instances is transformed according to (2) before the ACS algorithm is applied. Though this transformation is helpful, it is only a partial

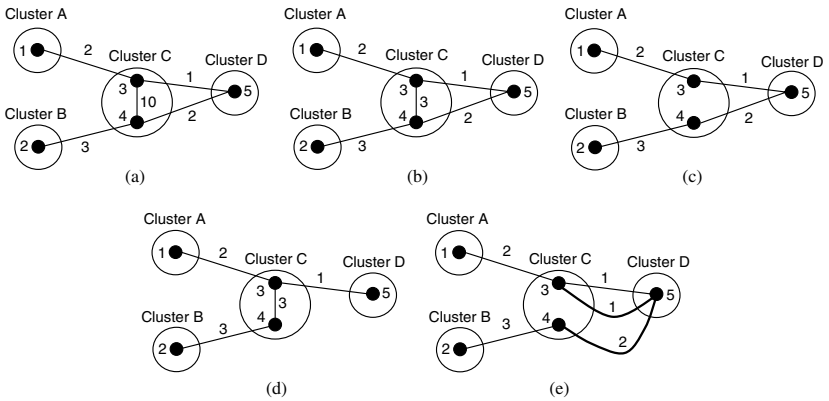

Figure 6. Illustrating the need for shortest path replacement and edge elimination for non-Euclidean weight matrices. (a) Input graph. The numbers above the edges are the weights. Note that the weights of the edges between nodes 3, 4 and 5 do not satisfy the triangle inequality. (b) Transformation of $\mathbf{W}_{34}$ according to (2). The shortest path between nodes 3 and 4 is $[3 \leftrightarrow 5; 5 \leftrightarrow 4]$ and its cost is 3. (c) Optimal solution for graph (a). Cost = 8. (d) Solution obtained using the ACS algorithm. Cost = 9. (e) Since the weight of the edge $3 \leftrightarrow 4$ was modified in (b), we replace it with the shortest path $[3 \leftrightarrow 5; 5 \leftrightarrow 4]$. The edges in the shortest path replacement are shown in bold. Doing so results in two edges between nodes 3 and 5, one of which can be eliminated to obtain the optimal solution in (c).

remedy, as illustrated in Fig. 6. Given a solution, it may be possible to improve it by replacing the edges which underwent a weight transformation ($i.e.$, the edges for which $\mathbf{W}_{ij} > cost(SP(i,j))$ in the input graph) with their shortest paths and eliminating any resulting multi-edges[1].

2.4. A local tree improvement heuristic

As indicated in Fig. 5, it is possible to embed a local tree improvement heuristic within the tree building algorithm. Alternately, the heuristic can be applied only on the best solution at time t, $T_{cbest}(t)$. In our implementation, we have incorporated a simple branch exchange heuristic for tree improvement. This is done for all ants, before their costs are evaluated, as shown in step 6 of Fig. 5. A brief description of the branch exchange algorithm follows.

Given an initial tree, the algorithm temporarily removes an edge (say e_c) from the tree and checks whether the two disconnected components (*e.g.* T_1 and T_2) created by the edge removal operation can be reconnected using a cheaper replacement edge (say e_r) whose end nodes are in T_1 and T_2. If so, the edge e_c is replaced by e_r and the algorithm is applied on the modified tree. This procedure is repeated till no further improvement is possible.

2.5. Edge reinforcement mechanism

In this section, we describe our edge reinforcement strategy. We adopt an *elitist* approach, or, in other words, only the best ant at each time instant is allowed to positively reinforce the edges constituting its solution. Let us first define the following parameters:

- $T_{gbest}(t)$ = global best tree *till* time t.
- $J_{gbest}(t)$ = cost of $T_{gbest}(t)$.
- $T_{cbest}(t)$ = best tree *at* current time t.
- $J_{cbest}(t)$ = cost of $T_{cbest}(t)$.

[1] A *multi-edge* is a collection of two or more edges having identical end nodes [16].

