Heuristics for Minimum Spanning k-Trees[*]

Alfredo Candia-Véjar[1] *Héctor Beck-Fernández*[2]

[1]Department of Systems Engineering,
University of Talca, Km.1 Los Niches,
Campus Curicó-Chile
acandia@pehuenche.utalca.cl

[2]Department of Computer Science,
University of Tarapacá, Casilla 6-D,
Arica-Chile
hbeck@huemul.decom.uta.cl

Abstract
We present a class of greedy heuristics for the Minimum Spanning k-Tree Problem, a NP-Hard combinatorial optimization problem. This problem is a generalization of the Minimum Spanning Tree Problem, a very well known problem in graphs, for which there are a number of polynomial algorithms, among them is Prim's algorithm. Our heuristics are based on a generalization of this algorithm. Computational experiences tested on random graphs show the performance of three versions of the heuristic when it is applied to minimum spanning $2-trees$ *instances.*

Resumo
Apresentamos uma clase de heurísticas greedy para o problema da $k-$*árvore geradora de peso mínimo, um problema de otimização combinatorial NP-árduo. Este problema é uma generalização do problema da árvore geradora de peso mínimo, um problema en grafos muito conhecido, para o qual vários algoritmos polinomiais tem sido disenhados, entre eles o algoritmo de Prim. Nossas heurísticas estão baseadas numa generalização desse algoritmo. Testes computacionais sobre grafos aleatorios ilustram o desempenho das heurísticas quando são aplicadas à instancias do problema da 2-árvore geradora de peso mínimo.*

Keywords: k-trees; greedy heuristics; invulnerable networks.

1 Introduction

We present a class of greedy heuristics for the *Minimum Spanning k-Tree Problem* (MSkT), a NP-Hard combinatorial optimization problem. MSkT generalizes a

[*]This work was financed by contract No 94 5B 022 of CNET.

classical problem in graphs, *Minimum Spanning Tree Problem* (MST), which has a number of applications. A lot of polynomial algorithms have been designed for MST, three of them are greedy algorithms: Prim's, Kruskal's, and Sollin's algorithms. Our greedy heuristics for MSkT are based on a generalization of Prim's algorithm.

An interesting application of $2-trees$ in optimal network design was developed by Farley [10]. He proved that $2-trees$ are *Minimal Isolated Failures Immune Networks* (IFI networks), that is, networks that cannot be disconnected by removing a set of vertices and edges in which no two vertices are adjacent, no two edges are adjacent, and no vertex is less than a distance of two edges from an edge. IFI networks remain connected even in the presence of a large number of failures. Wald and Colbourn [18] established the equivalence between IFI networks and $2-trees$. Farley [10] suggested to study the problem of finding minimum-length spanning $2-trees$ for a set of vertices, that is, to find IFI networks of minimum cost.

Section 2 gives definitions and properties of $k-trees$, and discusses some algorithmic problems concerning to $k-trees$, In Section 3 are presented known results about the complexity of minimum spanning $k-trees$. Section 4 presents the heuristic algorithm proposed for solving the minimum spanning $k-tree$ problem along with its complexity analysis; also includes an analysis of some variants of the heuristic. Section 5 presents numerical results obtained from the application of the heuristic to random instances of the minimum spanning $2-tree$ problem. Finally, in Section 6, the conclusions of the paper are presented.

2 k-Trees and some properties

In this section we give definitions and basic properties of $k-trees$ and mention some problems related to $k-trees$.

A $k-clique$ is a complete graph with k vertices and it will be denoted by K_n.

Definition 1. A $k-tree$ is a member of a class of undirected graphs defined recursively as follows:

- A $k-clique$ is a $k-tree$.
- If T is a $k-tree$ with $(n-1)$ vertices, then a new $k-tree$ with n vertices is formed by creating a new vertex v and adding edges between v and every vertex of an existing $k-clique$ in T.

It is easy to note that the set of $1-trees$ is the set of *trees*, and that a $k-tree$ with n vertices has exactly $k(k-1)/2+k(n-k)$ edges. The $k-tree$ in the first part of the definition is called the *starting clique.* Each $k-tree$ may be constructed using several different sequences of the operations described in the two parts of definition 1.

Rose [17] in an early work noted that $k-trees$ are perfect elimination graphs and gave simple characterizations of $k-trees$.

Here we present a summary of well-known properties of $k-trees$.

A simple cycle $[v_0, v_1, v_2, ..., v_l, v_0]$ is *chordless* if $(v_i, v_j) \notin E$ for i and j differing by more than $1 mod(l+1)$.

