Proceedings of the 1995 ACM/SIGAPP Symposium on Applied Computing, February 26-28, 1995, Nashville, TN, pp. 305-312, ACM Press.

Determinant Factorization and Cycle Basis: Encoding Schemes for the Representation of Spanning Trees on Incomplete Graphs*

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

The genetic algorithm (GA) heuristic is used to find near optimal solutions for the Probabilistic Minimum Spanning Tree problem (PMST), an NP-complete variation of the classical minimum spanning tree problem. Given a connected graph *G(V,E)*, a cost function c:$E \longrightarrow \Re^+$, and a probability function P:$2^V \longrightarrow [0,1]$, the problem is to find an *a priori* spanning tree of minimum expected length. For the incomplete graph problem, a new encoding scheme for spanning trees that is based on the *cycle basis* is compared to another new encoding scheme that is based on the factorization of the determinant of the in-degree matrix of the original graph. For edge connectivity probability < 0.4 and problem size of 20 nodes, our results show a significant improvement in using the *cycle basis encoding* over the use of the *determinant encoding*, or a greedy algorithm. For the 30 and 40 node problems the determinant encoding performs better than the cycle basis encoding for most edge connectivity probabilities.

Introduction

The classical minimum spanning tree (MST) is used to help solve many combinatorial optimization problems. The MST problem has important applications in transportation, communication network design, and distribution systems.

Recently Bertsimas[4] defined the Probabilistic Minimum Spanning Tree (PMST), which is a variation of the minimum spanning tree where each vertex is present with some given probability. The PMST model is more realistic and representative of many combinatorial optimization problems than the minimum spanning tree, especially if the MST is NOT the "best" solution for all instances of the problem. Note when the vertex probability is one, for each vertex, the PMST problem reduces to the MST problem. Many applications are natural for the PMST such as VLSI design, communication network design, and organizational structures design.

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

In Section 2 a formal description of the PMST problem is presented. Section 3 discusses our new cycle basis encoding, with some examples. Section 4 gives an introduction to the determinant factorization and defines the determinant encoding. Section 5 describes the genetic algorithm test cases and results.

The PMST Problem

Bertsimas has formally defined the Probabilistic Minimum Spanning Tree (PMST) problem as follows [4]: Given a connected undirected graph *G=(V,E)* , not necessarily complete, a cost function c:$E \longrightarrow \Re^+$, and a probability function P:$2^V \longrightarrow [0,1]$, the objective is to find a spanning tree, say T, that minimizes the expected active cost, $E[L_T]$, where

$$E[L_T] = \sum_{S \subseteq V} p(S) L_T(S).$$

The above summation is taken over all subsets of V, *T(S)* is the minimum subtree of T that is required to interconnect all the nodes in S, and $L_T(S)$ is the total cost of the edges in *T(S)*.

If one assumes that node i is active with probability p_i and that the nodes are independent, Bertsimas shows that the expected active cost, $E[L_T]$, for a given spanning tree is given by the expression

$$E[L_T] = \sum_{e \in T} c(e) \left\{ 1 - \prod_{i \in K_e} (1 - p_i) \right\} \left\{ 1 - \prod_{i \in V - K_e} (1 - p_i) \right\}.$$

The above summation is over all edges in the tree T. The removal of an edge, e, splits T into two subtrees. The set of vertices in one subtree is denoted by K_e and the set of vertices in the other subtree is denoted by $V - K_e$. The above summation computing $E[L_T]$ is $O(n^2)$ where n is the number of vertices. Bertsimas also shows how to compute $E[L_T]$ in $O(n)$ time.

Cycle Basis Encoding

In this section, we describe our new *cycle basis encoding* scheme. The set of all spanning trees in a connected graph, $G(V, E)$, corresponds to the set of feasible chromosomes resulting from the application of the scheme.

