Proceedings of the 1995 ACM/SIGAPP Symposium on Applied Computing, February 26-28, 1995, Nashville, TN, pp. 337-343, ACM Press.

Solving the Three-Star Tree Isomorphism Problem Using Genetic Algorithms*

Faris N. Abuali, Roger L. Wainwright, and Dale A. Schoenefeld
Department of Mathematical and Computer Sciences, The University of Tulsa
abuali,rogerw,dschoen@euler.mcs.utulsa.edu

Abstract

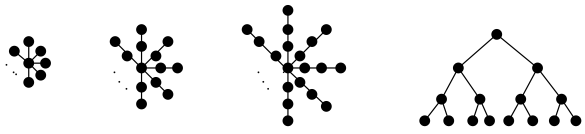
Genetic Algorithms (GA) are used to find minimum cost trees with special structures. The desired structure for the trees is M-Star isomorphic, where $M = 3$. The M-Star isomorphic problem is to find a spanning tree that is a star, and on each branch there are M vertices connected to form a path. The M-Star problem is NP-complete for $M > 2$. The cost is taken as the sum of the length of the edges forming the spanning tree. The determinant encoding, the Davis encoding, and the Prüfer encoding schemes are used to represent the spanning trees. The cost of a greedy algorithm is compared to the three GA encodings and a variety of crossover and mutation operators. Results show that the determinant encoding is better than the Prüfer encoding, and is as good as the Davis encoding for the 19 vertex problem. For the 61 vertex problem the Davis encoding produces better results over the determinant encoding in 8 out of 10 test cases.

Introduction

Papadimitriou and Yannakakis [14] analyzed the complexity of restricted spanning trees, and introduced the problem of isomorphism to some property P. The objective is to find a spanning tree that is of minimum length and has some specific property, P. This problem is denoted by MST(P). Example properties, P, are "isomorphic to 2-star", "isomorphic to 3-star", "isomorphic to full binary tree", and "isomorphic to a path". The "isomorphic to a path" problem is the traveling salesman problem (TSP). Figure 1 shows some examples of restricted spanning trees. Figure 1(a) is a star, Figure 1(b) is a 2-star, Figure 1(c) is a 3-star, and Figure 1(d) is a full binary tree. The problems "isomorphic to the star" and "isomorphic to the 2-star" are solved in $O(n^2)$ and $O(n^3)$ time respectively, where n is the number of vertices. The problems "isomorphic to 3-star" and "isomorphic to full binary tree" are NP-complete. Papadimitriou and Yannakakis show that the M-star isomorphism problem is NP-complete for $M > 2$ [14]. The M-star problem has applications in communication network design, and transportation and distribution systems.

---

* Research partially supported by OCAST Grant AR2-004 and Sun Microsystems, Inc.



(a) (b) (c) (d)

Figure 1: Some restricted spanning trees. (a) a star, (b) a 2-star. (c) a 3-star (d) a full binary tree.

In general, two graphs are said to be isomorphic if there is a one-to-one correspondence between their vertices and between their edges such that the incidence relationship among the vertices is preserved. That is, there is an edge between two vertices in one graph if and only if there is a corresponding edge between the corresponding vertices in the other graph. In the work presented here the property "isomorphic to 3-star" means that the spanning tree is a 3-star. An example 3-star is shown in Figure 1(c).

A k branch M-star means that the spanning tree has k branches. On each branch there are M vertices forming a path, and each branch is connected to the root. Such a tree is also called *regular* M-star. A k branch *non-regular* star tree is a spanning tree with k branches where each branch is connected to the root, and any number of vertices can form the path on each branch.

The remainder of the paper is organized as follows. Section 2 introduces the Prüfer encoding, and illustrates the encoding of M-stars with some examples. Section 3 defines our new determinant encoding and gives some examples for representing M-stars. Section 4 illustrates the use of the Davis encoding to represent M-stars. Section 5 derives formulas for counting the number of M-stars, and the number of M-stars and non-regular stars. Section 6 describes our greedy algorithm for generating M-stars. Section 7 gives a brief introduction to genetic algorithms (GA), reviews the encodings used to represent the M-stars, and describes the pool initialization strategy and the crossovers and mutations used. Finally, Section 8 describes the test cases, the results, and the conclusion.