For $t = 1$, the current best ant is designated the global best ant as shown below:

$$T_{gbest}(1) = T_{cbest}(1) \text{ and } J_{gbest}(1) = J_{cbest}(1) \quad (6)$$

For $t > 1$,

$$T_{gbest}(t) = \begin{cases} T_{cbest}(t) & \text{if } J_{cbest}(t) < J_{gbest}(t-1) \\ T_{gbest}(t-1) & \text{otherwise} \end{cases} \quad (7)$$

$$J_{gbest}(t) = \begin{cases} J_{cbest}(t) & \text{if } J_{cbest}(t) < J_{gbest}(t-1) \\ J_{gbest}(t-1) & \text{otherwise} \end{cases} \quad (8)$$

The reinforcement mechanism is straightforward, as shown in Fig. 7. Figure 7 also shows an optional pheromone thresholding step. Implementing this step might prevent premature stagnation of the search process for certain problem instances. However, we have not noticed any evidence of stagnation in the simulations we have conducted so far.

All edges which are part of the global best solution till time t, $T_{gbest}(t)$, receive a positive reinforcement which is inversely proportional to its cost, $J_{gbest}(t)$, and a negative reinforcement (evaporation) which is controlled by the parameter ρ (9). The smaller the cost of $T_{gbest}(t)$, the higher the positive reinforcement. The exact pheromone delivery amount is regulated by the parameter γ.

If the edge is not part of $T_{gbest}(t)$, but is part of the current best solution at time t, $T_{cbest}(t)$, it receives a positive reinforcement which is inversely proportional to its cost $J_{gbest}(t)$, weighted by the ratio $J_{gbest}(t)/J_{cbest}(t)$, and a negative reinforcement controlled by the parameter ρ (10). As in the previous paragraph, the exact pheromone delivery amount is regulated by γ. Note that, if $T_{cbest}(t)$ is almost as good as $T_{gbest}(t)$, the ratio $J_{gbest}(t)/J_{cbest}(t) \simeq 1$ and therefore $T_{cbest}(t)$ receives almost the maximum allowable reinforcement amount, $\gamma/J_{cbest}(t) \simeq \gamma/J_{gbest}(t)$. Conversely, if $T_{cbest}(t)$ is significantly worse than $T_{gbest}(t)$, its edges receive almost negligible reinforcement (attenuated as $J_{gbest}(t)/J^2_{cbest}(t)$).

All other edges receive only a negative reinforcement (11). Fig. 8 provides a high level description of the overall algorithm.

/* *Let* $edge\_list^k$ *denote the candidate edge list at iteration* k. */

1. Let $\mathbf{A}^k = \{a_{ij} : (i \rightarrow j) \in edge\_list^k\}$ be the decision matrix based on which an ant makes its decision for selecting an edge at iteration step k. The elements $\{a_{ij}\}$ are computed as follows:

$$a_{ij} = \begin{cases} \frac{\tau_{ij}(t) f_A(\eta_{ij})}{\sum_x \tau_{ix}(t) f_A(\eta_{ix})}, & \text{for Type A ants} \\ \frac{\tau_{ij}(t) f_B(\eta_{ij})}{\sum_x \tau_{ix}(t) f_B(\eta_{ix})}, & \text{for Type B ants} \end{cases} \quad (4)$$

where $\tau_{ij}(t)$ is the pheromone level on edge $(i \rightarrow j)$ at time t, $f_A(\eta_{ij}) = [\eta_{ij}]^{\beta_A(t)}$ is a function of the local visibility of node j from node i for Type A ants and $f_B(\eta_{ij}) = [\eta_{ij}]^{\beta_B(t)}$ is a function of the local visibility for Type B ants.

2. Sample q from a uniform distribution over [0,1].

/* $0 < q_0(t) \leq 1$ *is a time dependent threshold parameter. As* t *increases,* $q_0(t)$ *is driven towards 1.* */

3. **if**$(q < q_0(t))$ /* *deterministic decision making* */
   Choose the strongest edge, $(u \rightarrow v)$, from $\mathbf{A}^k$.

$$(u, v) = \text{argmax}_{i,j}\{a_{ij}\} \quad (5)$$

   else /* *explore, probabilistic decision making.* */
   Choose the edge $(u \rightarrow v)$ from $\mathbf{A}^k$ probabilistically, *e.g.*, using a roulette-wheel mechanism.
   end if

4. Let $(i_k \rightarrow j_k)$ denote the edge chosen at iteration k. Assign $(i_k \rightarrow j_k) := (u \rightarrow v)$ /* *this extra notation is introduced to maintain consistency with our previous discussion of the tree building algorithm* */

Figure 4. *Pseudo-random-proportional* edge selection criterion at any iteration k of the tree building process.