One of the traditional uses of a *cycle basis* for a graph is in computing the *cycle vector* invariant[14]. The cycle vector is an n-vector where the i-th component is the number of cycles of length i in a graph with n vertices. To compute this cycle vector one needs to find all of the cycles of the graph G. The first step in finding all of the cycles is to find any spanning tree T for G. Then, the addition of any single edge, e, to T results in the formation of a cycle, which can be found by removing all vertices of degree one from $T \bigcup \{e\}$ until none are left. The set of all cycles found in this way is a cycle basis for G.

If G has n vertices and e edges, then any spanning tree of G has $n-1$ edges. Hence, there are $e-n+1$ cycles in any cycle basis for G. Each cycle of G can be obtained by "adding", modulo 2, elements of a subset of the cycle basis. Conversely, each such modulo 2 addition introduces a possible cycle of G. The *cycle space* of G consists of all cycles of G.

The cycle basis encoding scheme is based on finding a cycle basis and, then for each cycle in the cycle basis, selecting an edge from that cycle for removal from G. Let the i-th allele of a chromosome indicate the label of the edge selected from the i-th cycle in the cycle basis for removal from G. The length of the chromosome is $e-n+1$, the number of cycles in a cycle basis. A chromosome is feasible if the collection of edges remaining in G after removal of the $e-n+1$ edges (one from each element of the cycle basis) is a spanning tree. Otherwise a chromosome is infeasible. The process of selecting the edges from the corresponding cycles and removing them from G will produce a tree if the code is valid (i.e. the removal of an edge does not cause components to become disconnected.) The length of the encoding scheme is the same as the number of cycles in the basis (i.e. $e-n+1$). We can select $e-n+1$ cycles from the cycle space as the basis for the cycle basis encoding so long as $\bigcup_{i \in C} C_i = E$ where C is the set of cycles in the basis, and E is the set of edges in the graph G. This condition is sufficient but not necessary. Two examples follow that illustrate the cycle basis encoding scheme for a spanning tree.

Example 1: Consider a 7 node graph shown in Figure 1(a), and an example spanning tree shown in Figure 1(b). From Figure 1(b) we see that the missing edges are edge e_2 and edge e_3. By adding edge e_2, cycle C_1 is formed consisting of edges e_1, e_2, e_6, e_5 as shown in Figure 1(c). When edge e_3 is added to Figure 1(b) cycle C_2 is formed, with edges e_6, e_3, e_4 as shown in Figure 1(d). Thus the resulting cycle basis from the spanning tree in Figure 1(b) for the graph in Figure 1(a) is cycles C_1 and C_2. To find all the cycles in the original graph we can take all combinations of the cycles in the cycle basis (if the number of combinations is NOT too large) or use the algorithms written by John T. Welch and Norman E. Gibbs[18, 12] that determine all cycles in a graph. In this case to find all the cycles in Figure 1(a) we add the cycles of the cycle basis together, as edge sets, modulo 2. For example, we may write $C_1 = e_1 + e_2 + e_6 + e_5$ and $C_2 = e_6 + e_3 + e_4$, then

$$
\begin{aligned}
C_1 + C_2 &= (e_1 + e_2 + e_6 + e_5 + e_6 + e_3 + e_4) \bmod 2 \\
&= (e_1 + e_2 + 2e_6 + e_5 + e_3 + e_4) \bmod 2 \\
&= e_1 + e_2 + e_3 + e_4 + e_5
\end{aligned}
$$

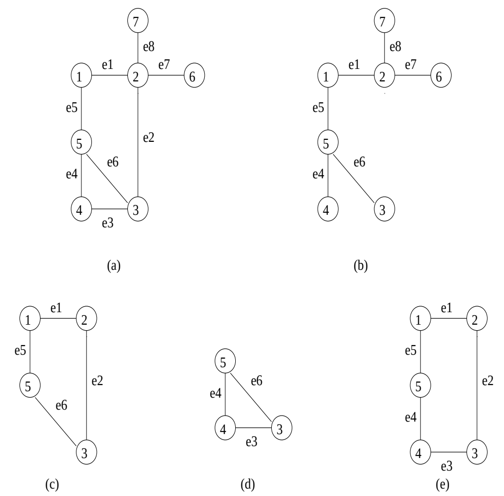

Figure 1: An incomplete graph with seven nodes and two cycles. (a) The original graph G, (b) A spanning tree from G, (c) The cycle C_1. (d) The cycle C_2, (e) The cycle $C_3 = C_1 + C_2$.