Prüfer encoding

One of the first theorems in graphical enumeration is Cayley's theorem [6] that there are $n^{(n-2)}$ distinct labeled trees on n vertices. Prüfer [15] provided a constructive proof of Cayley's theorem by establishing a one-to-one correspondence between such trees and the set of all strings of $n-2$

integers between 1 and n.

The key to Prüfer's code is the observation that for any tree there are always at least two vertices of degree one. Hence, in a labeled tree T, the vertex v incident to the lowest labeled vertex is uniquely determined, and v becomes the first symbol in the string. After deleting this edge and node, we have a tree on $n-1$ vertices. Repeating this operation until one edge is left produces $n-2$ integers between 1 and n inclusive [18].

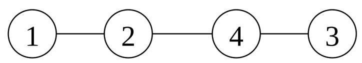
To reconstruct a tree T from Prüfer's code, we note that a particular vertex appears in the code exactly one time less than the degree of the vertex in T. Thus, it is possible to compute the degree of all the vertices of T, and identify the lowest labeled degree-one vertex in the tree, say u. Since the first symbol in the code is the vertex incident to u, the first edge is easily determined, and by repeating this operation, the remaining edges can be determined uniquely. Two examples are shown below illustrating Prüfer's encoding.

Example 1: Consider Prüfer's code for the 3 node complete graph; the code is represented as a single digit number in the range one to three. The result is 1, 2, or 3, each representing one of the three possible spanning trees. Code "1" represents the tree with the edges (1,2) and (1,3); code "2" represents the tree with the edges (2,1) and (2,3); and code "3" represents the tree with the edges (3,1) and (3,2).

Example 2: Prüfer's code for the 4 node complete graph, is represented as a string of 2 integers in the range one to four. Notice each possible 2-digit code represents a valid spanning tree. The codes for the sixteen possible spanning trees are as follows:

(1 1)	(2 1)	(3 1)	(4 1)
(1 2)	(2 2)	(3 2)	(4 2)
(1 3)	(2 3)	(3 3)	(4 3)
(1 4)	(2 4)	(3 4)	(4 4)

Consider for example the Prüfer code (2 4). The degree of nodes 1, 2, 3, 4 are respectively 1, 2, 1, 2. The smallest labeled node of degree one is node 1. Since 2 is the first allele, node 1 is adjacent to node 2 forming edge (1,2). Decrement the degree of node 1 to zero and decrement the degree of node 2 to one and continue. Now the smallest labeled node of degree one is node 2 and that must be adjacent to node 4 (the second allele). The resulting tree becomes:



For a more detailed discussion of Prüfer's encoding see [2].

The Prüfer Encoding of Star Trees

The Prüfer encoding is well suited for the representation of star trees. This is because the number of times a vertex appears in the code dictates the vertex's degree and hence the partial structure of the spanning tree. For example, for a non-regular star with k branches, the root must appear in the code $k-1$ times because the root's degree is k. The remaining vertices appear only once in the code, and the k leaf nodes do not appear at all.

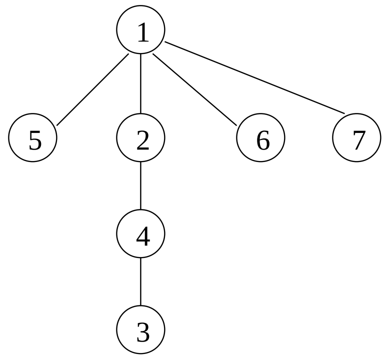


Figure 2: The Non-regular star corresponding to the Prüfer code "4 2 1 1 1".

For example, for a seven vertex non-regular star, the Prüfer code might be the integer string "4 2 1 1 1", where vertex one has a degree of four, vertices two and four have a degree of two, and vertices three, five, six, and seven are leaf nodes. The spanning tree corresponding to this Prüfer code is shown in Figure 2.

The Prüfer Encoding of M-Stars

From the previous section we can see that the Prüfer encoding is very conducive for the representation of star trees. Unfortunately, the representation of M-stars is less obvious because the interpretation of the Prüfer code is position dependent. Thus it is necessary to place the vertices in the appropriate positions to produce an M-star. These positions can only be found if one starts decoding the Prüfer code and then re-adjusts the position of the vertices. Sometimes M-stars can be produced by placing the root at $M-1$ intervals. Such codes produce an M-star if the root does not connect to a leaf, and k components are formed.