3. SIMULATION RESULTS

We have tested our algorithm on some of the problem instances used by Dror *et al* in [6] and Feremans in [9][2]. The edge weight matrices for the problem instances are symmetric, non-Euclidean and each element is an integer drawn from an uniform distribution between 1 and 50. See [6] for a discussion on how the datasets were generated. Since the elements of the input weight matrix do not satisfy the triangle inequality, we first transformed it according to (2).

Table 3 shows the values of the parameters used in the simulations. A key point to note in Table 3 is the dynamic nature of the parameters $q_0(t)$, $\beta_A(t)$ and the number of Type A and B ants. Gradually reducing $q_0(t)$ ensures that the bulk of the exploration work (step 3 in Fig. 4) is carried out during the initial stages of the algorithm, when the pheromone distribution on the edges is not too uneven and "trail conditions" are more suitable for Type A ants. Increasing $\beta_A(t)$ has the effect of reducing the local visibility of Type A ants so that they start behaving more like their Type B counterparts as t increases. In fact, for $\lfloor 0.6 * t_{MAX} \rfloor + 1 \leq t \leq t_{MAX}$, $\beta_A(t) = \beta_B(t)$, which ensures that all ants concentrate on a select group of edges and look for better solutions within their neighborhoods, during the final stage of the algorithm. The pheromone thresholding option in Figure 7 was disabled, but the local search and SPREE components in Figure 5 were enabled.

[2]The datasets were obtained through personal communication with Dr. Corinne Feremans.

Parameter	*Parameter value*
t_{MAX}	100/200
ρ	0.2
γ	0.2
$\tau_{ij}(t=0)$	0.001/0.0005
$\beta_B(t)$	1
$\beta_A(t)$	0.1, if $t \leq \lfloor 0.3 * t_{MAX} \rfloor$ 0.5, if $\lfloor 0.3 * t_{MAX} \rfloor + 1 \leq t \leq \lfloor 0.6 * t_{MAX} \rfloor$ 1, if $\lfloor 0.6 * t_{MAX} \rfloor + 1 \leq t \leq t_{MAX}$
$q_0(t)$	0.3, if $t \leq \lfloor 0.3 * t_{MAX} \rfloor$ 0.6, if $\lfloor 0.3 * t_{MAX} \rfloor + 1 \leq t \leq \lfloor 0.6 * t_{MAX} \rfloor$ 0.9, if $\lfloor 0.6 * t_{MAX} \rfloor + 1 \leq t \leq t_{MAX}$
$N_{ants(A)}(t)$	2, if $t \leq \lfloor 0.3 * t_{MAX} \rfloor$ 1, if $\lfloor 0.3 * t_{MAX} \rfloor + 1 \leq t \leq \lfloor 0.6 * t_{MAX} \rfloor$ 0, if $\lfloor 0.6 * t_{MAX} \rfloor + 1 \leq t \leq t_{MAX}$
$N_{ants(B)}(t)$	3, if $t \leq \lfloor 0.3 * t_{MAX} \rfloor$ 4, if $\lfloor 0.3 * t_{MAX} \rfloor + 1 \leq t \leq \lfloor 0.6 * t_{MAX} \rfloor$ 5, if $\lfloor 0.6 * t_{MAX} \rfloor + 1 \leq t \leq t_{MAX}$

Table 1. Parameter values used for the simulations. t_{MAX} was set to 100 for problem instances 1 to 6 (Group A) and 200 for problem instances 7 to 10 (Group B). The initial pheromone level, $\tau_{ij}(0)$ was set to 0.001 for Group A and 0.0005 for Group B.