Using the cycle basis consisting of C_1 and C_2 for the cycle basis encoding, there are 12 combinations of edges to select and remove from Figure 1(a). The actual number of spanning trees is 11. The following table shows all of the possible combinations of $C_1 \times C_2 = (1, 2, 6, 5) \times (6, 3, 4)$.

(1 3)	(2 3)	(5 3)	(6 3)
(1 4)	(2 4)	(5 4)	(6 4)
(1 6)	(2 6)	(5 6)	(6 6)

The 12 possible spanning trees corresponding to the above 12 codes (chromosomes) are shown in Figure 2. Edges e_7 and e_8 are in all the trees but are not shown.

Notice if we select cycles C_1 and C_3 to form the cycle basis, then there are 20 combinations of edges to remove from Figure 1(a), and if we select cycles C_2 and C_3 to form the cycle basis, we have 15 combinations of edges to remove from Figure 1(a). The increase in the number of combinations indicates that some chromosomes are infeasible (i.e. do NOT produce a tree) or that there is a many to one relation where many valid chromosomes map to the same tree.

Example 2: The seven node graph shown in Figure 3(a) has four cycles in the cycle basis. Figure 3(b), Figure 3(c), and Figure 3(d) show three possible spanning trees on the original graph in Figure 3(a). Using the spanning tree in Figure 3(b) we get the cycles C_1, C_2, C_3, and C_4 shown below, with search space size of $3 \times 6 \times 5 \times 3 = 270$. If we use the spanning tree in Figure 3(c) to form a cycle basis, we find the cycles $C_1, C_4, C_2 + C_4$, and $C_3 + C_4$, with search space size of $3 \times 3 \times 5 \times 4 = 180$. Finally, if we use the spanning tree in Figure 3(d) to form a cycle basis, we find the cycles $C_1, C_2 + C_3, C_4$, and $C_3 + C_4$ with search space size of $3 \times 3 \times 3 \times 4 = 108$. It is important to note that the cycles used in the cycle basis play a major role in the size of the resulting search space. Since every cycle is a *mod* 2 sum of elements in a subset of the cycle basis, the cardinality of the cycle space in this example is bounded above by $2^4 - 1 = 15$.

Determinant Encoding

The Number of Spanning Trees

Even [11] describes the following technique for finding the number of spanning trees in a directed graph (developed by Tutte [17]):
The in-degree matrix D of a digraph $G(V,E)$ is defined as follows:

$$D(i,j) = \begin{cases} d_{in(i)} & \text{if } i = j, \\ -k & \text{if } i \neq j \end{cases}$$

where $d_{in(i)}$ is the number of edges coming into node i, and k is the number of edges in G from i to j.

Lemma 1 ([11]) *A finite digraph G(V,E), with no self-loops is a directed tree with root r if and only if its in-degree matrix* D *has the following two properties:*

1. *The root has no edges coming in, and all other nodes have one edge coming in, formally,*

$$D(i,i) = \begin{cases} 0 & \text{if } i = r, \\ 1 & \text{if } i \neq r. \end{cases}$$

2. *The minor, resulting from erasing the* r*th row and column from* D *and computing the determinant, is 1.*

Theorem 1 ([11]) *The number of directed spanning trees with root r of a digraph with no self-loops is given by the minor of its in-degree matrix which results from the removal of the* r*th row and column.*

The PMST problem is defined on an undirected graph $G(V,E)$, and any solution to the PMST problem is an undirected tree T. To see the relationship between directed and undirected spanning trees, consider the digraph $G'(V, E')$ defined as follows: For every edge $u \overset{e}{-} v$ in G define the two edges $u \overset{e'}{\rightarrow} v$ and $v \overset{e''}{\rightarrow} u$ in G'. Regardless of the choice of $r \in V$, there is a one-to-one correspondence between the set of spanning trees of G and the set of directed spanning trees of $G'(V, E')$ with root r: Let T be a spanning tree of G. If the edge $u \overset{e}{-} v$ is in T and if u is closer than v to r in T then pick e' for T'; if v is closer, pick e''. Also, given T', it is easy to find the corresponding T by simply ignoring the directions; i.e., the existence of either e' or e'' in T' implies that e is in T[11]. Defining the *degree* matrix of G to be the in-degree matrix of G', we can state:

Theorem 2 ([11]) *The number of spanning trees of an undirected graph with no self-loops is equal to any of the minors of its degree matrix which results from the erasure of a row and a corresponding column.*

The 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 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. A k in the $j-1$ position of DC corresponds to an edge from vertex k to j.

Two examples are shown below illustrating the determinant factorization.

Example 1: Given the in-complete graph in Figure 1(a), the in-degree matrix D is:

$$D = \begin{bmatrix} 2 & -1 & 0 & 0 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 & 0 & -1 & -1 \\ 0 & -1 & 3 & -1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ -1 & 0 & -1 & -1 & 3 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

If we remove the first row and column (assuming vertex 1 to be the root), and compute the determinant of the resulting matrix, which is the number of spanning trees T, we get:

$$T = \begin{vmatrix} 4 & -1 & 0 & 0 & -1 & -1 \\ -1 & 3 & -1 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & -1 & 3 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 \end{vmatrix} = 11.$$

The location of the negative one's in the in-degree matrix are listed below:

column:	2	3	4	5	6	7
	1	2	3	1	2	2
	3	4	5	3		
Row:	6	5		4		
	7					

The number of combinations for the determinant codes is $4 \times 3 \times 2 \times 3 \times 1 \times 1 = 72$. The first few determinant codes are listed below in lexical order:

$$\begin{array}{llllll} (1 & 2 & 3 & 1 & 2 & 2) \\ (1 & 2 & 3 & 3 & 2 & 2) \\ (1 & 2 & 3 & 4 & 2 & 2) \\ (1 & 2 & 5 & 1 & 2 & 2) \end{array}$$

and so on.

The above code (1 2 5 1 2 2), for example represents a spanning tree for Figure 1(a). The value of "1" as the first allele, represents the -1 in position (1,2) in D which defines edge (1,2). The second allele, 2, defines edge (2,3), and the remaining alleles 5 1 2 2 represent edges (5,4), (1,5), (2,6), and (2,7) respectively. This corresponds to the edges $e_1, e_2, e_4, e_5, e_7, e_8$, respectively.

Example 2: Given the in-complete graph in Figure 3(a), the in-degree matrix D is:

$$D = \begin{bmatrix} 2 & -1 & 0 & 0 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 & 0 & -1 & -1 \\ 0 & -1 & 4 & -1 & -1 & 0 & -1 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ -1 & 0 & -1 & -1 & 3 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 2 & -1 \\ 0 & -1 & -1 & 0 & 0 & -1 & 3 \end{bmatrix}$$

If we remove the first row and column (assuming vertex 1 to be the root), and compute the determinant of the resulting matrix (i.e the number of spanning trees, T) we get:

$$T = \begin{vmatrix} 4 & -1 & 0 & 0 & -1 & -1 \\ -1 & 4 & -1 & -1 & 0 & -1 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & -1 & 3 & 0 & 0 \\ -1 & 0 & 0 & 0 & 2 & -1 \\ -1 & -1 & 0 & 0 & -1 & 3 \end{vmatrix} = 79.$$

The location of the negative one's in the in-degree matrix are listed below:

column:	2	3	4	5	6	7
	1	2	3	1	2	2
	3	4	5	3	7	3
Row:	6	5		4		6
	7	7				

The number of combinations of the determinant code is $4 \times 4 \times 2 \times 3 \times 2 \times 3 = 576$. The first few determinant codes are listed below in lexical order:

$$\begin{array}{l} (1\ \ 2\ \ 3\ \ 1\ \ 2\ \ 2) \\ (1\ \ 2\ \ 3\ \ 1\ \ 2\ \ 3) \\ (1\ \ 2\ \ 3\ \ 1\ \ 2\ \ 6) \\ (1\ \ 2\ \ 3\ \ 1\ \ 7\ \ 2) \end{array}$$

and so on.