For example, for a 3-star with 3 branches, we have ten vertices in the graph. The Prüfer code for such a 3-star might be the integer string "1 2 10 3 4 10 5 6", where vertex ten has a degree of three, vertices 1, 2, 3, 4, 5, and 6 have a degree of two, and vertices 7, 8, and 9 are leaf nodes. The spanning tree uniquely corresponding to this Prüfer code is shown in Figure 3. The Prüfer code "4 5 10 6 7 10 8 9" is an example where placing the root (node 10) at $M-1$ intervals does NOT necessarily produce an M-star. The spanning tree corresponding to this Prüfer code is shown is Figure 4.

The Determinant Encoding

Define the edge-range vector R of a digraph $G(V,E)$ as follows: $R(j) = \{i \mid D(i,j) = -1\}$ for $j = 1$ to n where D is the in-degree matrix. Furthermore, define a determinant

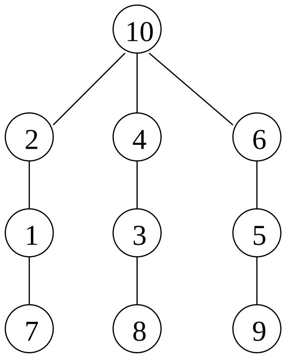


Figure 3: The 3-star corresponding to the Prüfer code "1 2 10 3 4 10 5 6".

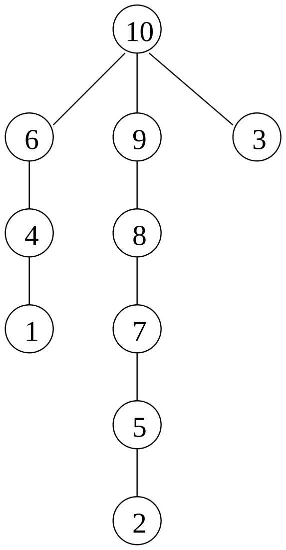


Figure 4: The Non-regular star corresponding to the Prüfer code "4 5 10 6 7 10 8 9".

code, DC to be a string of $n-1$ integers, where n is the number of vertices in the original graph, and the $j-1$ position in the string is selected from $R(j)$ for $j=2$ to n. For example, the set of all determinant codes for a 4 vertex graph is given by: $\{(x_2, x_3, x_4) \mid x_j \in R(j) \; \forall j = 2,3,4\}$. If DC is a determinant code, the $j-1$ position in DC represents the row index of an entry of -1 in the j-th column of the in-degree matrix. An x in the $j-1$ position of DC corresponds to an edge from vertex x to j. The reader is referred to [4].

For the star isomorphism problem we define a modified determinant code, MDC to be a string of n integers, where n is the number of vertices in the original graph, and the j position in the string is selected from $R(j)$ for $j=1$ to n. If we select vertex j to be the root then column j is skipped over when generating the tree from MDC.

The Determinant Encoding of Star Trees

The determinant encoding is highly robust in the representation of special spanning tree structures. To represent star trees with k branches, the determinant encoding places the root in the determinant code k times whereas in the Prüfer encoding the root is placed in the code $k-1$ times. The internal vertices appear only once, and the leaf vertices do NOT appear in the code at all. A code constructed under these restrictions produces a star tree with k branches, provided the code does NOT have a cycle. For example, for the seven vertex non-regular star shown in Figure 2, the determinant code is "0 1 4 2 1 1 1". The zero in the first position indicates that vertex one is the root and position one should be skipped.

The Determinant Encoding of M-Stars

The process for generating M-stars using the determinant encoding is straightforward. For each vertex, one places the label of the parent vertex in the code. For example, the determinant code corresponding to Figure 3 is "2 10 4 10 6 10 1 3 5 0", where the zero in column ten indicates that vertex 10 in the root. The determinant encoding scheme is discussed in more detail in [4].

The Davis Encoding

Jones and Beltramo [13] investigated partitioning problems using genetic algorithms. The goal is to partition a permutation into k groups according to some criterion. They presented several methods for encoding a partition as a permutation of n objects. One of the methods is a greedy heuristic which they attribute to L. Davis. The greedy heuristic takes the first k objects in the permutation to initialize the k groups. The remaining objects of the permutation are then added to the groups one at a time. Each of the remaining objects is added to the group that yields the best value for the objective function.