Theorem 1. Let k be a positive integer and T be a $k-tree$ with n vertices, then:

- T contains no chordless cycle of length at least 4;
- T contains no *clique* of size $k+2$;
- T has exactly $(n-k)$ *cliques* of size $k+1$;
- either T is a $k-clique$ or every maximal *clique* of T is of size $k+1$;
- the neighborhood of any vertex of T is a $(k-1)-tree$;
- if $n > k$, every vertex of T has degree at least k;
- T has no cutset of size less than k.

$k-trees$ have been extensively studied lately. Some NP-Hard problems are tractable when they are restricted to $k-trees$; for instance, the Steiner Problem in graphs, a classical NP-Hard problem, accepts a linear time algorithm when restricted to $2-trees$, see Wald and Colbourn [18]. Other optimization problems on $k-trees$ are discussed in the work of Granot and Skorin-Kapov [12]; in particular, they design polynomial algorithms for several $k-cable$ distance optimization problems. Finally, Fernández-Baca and Medepalli [11] considered the problem of finding a minimum-cost assignment of program modules for processors in a distributed system where one of the processors has a limited memory. This problem is NP-Hard, even if the communication graph is a *tree*; they showed that a fully polynomial-time approximation scheme exists for the case when the communication graph is a partial $k-tree$, that is, a graph which is embedded in a $k-tree$ with the same vertex set.

Arnborg et al. [2] have shown that the embedding problem of a partial $k-tree$ into a $k-tree$ is NP-Hard for an arbitrary k, but it can be solved in $O(n^{k+2})$ for fixed k. Bodlaender [7] improved their result and showed that the recognition and embedding of a partial $k-tree$, when k is fixed, can be solved in $O(n^2)$. For partial $2-trees$ and partial $3-trees$ the recognition and embedding problems have been solved in linear time by Wald and Colbourn [18] and Matousek and Thomas [15], respectively. Mc Morris et al. [14] applied the results of Arnborg et al. [2] for recognizing partial $k-trees$ on the recognition of c-triangulated $(k-1)$-colored graphs.

Arnborg and Proskurowski [3] have defined an algorithm design methodology, called a *reduction paradigm*, for partial $k-trees$ that leads to the development of efficient algorithms for a variety of NP-Hard problems restricted to partial $k-trees$. Mata-Montero [13] applied these results in his research on *resilience* of networks.

3 The Minimum Spanning k-Tree Problem

Definition 2. Let k be a positive integer and $G = (V, E)$ an undirected graph with non-negative edge lengths. The *Minimum Spanning k-Tree Problem* (MSkT) is to find in G a $k - tree$ with vertex set V and edge set $E' \subseteq E$ of minimum total length.

We note that the MS1T corresponds to the classical *minimum spanning tree problem*, for which there are some polynomial time algorithms, Prim's algorithm [16], and Kruskal's algorithm [1], for instance.

In his doctoral thesis, Bern [6] proved that MSkT is NP-Hard for $k \geq 2$, even if the input graph satisfies the triangular inequality, showing that 3-SAT is polynomially reduced to MSkT. Furthermore, Cai and Maffray [8] proved that MS2T is NP-Hard for maximal planar graphs.

We will need the following definitions. An *addition sequence* for a $k - tree$ T with $n \geq k$ vertices is a sequence $(v_1, K_1), (v_2, K_2), ..., (v_{n-k}, K_{n-k})$ where each v_i is a newly created vertex and each K_i is an already existing $k - clique$ to which v_i is to be attached. The first $k - clique$ K_1 will be called the *starting clique*, and v_1 is the first new vertex. We say that v_i is *added to clique* K_i. A vertex that is adjacent to every vertex in a clique K_i will be said to be *adjacent to* K_i. (ϕ, K_1) represents an addition sequence for the $k - tree$ consisting of just a $k - clique$ K_1.

The following Lemmas 1 and 2 were proved by Bern [6].

Lemma 1. The number of distinct $k - cliques$ in a $k - tree$ with n vertices is $1 + (n - k)k$.

Lemma 2. The number of legal addition sequences for $k - trees$ with n vertices, where $n > k$, is $\binom{n}{k} \prod_{i=1}^{n-k} i(ik - k + 1)$.

These Lemmas show that a brute force algorithm, based on Lemma 2, would have running time no better than $\Omega(\binom{n}{k} \prod_{i=1}^{n-k} i(ik - k + 1))$, which is $\Omega(n!(n - k - 1)!/k!)$ a quite slow algorithm.

Bern [6] designed an exact algorithm based on a dynamic programming approach, of much lower complexity than exhaustive search. He proved the following theorem.