We ran the ACS algorithm three times for every problem instance. The results, along with the optimal costs and those reported by Dror *et al* in [6] are shown in Table 3. While Dror *et al* provide results for four heuristics and a genetic algorithm (GA), we cite only the GA results here since they produce the best solutions. For each test instance, we used a population of 5 ants. The maximum running time of the algorithm, t_{MAX}, was set to 100 for problem instances 1 to 6 (which we refer as Group A) and to 200 for problem instances 7 to 10 (referred to as Group B)[3]. The ACS costs reported here therefore represent the best of a maximum of 500 solutions for Group A and 1000 solutions for Group B. In comparison, the results reported in [6] represent the best of a maximum of 2000 solutions (population size = 100 and number of generations = 20).

From Table 3, it can be seen that:

- for problem instances 1 to 5 and 7, both the ACS algorithm and Dror *et al*'s GA algorithm are able to find the optimal solutions.
- for problem instance 6, the ACS algorithm is able to find the optimal solution. The reported GA solution is about 110% of the optimal cost.
- for problem instances 8 and 10, neither algorithm finds the optimal solution. However, in both cases, the ACS algorithm is able to improve upon the GA solution.
- for problem instance 9, the GA algorithm finds the optimal solution but the ACS algorithm does not. The best ACS solution is approximately 123% of the optimal cost.

We are currently in the process of generating other (both

[3]It appears from [6] that the authors ran their GA algorithm 4000 times for each problem instance.

Problem Instance	N	E	C	*Optimal*	*DHC-GA* [6]	*ACS - Run 1*	*ACS - Run 2*	*ACS - Run 3*
gmst 1	25	50	4	23	23	23	27	23
gmst 2	25	100	8	41	41	41	41	41
gmst 3	25	150	10	36	36	37	37	36
gmst 4	50	150	5	18	18	22	20	18
gmst 5	50	300	10	27	27	27	27	27
gmst 6	75	200	8	55	60	66	55	64
gmst 7	75	300	10	67	67	70	67	77
gmst 8	75	400	15	53	60	57	62	55
gmst 9	100	300	7	35	35	43	45	43
gmst 10	100	500	10	48	50	51	49	50

Table 2. Computational results. The first column lists the problem instance numbers (same as those used in [6] and [9]); the second, third and fourth columns list the number of nodes (N), number of undirected edges (E) and number of clusters (C); the fifth column lists the optimal costs; the sixth column lists the GA solution costs reported by Dror, Haouari and Chaouachi in [6] and the last 3 columns list the best solution costs obtained by running our ACS algorithm 3 times on each problem instance.

Euclidean and non-Euclidean) test instances with the following characteristics[4]:

- a spanning tree exists for each cluster ($i.e.$, the graph $(\mathcal{N}(\mathcal{C}_c), \mathcal{E}(\mathcal{C}_c))$ is connected for all c).
- the weights of the intra-cluster edges are smaller than the weight of any inter-cluster edge.

Computational results for these datasets and a second batch of 10 test instances (gmst 11 to gmst 20) from [6] will be reported in an upcoming journal version of the paper.

4. CONCLUSION

In this paper, we considered the "at least" version of the Generalized Minimum Spanning tree problem (L-GMST), which is known to be NP-hard. We proposed an ant colony system based solution approach for the L-GMST. A key feature of our algorithm is its use of ants of different behavioral characteristics, which are adapted over time. Computational results on datasets used in earlier literature indicate that our algorithm provides similar or better results for most of them.

Acknowledgement

The authors wish to thank Dr. Corinne Feremans for providing them with the test instances used in this paper and their optimal costs.

5. REFERENCES

[1] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, "*Network flows: theory, algorithms, and applications*, Prentice-Hall, 1993.

[2] Y.S. Myung, C.H. Lee, and D.W. Tcha, "On the generalized minimum spanning tree problem," *Networks*, vol. 26, pp. 231-241, 1995.

[3] E. Ihler, G. Reich, P. Widmayer, "Class Steiner trees and VLSI-design," *Discrete Applied Mathematics*, vol. 90, pp. 173-194, 1999.

[4]These are motivated by typical topological characteristics of large scale clustered sensor networks.