The Davis Encoding of Star Trees

The star tree problem can be thought of as a set partitioning problem, where the objective is to find the minimum cost partitioning of the vertices into k subsets representing the branches in the star tree, where $0 < k < n-1$, and the vertices on each branch are connected to form a path. In this case, the meaning of this permutation is as follows: Pick the first vertex in the permutation as the root. For the remaining vertices in the permutation, select the vertex in order, and connect it to the minimum cost leaf vertex, or connect it directly to the root whichever is least costly.

The Davis Encoding of M-Stars

The M-star problem can also be thought of as a set partitioning problem, where the objective is to find the minimum cost partitioning of the vertices into k subsets representing the branches in the M-star, and where each branch has M vertices connected to form a path. The meaning of this permutation for the M-star is as follows: Pick the first vertex in the permutation as the root. Assign the next k vertices to the k branches in order, connecting each of these vertices to the root. For the remaining vertices in the permutation, select each vertex in order, and connect it to the minimum cost leaf vertex, such that the number of vertices in that branch does not exceed M.

Counting The Number of Stars in a Graph

The Number of Regular Stars (M-Stars)

The number of stars with k branches, where each branch has M vertices (the total number of vertices is then $n = kM+1$) is found as follows: Assume that we have an integer string of size n, where the i-th position in the string contains the branch number that vertex i is in. Since we can select M positions (to be in the first branch) out of the n positions, this gives us $C(n, M)$ combinations for the first branch. Then, for the second branch we can select another M positions from the $n - M$ remaining positions. Similarly, we can do the same for the k-th branch. Hence the total number of combinations is:

$$\prod_{i=0}^{k-1} \binom{n-(iM)}{M} = \frac{n!}{(M!)^k}.$$

Then, within each branch there are $M!$ permutations for the M vertices. Finally, we have to remove duplicate strings where vertices are in virtually the same branch and are connected the same way, so we divide by $k!$. Therefore, the number of stars with k branches, where each branch has M vertices is:

$$\frac{(M!)^k}{k!} \prod_{i=0}^{k-1} \binom{n-(iM)}{M} = \frac{n!}{k!}.$$

The Number of Stars in a Graph (Regular and Non-regular)

The number of regular and non-regular stars with k branches can easily be computed if one approaches the problem as finding the number of Prüfer codes corresponding to regular or non-regular stars with k branches. We know that there is one-to-one correspondence between Prüfer codes and spanning trees. We also know that in the Prüfer code the root of the star must appear $k-1$ times in the $n-2$ positions in the Prüfer code. The number of different combination for placing the root in those $n-2$ positions is $C(n-2, k-1)$. There are n choices for the root, thus there are $n \times C(n-2, k-1)$ different Prüfer codes. For the remaining $n-2-(k-1)$ positions there are $n-1$ choices for the internal nodes, hence there are $P(n-1, n-k-1)$ permutations. Therefore, the number of Prüfer codes that correspond to regular and non-regular stars with k branches is:

$$n \times C(n-2, k-1) P(n-1, n-k-1)$$

or

$$\frac{(n-2)!}{(k-1)!(n-2-(k-1))!} \frac{n(n-1)!}{((n-1)-(n-k-1))!}$$

which is equal to:

$$\frac{(n-2)!}{(k-1)!(n-1-k)!} \frac{n!}{k!}$$

for a fixed k.

Finally, the number of stars with n nodes is:

$$\sum_{k=2}^{n-1} \frac{n!}{k!} \binom{n-2}{k-1}.$$

The Greedy Algorithm

The greedy algorithm that we developed to construct a k branch M-star spanning tree is described as follows. Select one of the vertices to be the center of the star. Next find the k closest vertices to the center, and create links between each vertex and the center. These vertices will make-up the first level of the star. For each vertex on the first level of the star, do the following: Let the first level vertex be the "current" vertex. For $i = 1$ to $M - 1$ find a vertex in the set of remaining vertices that is closest to the "current" vertex, and create a link to it. Update the "current" vertex to be the new vertex found. This algorithm is repeated where each vertex is considered as the center for the M-star, and the cost of the spanning tree was computed as the sum of the edge weights. The minimum cost spanning tree constructed in this manner is the result of the greedy algorithm.