Theorem 2. A minimum spanning $k - tree$ in an arbitrary n-node network can be computed in $O(n^{k+1} 3^n)$ time.

Because the high time complexity of the dynamic programming algorithm for finding exact solutions, we suggest a greedy based heuristics of polynomial time for finding good feasible solutions at a low computational cost.

4 Greedy Heuristics for MSkT.

The *length of a* $k - clique$ is the sum of the edge lengths belonging to the $k - clique$. The *distance* between a vertex $v \in V$ and a $k - clique$ in K is the sum of the edge lengths that connect v with every vertex of K.

The next algorithm computes feasible solutions for the MSkT in a complete graph with edge-lengths . The algorithm accomplishes a greedy strategy based on the recursive definition of the spanning $k - tree$ for finding solutions.

GREEDY
Input: A complete graph $G = (V, E)$ with non-negative edge lengths and an integer $k \geq 2$.
Output: A spanning $k - tree$ in G.
Method:
Step 1. (Compute the starting clique)
Choose arbitrarily a $k - clique$ in G. Let T be such a $k - clique$

Step 2. (Add a new vertex to partial $k - tree$ T)
Compute the smallest distance between a vertex not in T to some $k - clique$ in T. Add the new vertex and the corresponding edges to T.

Step 3. (Stop condition)
If T includes all vertices in V then the algorithm finishes with solution T. Otherwise, go to step 2.

We note that GREEDY is a natural generalization of Prim's algorithm [16] for the *minimum spanning tree problem* to the minimum spanning $k - tree$ problem.

PRIM'S ALGORITHM
Input: A connected graph $G = (V, E)$ with non-negative edge lengths.
Output: A minimum spanning tree in G.
Method:
Step 1.(Compute the initial vertex)
Choose arbitrarily a vertex u in G, forming a tree $T = (U, E')$, where $U = \{u\}$ and $E' = \emptyset$.

Step 2.(Add a new vertex to partial tree T)
Find an edge of minimum length between a tree vertex $u \in U$ and a non tree vertex $v \in V - U$. Add the new vertex v and the edge (u, v) to T.

Step 3. (Stop Condition)
If T includes all vertices in V then the algorithm finishes with solution T. Otherwise, go to step 2.

Lemma 3. GREEDY finds a spanning $k-tree$ in G.

Proof. Start with a k-clique in G. In the $(i-1)st$ iteration of step $2, i \geq 2$, we have a $k-$tree on $k+i-1$ vertices. In the $i-th$ iteration, construct the cheapest $k-tree$ on $k+i$ vertices that contains the above $k-tree$ on $k+i-1$ vertices.

We will use the following lemma in the analysis of GREEDY.$\diamondsuit$

Lemma 4. In the $i-th$ iteration of step 2, $i \geq 1$, there are $(ik-k+1)(n-k-i+1)$ distances for computing the lowest distance.

Proof. (Induction on the number of iterations)

At first iteration we have $n-k$ distances, because after step 1 there is one $k-clique$ and $n-k$ possible vertices for adding.

After $i-th$ iteration k new $k-cliques$ were created. We note that it is possible to eliminate from the set of distances those connecting the last vertex added with all $k-cliques$ formed until $(i-1)-th$ iteration, that is, we can eliminate $k(i-1)+1 = ik-k+1$ distances (by lemma 1). At the $(i+1)-th$ iteration there are $n-k-i$ possible vertices for adding, and then there are $k(n-k-i)$ new distances. Thus we have:

$$(ik-k+1)(n-k-i+1)-(ik-k+1)+k(n-k-i)=(ik+1)(n-k-i)$$

distances.$\diamondsuit$

Theorem 3. GREEDY finds a feasible solution for the MSkT problem in $O(k(n-k)^3)$ comparisons.

Proof. The time complexity of our heuristic is determined by step 2, because step 1 can be implemented in $O(1)$ time. We must consider comparisons done at every iteration in the step 2. Therefore, using lemma 4 we have:

$$\begin{aligned} S &= \textstyle\sum_{i=1}^{n-k}(ik-k+1)(n-k-i+1) \\ &= (nk-k^2-1)\tfrac{(n-k)(n-k-1)}{2} - \tfrac{k(n-k)(n-k+1)(2(n-k)+1)}{6} + (n-k)^2 \end{aligned}$$

Thus:

$$S = O(k(n-k)^3)$$

$\diamondsuit$

Remarks.

- If $k << n$ then the time complexity of GREEDY is $O(n^3)$, and if $k=n$ then the MSkT problem is transformed into a trivial one, that is, find a n-clique in a complete graph with n vertices.