Genetic Algorithm Test Cases and Results

Several researchers have investigated the benefits of solving combinatorial optimization problems using genetic algorithms [1, 3, 2, 5, 6, 10, 19]. Davis, Goldberg and Rawlins provide an excellent in depth study of genetic algorithms [8, 9, 13, 15]. It is assumed that the reader is familiar with the fundamentals of genetic algorithms(GA). The GA package used in this research is LibGA [7].

Three data sets with 20, 30, and 40 nodes were devised. In each case the edge connectivity probability of 0.1, 0.2, 0.3, 0.4 and 0.5 was considered. Edge connectivity probability is the probability that an edge is present between any randomly chosen pair of vertices. As the edge connectivity increases, the graph becomes more edge-dense. The nodes were randomly placed on a 100×100 grid. The cost of each edge is the Euclidean distance. The PMST probability as defined in Section 2, associated with each node was fixed at 0.1. Each data set was replicated 10 times using different random number seeds to generate the coordinate positions for the nodes, and to establish the connectivity (placement of the edges). For each of the 10 repetitions, the expected active cost of a greedy strategy was compared to the expected active cost of various spanning trees evolved by applying genetic algorithm strategies to our two new encodings of spanning trees (determinant and cycle basis). The greedy strategy simply finds the expected active cost of the classical minimum spanning tree.

Table 1 shows the results of the PMST problem for the 20 node data set with an edge connectivity probability of 0.1. For the cycle basis encoding and the determinant encoding, we used the crossover operator *uniform* and the mutation operator *Alter Allele*. The expected active cost for the PMST problem using the greedy strategy, the determinant encoding, and the cycle basis encoding are shown for each of the 10 random number seeds. The percent improvement column in Table 1 compares the best of the determinant or cycle basis encodings to the greedy strategy. From Table 1 we can see a significant improvement using the cycle basis encoding. Also, in Table 1, the cycle basis encoding produced the same or better expected active cost compared to the determinant encoding in each of the 10 cases. The symbol NF in the tables means that no feasible chromosomes were generated. The cycle basis encoding scheme yielded the same or superior results compared to the greedy algorithm in all of the 10 cases. In all of the tables an "*" indicates the best results obtained.

The results for the 20 node problem with edge connectivity probabilities 0.2, 0.3, 0.4, and 0.5 are shown in Table 2 through Table 5, respectively. Notice in all of the 20 node cases shown in Table 1 through Table 5 that in every instance except one (test case IX in Table 4) that one of the two GA algorithms was always superior to the greedy algorithm. Notice further in Table 2 through Table 5 that as the edge connectivity increases from 0.1 to 0.5 that the better of the two GA results swings from cycle basis for the lower edge connectivity probabilities to the determinant scheme for the higher edge connectivity probabilities. This is easily observed by noting the pattern of "*" from one table to the next.

The results for the 30 node problem with edge connectivity probabilities of 0.1, 0.2, 0.3, 0.4, and 0.5 are shown in Table 6 through Table 10, respectively. Notice in all of the 30 node cases that in every instance one of the two GA algorithms was always superior to the greedy algorithm. Notice further in Table 6 through Table 10 that as the edge connectivity increases from 0.1 to 0.5 that the better of the two GA results swings from cycle basis for the lower edge connectivity probabilities to the determinant scheme for the higher edge connectivity probabilities. Again this is easily observed by noting the pattern of "*" from one table to the next. In fact the only case in which the cycle basis was clearly superior was in the 0.1 edge connectivity probability case (Table 6).