Another approach for the M-star algorithm uses a Minimum Spanning Tree (MST) construction. The algorithm places the edges on a min-heap, then selects one edge at a time, adding the edge to form the spanning tree. One must ensure that by adding the edge no cycles are created and the current set of edges form a partially completed M-star.

Genetic Algorithm

Several researchers have investigated the benefits of solving combinatorial optimization problems using genetic algorithms [1, 3, 2, 5, 7, 11, 19]. Davis, Goldberg, and Rawlins provide excellent studies of genetic algorithms [9, 10, 12, 16]. It is assumed that the reader is familiar with the fundamentals of genetic algorithms(GA). The GA package used in this research is LibGA [8].

In this research, the three encoding schemes for M-star trees were used to find near-optimal solutions to the M-star isomorphism problem. The first encoding was the determinant encoding. The determinant encoding can be used to describe virtually any tree structure. For example, if we want to limit our solution space to non-regular stars, then each code must have unique alleles except for the root or center of the star which can occur as many times as branches. This is true if the determinant corresponding to the code is not zero (i.e. the determinant factor is non-singular.) Along the same line, if it is required that the spanning tree structure be an M-star, then each code must satisfy the conditions above, and each path from the root to a leaf must have a length of M.

The second encoding for M-star trees is the Prüfer encoding. Unlike the determinant encoding, where the structure of the spanning tree is intuitive, the structure of the spanning tree corresponding to the Prüfer code is difficult to visualize (without physically generating the tree). This is because the Prüfer encoding is position-dependent. The strategy we used to increase the likelihood of producing an M-star tree was to place the root of the star at $M-1$ intervals in the code. To verify that a spanning tree is an M-star we use path length information on each branch (must be 3, for the 3-star). We also ensure that the internal vertices have degree 2, and the root has degree k, for a k branch 3-star. This verification process is also used with determinant encoding and the Davis encoding.

The third encoding is an integer permutation of the vertices with the Davis encoding scheme. The integer permutation is mapped to a spanning tree through a greedy assignment process, to form the M-star. Notice any spanning tree structure can be realized using this representation.

In all of the GA expirements we used a generational model, with a roulette wheel selection, population size of 350 chromosomes, and a variety of crossover and mutation operators. The crossover rate was set at 1.0. The mutation rate was "adaptive" based on the population's standard deviation. The mutation rate increases as the population's standard deviation becomes smaller.

Pool Initialization

Initialization for the Determinant Encoding

Two strategies were used to initialize the first generation in the genetic algorithm: First, to initialize with non-regular k branch stars, select a vertex randomly to be the root. Then randomly select k different positions in the chromosome to place the root. None of the k positions can be the root, because the position in the code that equals the root is ignored. Remove the root from the set of remaining vertices. For each of the remaining unfilled positions in the chromosome randomly select a vertex from the set of remaining vertices, then remove the selected vertex from the set of remaining vertices.

The second strategy initializes with M-stars: The step for selecting and placing the root in the chromosome is the same as the non-regular k branch stars shown above. The second part involves selecting $n-1-k$ vertices from the $n-1$ remaining vertices to fill the remaining positions in the chromosome. This is accomplished as follows: For each position that the root was placed in, do the following: Let the current position be the selected position. Then for $i = 1$ to $M-1$ select a vertex randomly from the set of remaining vertices. Place the value of current position in the position corresponding to the selected vertex in the chromosome. Update the current position to be the newly selected vertex. Then update the set of remaining vertices by removing the selected vertex from consideration.

Initialization for the Prüfer Encoding

In this algorithm we fill in the $n-2$ positions in the Prüfer code by selecting from integers in the range 1 to n, making sure that each vertex appears once only in the code. Select the root from the remaining two vertices, and place it in $k-1$ positions selected at random. This process produces non-regular stars. It is possible to produce M-stars with this same process, but this is very difficult for large graphs. Instead, to increase the likelyhood of producing M-stars, we deterministically placed the root in the code. It was observed that if the root was placed at intervals of length $M-1$ that some M-stars were produced.

Initialization for the Davis Encoding

The initial pool was created by randomly generating permutations of integers in the range 1 to n, where n is the number of vertices in the graph. Then for each permutation the Davis encoding scheme was used to find the M-star corresponding to it.