- Consider the input graph $G = K_n$ for GREEDY and $l_{i,j} = 1, \forall\ (i,j) \in G$, $i \neq j$, except $l_{i_0 j_0} = M, i_0 \neq j_0, M \in R^+$. If GREEDY chooses in step 1 a $k - clique$ $(k \leq n-2)$ that contains the edge (i_0, j_0) then the length of the solution given by GREEDY is $k(k-1)/2 + k(n-k) - 1 + M$, and the optimal solution has length $k(k-1)/2 + k(n-k)$. Thus, when M is arbitrarily large GREEDY can give a solution far away of the optimum.

- It is possible to implement GREEDY considering two alternatives for step 1, the starting $k - clique$ is chosen randomly or the starting $k - clique$ is the $k - clique$ of minimum length.
  Furthermore, we can formulate a repetitive version of GREEDY which obtains $|V|$ solutions, such that at each repetition, step 1 chooses a different starting $k - clique$ within the first $|V|$ $k - cliques$ of minimum length. Probably, the set of $|V|$ solutions so obtained contains some repeated solutions. It is clear that repetitive GREEDY dominates simple GREEDY.

5 Experimental Results

In Beck and Candia [4] and Candia et al. [9] were proposed greedy heuristics for MS2T. After that, Beltrán and Skorin-Kapov [5], in a recent work developed four heuristic algorithms to obtain good feasible solutions for MS2T. They used these feasible solutions as starting solutions for a tabu search based heuristic. Computational experience tested on both random graphs and real data sets showed that all four heuristics occasionally led to the best solutions, dominating each other in different cases.

Table 1 shows our results obtained by GREEDY for MS2T, using two alternatives for step 1, H1 chooses randomly the starting *2-clique* and H2 chooses as starting *2-clique* the edge of minimum length, H3 is a repetitive version of GREEDY where the value shown is the best of the set of feasible solutions obtained considering the best $|V|$ *2-cliques* as starting *2-clique* in step 1. SH is one of the heuristics given by Beltrán and Skorin-Kapov [5] (Star Heuristic). It is based on an embedding of a minimum cost spanning tree into a *2-tree*. If $G = (V, E)$ is a weighted complete graph, the heuristic first finds a minimum spanning tree $T = (V, E_T)$ in G. For $i \in V$, let $N_T(i)$ represent the set of nodes adjacent to i in T. For each node i of degree greater than one in T, the corresponding star S_i is a subgraph of T induced by $\{i\}$ and all of its neighbours $N_T(i)$. For each star S_i we find a minimum spanning tree T_i on the subgraph of G induced by the node set $N_T(i)$. The column OPT shows the optimal solution (obtained by Bern's dynamic programming algorithm). The last column shows $dist = 100 * (H3 - OPT)/OPT$, that is, the relative distance to optimum.

n	H1	H2	H3	SH	OPT	*dist*
5	210	210	210	218	210	0.0
	161	161	161	161	161	0.0
	129	129	129	203	129	0.0
	246	234	234	250	234	0.0
	212	213	212	240	212	0.0
10	421	446	421	479	409	2.93
	282	282	278	303	269	3.35
	273	273	273	433	270	1.11
	252	326	252	419	252	0.0
	382	382	358	506	358	0.0

Table 1: Values of H1, H2, H3, SH, OPT and *dist* for $n = 5, 10$

It is clear from table 1 that, for small values of n, the heuristic H3 produces very good solutions. However, for higher values of n, because the computation increases so rapidly it is not possible to know the optimal solution. Table 2 compares H1, H2, H3 and SH and shows the relative distance between H3 and the best between H1, H2, and SH for random instances of different sizes. For most instances generated, H3 produces solutions with a cost significantly lower than H1, H2 and SH. It is also clear that GREEDY is not very sensitive to starting clique, that is, in some cases H1 is better than H2 and in other cases H2 is better than H1. Because the second phase of SH heuristic, that is, the process of embedding of a minimum cost spanning tree into a *2-tree* by local criteria of optimization, the performance of SH is really poor.

n	H1	H2	H3	SH	*dist*	n	H1	H2	H3	SH	*dist*
25	675	634	620	1428	2.26	75	1151	1171	1126	2991	2.20
	653	679	653	1113	0.00		1040	1102	975	3221	6.67
	631	608	608	946	0.00		1070	1138	986	2966	8.52
	598	584	551	993	5.99		1073	1103	1000	3098	7.30
	625	572	545	986	4.95		1048	999	997	3135	0.20
50	971	957	876	2279	9.25	100	1376	1337	1241	4367	7.74
	857	948	815	2345	5.15		1264	1261	1144	3694	10.23
	819	866	758	2147	8.05		1290	1377	1243	4011	3.78
	893	837	796	2130	5.15		1185	1181	1139	3923	7.30
	935	964	851	2296	9.87		1247	1239	1210	3823	2.40