[4] U. Faigle, W. Kern, P. Pop and G. Still, "The Generalized minimum spanning tree problem," Working Paper, Dept. of Operations Research & Mathematical Programming, University of Twente, 2000.

[5] M. Dror, M. Haouari and J. Chaouachi, "Generalized Steiner problems and other variants," *J. Combinatorial Optimization*, vol. 4, pp. 415-436, 2000.

[6] M. Dror and M. Haouari, "Generalized spanning trees," *European J. Operational Research*, vol. 120, pp. 583-592, 2000.

[7] P. Pop, W. Kern, and G. Still, "An approximation algorithm for the generalized minimum spanning tree problem with bounded cluster size," Working Paper, Dept. of Operations Research and Mathematical Programming, University of Twente, 2001.

[8] C. Feremans, M. Labbé, and G. Laporte, "On generalized minimum spanning trees," *European J. Operational Research*, vol. 134, 2001, 457-458.

[9] C. Feremans, *Generalized spanning trees & extensions*, Ph.D thesis, Universite Libré de Bruxelles, 2001.

[10] P.C. Pop, *The generalized minimum spanning tree problem*, Ph.D thesis, University of Twente, 2002.

[11] C. Feremans, M. Labbé, and G. Laporte, "A comparative analysis of several formulations for the generalized minimum spanning tree problem", *Networks*, vol. 39, pp. 29-34, 2002.

[12] C. Feremans, M. Labbé, and G. Laporte, "Generalized network design problems," March 2002. *http://citeseer.ist.psu.edu/feremans02generalized.html*

[13] M. Dorigo and L.M. Gambardella , "Ant colonies for the traveling salesman problem," *BioSystems*, vol. 43, pp. 73-81, 1997.

[14] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm intelligence: from natural to artificial systems*, Oxford University Press, 1999.

[15] C.H. Papadimitriou and K. Steiglitz, "*Combinatorial optimization: algorithms and complexity*", Dover Publications, 1998.

[16] J. Gross and J. Yellen, *Graph theory and its applications*," CRC Press, 1999.