Crossovers and Mutations

For the Determinant Encoding

For the determinant encoding we developed the crossover operators *Swap With Root*, *Swap Leaf With Root*, *Swap Roots*, *Swap Alleles*, and *Swap Vertices*. The *Swap With Root* operator is an asexual operator, that randomly selects a position in the chromosome and swaps the value in that position with the root of the spanning tree. The limitation to this crossover is that the leaf vertices do not participate in the crossover. That is, leaf vertices in the parent chromosome remain leaf vertices in the child chromosome. The *Swap Leaf With Root* is also an asexual crossover, where only leaf vertices are swapped with the root. The *Swap Roots* crossover uses two parents and simply swaps the roots of the two spanning trees. The *Swap Alleles* operator swaps two randomly selected values in the chromosome. Finally, *Swap Vertices* swaps two vertices in the spanning tree domain. The mutation operators *Swap With Root*, *Swap Leafs*, *Swap Vertices*, and *Swap Alleles* have the same behavior as the crossover operators described above.

For Prüfer and Davis Encodings

For the Prüfer encoding we used the crossover operators *Swap Alleles*, *Swap Roots*, and *Swap With Root* and the mutation operators *Swap Alleles* and *Swap Vertices*. The underlying behavior of the operators is the same as the operators used for the determinant encoding. For the Davis encoding we used the crossover operators *PMX* and *Cycle* and the mutation operator *Swap*. For information on *PMX* and *Cycle* see [12].

Test cases, Results, and Conclusions

Two data sets with 19 and 61 nodes were devised. In each case a complete graph was generated. The nodes were randomly placed on a 100×100 grid. The cost of each edge is its Euclidean length. Each data set was replicated 10 times using different random number seeds to generate the coordinate positions for the nodes. For each of the 10 test cases, the spanning tree cost obtained from the greedy algorithm was compared to the cost of spanning trees evolved by applying the genetic algorithm strategy using three different encodings of M-star spanning trees (Determinant, Prüfer, and Davis).

Table 1 shows the results for the 19 node problem ($M = 3, k = 6$) using the determinant encoding. The greedy algorithm used for constructing the spanning trees is described in Section 6. The spanning tree cost of the greedy algorithm was then computed. For the determinant encoding the pool was initialized with M-stars. We used the crossover operators *Swap With Root*, *Swap Leaf With Root*, *Swap Roots*, *Swap Alleles*, and *Swap Vertices*, and the mutation operators *Swap With Root*, *Swap Leafs*, *Swap Vertices*, and *Swap Alleles*. The results for these 20 combinations of crossovers and mutations (repeated 10 times each) are shown in Table 1. The best result in each case is italicized. In general, the best results for the determinant encoding were obtained when using *Swap Vertices* crossover. The determinant encoding produced better results than the greedy algorithm in all of the ten cases.

For the Prüfer encoding we used the crossover operators *Swap Alleles*, *Swap Roots*, and *Swap With Root* and the mutation operators *Swap Alleles* and *Swap Vertices*. The results for the six combinations of crossovers and mutations (each repeated 10 times) are shown in Table 2. Note in Table 2 for crossover *Swap With Root* that both mutation operators yield the same results. This is because the "best" solution for *Swap With Root* crossover was found in the first generation. The best result in each case is italicized. The best results for the Prüfer encoding were obtained when using *Swap Alleles* crossover and *Swap Vertices* mutation. The Prüfer encoding produced better results compared to the greedy in nine cases out of ten.

Table 3 shows the results of the three-star spanning tree problem for the 19 node data set using the Davis encoding. For the Davis encoding we used the crossover operators *PMX* and *Cycle* and the mutation operator *Swap*. These two crossover operators are generally best suited for order-based representations, and generally produce good results. Table 3 shows that both *PMX* and *Cycle* produce better results than the greedy algorithm. The *PMX* operator produced better results over *Cycle* six out of ten times. The greedy algorithm lost in every case.

Table 4 shows the results for the 61 node ($M = 3, k = 20$) problem using the Davis encoding with crossover *PMX* and mutation *Swap*, and the determinant encoding with crossover *Swap Vertices*, and mutation *Swap Vertices*. In Table 4 the Davis encoding produced better results than the determinant encoding in 8 out of 10 cases. In all 10 data cases the Davis encoding and the Determinant encoding produce better results than the greedy algorithm.