The results for the 40 node problem with edge connectivity probabilities of 0.1, 0.2, 0.3, 0.4, and 0.5 are shown in Table 11 through Table 15, respectively. The results for the 40 cases are virtually a repeat of the results for the 20 and 30 node cases. Notice in all of the 40 node cases that in every instance except one (test case V in Table 12) that one of the two GA algorithms was always superior to the greedy algorithm. Notice further in Table 11 through Table 15 that as the edge connectivity increases from 0.1 to 0.5 that the better of the two GA results swings from cycle basis for the lower edge connectivity probabilities to the determinant scheme for the higher edge connectivity probabilities. Again this is easily observed by noting the pattern of "*" from one table to the next.

In conclusion, we have presented a determinant encoding scheme and a cycle basis encoding scheme to represent spanning trees in an attempt to solve the PMST problem on incomplete graphs. In all of our test cases we used a node probability of 0.1, which is quite realistic. The GA algorithms using either the determinant or the cycle basis encoding scheme clearly outperformed the greedy algorithm (the best known algorithm to date for this problem). We observed in general that the cycle basis encoding scheme works best for graphs with edge connectivity probability ≤ 0.2, and the determinant encoding scheme works best for edge connectivity probabilities ≥ 0.2. Another significant point to this research is that the new encoding schemes presented for spanning trees are not limitd to GA chromosomes, but indeed are a general encoding scheme.

Acknowledgements

This research has been supported by OCAST Grant AR2-004. The authors wish to acknowledge the support of Sun Microsystems Inc.

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] D. J. Bertsimas. The probabilistic minimum spanning tree problem. *Networks*, 20:245–275, 1990.

[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. 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.

[7] 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.

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

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

[10] Kenneth A. De Jong and W. M. Spears. Using genetic algorithms to solve np-complete problems. In Schaffer [16], pages 124–132.

[11] S. Even. *Graph Algorithms.* Computer Science Press, 1979.

[12] N. E. Gibbs. A cycle generation algorithm for finite undirected linear graphs. *Journal of the Association for Computing Machinery*, 16:564–568, 1969.

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

[14] R. R. Korfhage. *Discrete Computational Structures.* Academic Press, 1974.

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

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

[17] W. T. Tutte. The dissection of equilateral triangles into equilateral triangles. *Proc. Cambridge Phil. Soc.*, 44:463–482, 1948.

[18] John T. Welch. A mechanical analysis of the cyclic structure of undirected linear graphs. *Journal of the Association for Computing Machinery*, 13:205–210, 1966.

[19] Darrell Whitley, T. Starkweather, and D. Fuquay. Scheduling problems and traveling salesman: The genetic edge recombination operator. In Schaffer [16].

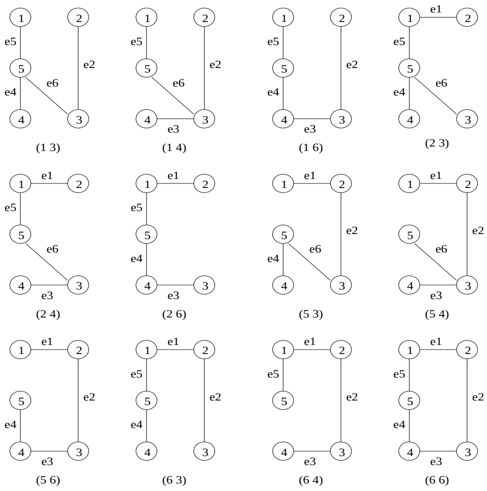

Figure 2: The graphs corresponding to the *cycle basis encoding* using cycles C_1 and C_2, 11 out of the 12 graphs are spanning trees. The (6 6) cycle code may be modified to map to a valid spanning tree. For simplicity, edges e_7 and e_8 are in all the trees above, but are not shown.