1. /* *Initializations* */
• $k = 0;\ tree = \emptyset$;
• $node\_labels[n] = 0; 1 \leq n \leq N$
• $cluster\_labels[c] = 0; 1 \leq c \leq C$
• $rooted\_forests[c] = \emptyset; 1 \leq c \leq C$
• Choose an initial cluster, either randomly or probabilistically based on the cluster node densities. Suppose the initial cluster is $\mathcal{C}_a$.
• Assign: $\mathbf{CS} := [\mathcal{C}_a];\ \mathbf{CNS} := \mathcal{C} \setminus \mathbf{CS}$;
• Increment: $k = k + 1$;
2. /* *The tree building algorithm* */
for ($k = 1$ to $C - 1$)
if ($k == 1$)
• Find the candidate edge list $edge\_list^k = [\{m \to n\}]$ such that $m \in \mathcal{N}(\mathbf{CS}), n \in \mathcal{N}(\mathbf{CNS})$ and $(m \to n) \in \mathcal{E}$, where $\mathcal{N}(\mathbf{CS})$ denotes the set of nodes in $\mathbf{CS}$.
• Choose an edge $(i_k \to j_k)$ according to Figure 4. Suppose $j_k \in \mathcal{C}_b$.
• /* *Append the edge to the existing tree* */ $tree = [tree;\ (i_k \to j_k)]$;
• Grow the local forests $F(i_k, \mathcal{C}_a)$ and $F(j_k, \mathcal{C}_b)$. The forest growing algorithm should return i_k (j_k) if i_k (j_k) is isolated in $\mathcal{C}_a$ ($\mathcal{C}_b$).
• /* *Updates. Note that* $i_k \in \mathcal{C}_a$. */
$cluster\_labels[c] = 1;\ c = \mathcal{C}_a, \mathcal{C}_b$
$rooted\_forests[\mathcal{C}_a] \leftarrow F(i_k, \mathcal{C}_a)$;
$cluster\_labels[\mathcal{C}_b] \leftarrow F(j_k, \mathcal{C}_b)$;
$node\_labels[n] = 1;\ \forall n \in F(i_k, \mathcal{C}_a)$ and $F(j_k, \mathcal{C}_b)$
$\mathbf{CS} \leftarrow \mathbf{CS} \cup \mathcal{C}_b$ and $\mathbf{CNS} \leftarrow \mathcal{C} \setminus \mathbf{CS}$;
else
• Find the candidate edge list $edge\_list^k = [\{m \to n\}]$ such that $m \in \mathcal{N}(\mathbf{CS}), n \in \mathcal{N}(\mathbf{CNS})$, $(m \to n) \in \mathcal{E}$ and $node\_labels[m] = 1$.
• Choose an edge $(i_k \to j_k)$ according to Figure 4. Suppose $j_k \in \mathcal{C}_c$.
• /* *Append the edge to the existing tree* */ $tree = [tree;\ (i_k \to j_k)]$;
• Grow the local forest $F(j_k, \mathcal{C}_c)$. The forest growing algorithm should return j_k if it is isolated in $\mathcal{C}_c$.
• /* *Updates* */
$cluster\_labels[c] = 1;\ c = \mathcal{C}_c$
$rooted\_forests[\mathcal{C}_c] \leftarrow F(j_k, \mathcal{C}_c)$;
$node\_labels[n] = 1;\ \forall n \in F(j_k, \mathcal{C}_c)$
$\mathbf{CS} \leftarrow \mathbf{CS} \cup \mathcal{C}_c$ and $\mathbf{CNS} \leftarrow \mathcal{C} \setminus \mathbf{CS}$;
end if
end for
3. Prune the redundant edges from $rooted\_forests[c], \forall c$. Let $intra\_edges$ denote the set of all remaining edges.
4. Assign: $T_m(t) \leftarrow [tree;\ intra\_edges]$;
/* *Step 5 is used if the weight matrix is non-Euclidean and transformation (2) is applied.* */
5. Apply procedure SPREE (Section 2.3) on $T_m(t)$;
6. Apply local tree improvement on $T_m(t)$. /* *Optional* */
7. Compute the cost of $T_m(t)$ and assign it to the variable $Y_m(t)$.

Figure 5. A pseudocode of the complete tree building algorithm. Recall that $T_m(t)$ denotes the tree built by ant m at time t. Though not shown, the above algorithm should terminate prematurely if the list of candidate edges is empty at any iteration and not all clusters have been spanned. In that case, the cost $Y_m(t)$ is set to ∞.

/* *ρ and γ are parameters which regulate pheromone evaporation and delivery.* */

for all $(i, j) \in \mathcal{E}$,

if $(i, j) \in T_{gbest}(t)$

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \left(\frac{\gamma}{J_{gbest}(t)}\right) \tag{9}$$

elseif $(i, j) \in T_{cbest}(t)$

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \left(\frac{J_{gbest}(t)}{J_{cbest}(t)}\right)\left(\frac{\gamma}{J_{cbest}(t)}\right) \tag{10}$$

else

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) \tag{11}$$

endif

/* *Optional pheromone thresholding step. τ_{MAX} and τ_{MIN} are user defined upper and lower threshold levels.* */

if $(\tau_{ij}(t+1) > \tau_{MAX})$

$$\tau_{ij}(t+1) = \tau_{MAX}; \tag{12}$$

elseif $(\tau_{ij}(t+1) < \tau_{MIN})$

$$\tau_{ij}(t+1) = \tau_{MIN}; \tag{13}$$

endif

Figure 7. Edge reinforcement rules, with optional pheromone thresholding.

1. Set $t = 0$.
2. Initialize the pheromone levels on the edges, $\tau_{ij}(0)$.
3. Increment $t = t + 1$.
4. Set the time dependent parameters of the ACS algorithm (see Section 2.1).
5. Build a tree for each ant iteratively (see Section 5).
6. Find the local best ant and the global best ant and their costs (see Section 2.5).
7. If $t > t_{MAX}$, stop. Otherwise, increment $t = t + 1$ and repeat steps 4 to 6.

Figure 8. A high level description of the overall ACS algorithm.