In conclusion we have shown a strategy to encode M-star spanning trees using the Prüfer encoding and the determinant encoding. A general encoding strategy such as the Davis encoding performs better than an encoding that is specific, such as the determinant encoding. We determined that the Davis encoding responds better to changes in the cost matrix since the same permutation of vertices could map to a different spanning tree. On the other hand, the determinant encoding is invariant to the cost matrix, since a chromosome will still map to the same spanning tree after a change in the cost matrix.

ACKNOWLEDGEMENTS
This research has been supported by OCAST Grant AR2-004. The authors wish to acknowledge the support of Sun Microsystems Inc. The authors would like to thank the referees, especially referee number 8 for his thorough reading of the paper which resulted in numerous improvements and clarifications.

References

[1] F. N. Abuali, D. A. Schoenefeld, and R. L. Wainwright. The design of a multipoint line topology for a communication network using genetic algorithms. In John Y. Cheung, editor, *Proceedings of the Seventh Oklahoma Symposium on Artificial Intelligence*, pages 101–110, Norman, Oklahoma, November 1993.

[2] F. N. Abuali, D. A. Schoenefeld, and R. L. Wainwright. Designing telecommunications networks using genetic algorithms and probabilistic minimum spanning trees. In Ed Deaton, K. M. George, Hal Berghel, and George Hedrick, editors, *Proceedings of the 1994 ACM/SIGAPP Symposium on Applied Computing*, pages 242–246, New York, 1994. ACM Press.

[3] F. N. Abuali, D. A. Schoenefeld, and R. L. Wainwright. Terminal assignment in a communications network using genetic algorithms. In Dawn Cizmar, editor, *Proceedings of the Twenty Second Annual ACM Computer Science Conference*, pages 74–81, New York, 1994. ACM Press.

[4] F. N. Abuali, R. L. Wainwright, and D. A. Schoenefeld. Determinant factorization and cycle basis: Encoding scheme for the representation of spanning trees on incomplete graphs. In K. M. George, Janice H. Carroll, Ed Deaton, Dave Oppenheim, and Jim Hightower, editors, *Proceedings of the 1994 ACM Symposium on Applied Computing*, pages 305–312, New York, 1995. ACM Press.

[5] J. L. Blanton and R. L. Wainwright. Multiple vehicle routing with time and capacity constraints using genetic algorithms. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 452–459, Urbana-Champaign, Illinois, July 1993. Morgan Kaufmann.

[6] A. Cayley. A theorem on trees. *Quarterly Journal of Mathematics*, 23:376–378, 1889.

[7] A. L. Corcoran and R. L. Wainwright. A genetic algorithm for packing in three dimensions. In Hal Berghel, Ed Deaton, George Hedrick, David Roach, and Roger Wainwright, editors, *Proceedings of the 1992 ACM/SIGAPP Symposium on Applied Computing*, pages 1021–1030, New York, 1992. ACM Press.

[8] A. L. Corcoran and R. L. Wainwright. LibGA: A user-friendly workbench for order-based genetic algorithm research. In Ed Deaton, K. M. George, Hal Berghel, and George Hedrick, editors, *Proceedings of the 1993 ACM/SIGAPP Symposium on Applied Computing*, pages 111–118, New York, 1993. ACM Press.

[9] L. Davis, editor. *Genetic Algorithms and Simulated Annealing*. Morgan Kaufmann, 1987.

[10] L. Davis, editor. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.

[11] Kenneth A. De Jong and W. M. Spears. Using genetic algorithms to solve np-complete problems. In Schaffer [17], pages 124–132.

[12] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, Massachusetts, 1989.

[13] David R. Jones and Mark A. Beltramo. Solving partitioning problems with genetic algorithms. In Richard K. Belew and Lashon B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 442–449, San Diego, California, 1991. Morgan Kaufmann.

[14] Christos H. Papadimitriou and Mihalis Yannakakis. The complexity of restricted spanning tree problems. *Journal of the Association for Computing Machinery*, 29(2):285–309, 1982.

[15] H. Prüfer. Neuer beweis eines satzes über permutationen. *Arch. Math. Phys.*, 27:742–744, 1918.

[16] G. Rawlins, editor. *Foundations of Genetic Algorithms*. Morgan Kaufmann, 1991.

[17] J. David Schaffer, editor. *Proceedings of the Third International Conference on Genetic Algorithms*, Arlington, Virginia, 1989. Morgan Kaufmann.