Table 2: Values of H1, H2, H3 SH and *dist* for $n = 25, 50, 75, 100$

6 Conclusions

In closing, we have designed an heuristic for the *Minimum Spanning k-Tree Problem*. Being this problem NP-Hard, our heuristic obtain feasible solutions in polynomial time. The heuristic is a generalization of Prim's algorithm for solving the classical *Minimum Spanning Tree Problem*. Lemma 4 proved the heuristic's correctness and Theorem 3 analyzed the heuristic's complexity. We also analyzed some variations of the heuristic leading to improved numerical results.

We implemented GREEDY and its variations for MS2T obtaining feasible solutions at low computational cost. These solutions could serve as good starting solutions in efficient applications of other strategies to MS2T, for example, Tabu Search, Simulated Annealing or Genetic Algorithms.

Acknowledgment

We would like to thank Dr. Ricardo Baeza-Yates and the referees for valuable suggestions which led to a significantly improved exposition of the paper. Thanks to Prof. Nelson Maculan for stimulating discussions and provide us reference 6. This research was supported in part by a grant from FONDECYT # 1-95-0622.

References

[1] A.V. Aho, J. E. Hopcroft and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley Publishing Company, 1974.

[2] S. Arnborg, D. Corneil, and A. Proskurowski, *Complexity of Finding Embedding in a k-Tree*, SIAM Journal Alg. and Disc. Math. **8** (1987), 277-284.

[3] S. Arnborg and A. Proskurowski, *Linear Time Algorithms for NP-Hard Problems Restricted to Partial k-trees*, Discrete Applied Mathematics **23** (1988), 11-24.

[4] H. Beck and A. Candia, *An Heuristic for the Minimum Spanning 2-Tree Problem,* Apuntes de Ingeniería (Special number in Computer Science) **47** (1993), 97-109.(in Spanish).

[5] H. Beltrán and D. Skorin-Kapov, *On Minimum Cost Isolated Failure Immune Network,* Second International Conference on Telecommunications System-Modeling and Analysis, Vanderbilt University, Nashville Tennessee, 1993, 444-453.

[6] M. Bern, *Networks Design Problems: Steiner Trees and Spanning k-Trees*, Ph. D. Thesis, University of Berkeley, 1987.

[7] H. L. Bodlaender, *Improved Self-reduction Algorithms for Graphs with Bounded Treewidth,* in: Proceedings 15th International Workshop on Graph Theoretic Concepts in Computer Science WG'89, Lecture Notes in Computer Science **411**

(Springer Berlin 1990), 232-244.

[8] L. Cai and F. Maffray, *On the Spanning k-Tree Problem,* Discrete Applied Mathematics **44** (1993), 139-156.

[9] A. Candia, H. Beck, H. Bravo, M. Caro and R. Cornejo, *Optimal Design of Invulnerable Networks*, Research Report # 4732-92, Universidad de Tarapacá, March (1994) (in Spanish).

[10] A. Farley, *Networks Immune to Isolated Failures*, Networks **11** (1981), 255-268.

[11] D. Fernández-Baca and A. Medepalli, *Allocating Modules to Processors in a Distributed System with Limited Memory*, in: Proceedings of the XI International Conference of the Chilean Computer Science Society, University of Chile, 1991, 349-369.

[12] D. Granot and D. Skorin-Kapov,*On Some Optimization Problems on k-Trees and Partial k-Trees,* Discrete Applied Mathematics **48** (1994), 129-145.

[13] E. Mata-Montero, *Resilience of Partial k-tree Networks with Edge and Node Failures*, Networks **21** (1991), 321-344.

[14] F. R. McMorris, T. J. Tandy and T. Wimer, *Triangulating Vertex-Colored Graphs*, SIAM J. Discrete Math. **7** (1994), 296-306.

[15] J. Matousek and R. Thomas, *Algorithms Finding Tree-decomposition of Graphs*, J. Algorithms **12** (1991), 1-22.

[16] R. Prim, *Shortest Connection Networks and Some Generalizations*, Bell Systems Techn. J. **36** (1957), 1389.

[17] D. Rose , *On Simple Characterizations of k-trees,* Discrete Mathematics **7** (1974), 317-322.

[18] J. Wald and C. Colbourn, *Steiner Trees, Partial 2-Trees, and Minimum IFI Networks*, Networks **13** (1983), 159-167.