[18] S. Skiena. *Implementing Discrete Mathematics, Combinatorics and Graph Theory with Mathematica*. Addison Wesley, 1990.

[19] Darrell Whitley, T. Starkweather, and D. Fuquay. Scheduling problems and traveling salesman: The genetic edge recombination operator. In Schaffer [17].

Algorithm		Three-Star Cost									
Genetic Algorithm		Test Cases									
Crossover	Mutation	I	II	III	IV	V	VI	VII	VIII	IX	X
Swap With Root	Swap With Root	636	682	618	705	678	585	750	632	635	682
	Swap Leafs	588	635	618	676	515	488	632	588	592	610
	Swap Vertices	413	*417*	457	497	413	403	476	*365*	542	430
	Swap Alleles	520	621	551	663	483	483	587	541	556	479
Swap Leaf With Root	Swap With Root	580	648	583	612	558	491	604	537	539	582
	Swap Leafs	580	593	568	612	540	446	604	534	579	593
	Swap Vertices	444	583	583	499	467	416	528	502	484	531
	Swap Alleles	475	507	488	595	475	486	553	435	525	478
Swap Roots	Swap With Root	493	597	533	561	570	463	608	527	556	591
	Swap Leafs	464	593	593	512	527	463	579	535	534	517
	Swap Vertices	397	520	509	494	*383*	417	469	479	404	452
	Swap Alleles	551	591	488	563	475	431	559	504	501	504
Swap Alleles	Swap With Root	420	538	435	494	452	485	507	447	450	454
	Swap Leafs	416	533	435	491	476	508	507	445	519	436
	Swap Vertices	373	446	394	467	411	373	466	388	387	398
	Swap Alleles	444	448	494	499	479	471	523	483	446	464
Swap Vertices	Swap With Root	*359*	425	382	452	387	383	435	402	393	388
	Swap Leafs	383	456	391	*430*	405	383	*422*	397	*365*	399
	Swap Vertices	371	443	*378*	431	403	384	428	389	384	*385*
	Swap Alleles	416	435	406	441	*383*	*362*	434	382	381	388
Greedy		450	465	492	508	486	388	530	494	457	510

Table 1: The Three-Star Cost for 19 vertices, Determinant Encoding and the Greedy. The best is italicized.

Algorithm		Three-Star Cost									
Genetic Algorithm		Test Cases									
Crossover	Mutation	I	II	III	IV	V	VI	VII	VIII	IX	X
Swap Alleles	Swap Alleles	531	507	*395*	568	452	524	683	549	473	448
	Swap Vertices	*430*	504	473	*468*	*391*	*378*	*474*	*410*	*440*	*408*
Swap Roots	Swap Alleles	561	692	542	633	554	542	479	552	477	520
	Swap Vertices	443	619	608	558	565	515	475	614	529	545
Swap With Root	Swap Alleles	846	840	725	915	812	901	842	897	697	784
	Swap Vertices	846	840	725	915	812	901	842	897	697	784
Greedy		450	*465*	492	508	486	388	530	494	457	510

Table 2: The Three-Star Cost for 19 vertices, Prüfer Encoding and the Greedy. The best is italicized.

Algorithm		Three-Star Cost									
Genetic Algorithm		Test Cases									
Crossover	Mutation	I	II	III	IV	V	VI	VII	VIII	IX	X
PMX	Swap	380	*394*	*378*	473	*394*	*346*	436	*355*	*380*	385
Cycle	Swap	*368*	412	393	*441*	410	382	*424*	376	425	*372*
Greedy		450	465	492	508	486	388	530	494	457	510

Table 3: The Three-Star Cost for 19 vertices, Davis Encoding and the Greedy. The best is italicized.

Algorithm	Three-Star Cost									
	Test Cases									
	I	II	III	IV	V	VI	VII	VIII	IX	X
GA Davis Encoding	*1004*	1117	*1002*	*1001*	*1078*	*1108*	*1067*	*1031*	1065	*1028*
GA Determinant Encoding	1050	*1100*	1034	1036	1106	1158	1099	1080	*1062*	1053
Greedy	1213	1333	1232	1221	1350	1341	1366	1262	1256	1302

Table 4: The Three-Star Cost for 61 vertices, Davis Encoding, Determinant Encoding, and the Greedy. The best is italicized.