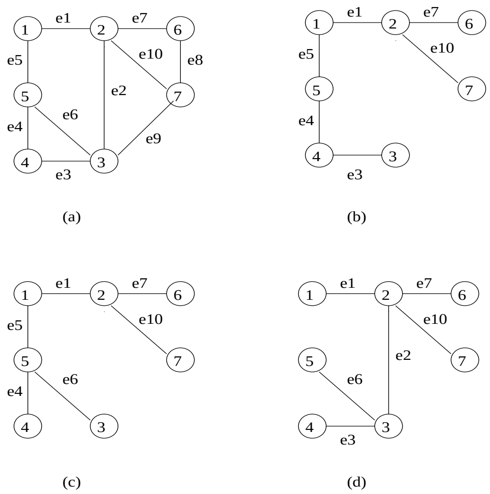

Figure 3: An incomplete graph with seven nodes and four cycles. (a) The original graph G, (b) Spanning tree I from G, (c) Spanning tree II from G, (d) Spanning tree III from G

Test Cases	Algorithms			Percent Improvement
	Greedy	Det. (GA)	Cycle basis (GA)	
I	197.5	NF	195.2*	1.1
II	104.6*	NF	104.6*	0.0
III	151.4	144.2*	144.2*	4.8
IV	116.2	NF	113.0*	2.7
V	143.0	NF	140.9*	1.5
VI	133.7	181.4	125.0*	6.5
VII	124.2	173.6	122.1*	1.8
VIII	111.1	110.3*	110.3*	0.7
IX	117.7	NF	115.8*	1.5
X	143.1	149.0	141.9*	0.8

Table 1: The Expected Active Cost for the PMST problem, for 20 nodes (node probability is 0.1), and edge connectivity probability 0.1. NF means no feasible chromosomes were generated.

Test Cases	Algorithms			Percent Improvement
	Greedy	Det. (GA)	Cycle basis (GA)	
I	96.3	NF	86.7*	10.0
II	124.2	105.5*	105.5*	15.1
III	84.9	79.6	79.3*	6.6
IV	119.7	NF	112.9*	5.7
V	105.8	94.1	94.0*	11.2
VI	100.7	85.5*	85.5*	15.1
VII	130.0	118.9	116.4*	10.4
VIII	153.8	NF	143.0*	7.0
IX	122.2	114.7	114.2*	6.6
X	116.8	122.1	112.5*	3.6

Table 2: The Expected Active Cost for the PMST problem, for 20 nodes (node probability is 0.1), and edge connectivity probability 0.2. NF means no feasible chromosomes were generated.

Test Cases	Algorithms			Percent Improvement
	Greedy	Det. (GA)	Cycle basis (GA)	
I	84.1	82.2	79.7*	5.2
II	91.6	85.0*	86.8	7.1
III	76.3	73.4*	76.2	3.8
IV	108.8	96.5	93.8*	13.7
V	83.9	78.8	77.0*	8.2
VI	90.8	70.7*	73.6	22.1
VII	108.0	94.2*	94.8	12.7
VIII	97.8	95.1	88.0*	10.0
IX	125.5	101.8	99.9*	20.4
X	97.0	99.1	93.3*	3.8

Table 3: The Expected Active Cost for the PMST problem, for 20 nodes (node probability is 0.1), and edge connectivity probability 0.3.

Test Cases	Algorithms			Percent Improvement
	Greedy	Det. (GA)	Cycle basis (GA)	
I	73.8	67.1*	78.8	9.1
II	86.0	79.6*	79.9	7.4
III	75.4	67.1*	73.3	11.1
IV	104.0	86.5*	90.0	16.8
V	69.9	68.1*	69.7	2.5
VI	62.4	57.1*	63.7	8.4
VII	84.0	74.8*	78.0	11.0
VIII	77.9	75.2*	75.7	3.5
IX	83.9*	85.6	94.7	-2.0
X	84.8	73.7*	75.0	13.0

Table 4: The Expected Active Cost for the PMST problem, for 20 nodes (node probability is 0.1), and edge connectivity probability 0.4.

Test Cases	Algorithms			Percent Improvement
	Greedy	Det. (GA)	Cycle basis (GA)	
I	68.8	60.1*	65.1	12.7
II	78.0	69.1*	69.4	11.4
III	72.7	65.4*	69.5	9.9
IV	89.0	75.5*	80.4	15.2
V	68.5	63.1*	65.4	7.8
VI	56.8	55.5*	57.8	2.4
VII	93.8	71.2*	74.1	24.1
VIII	76.8	69.8*	70.6	9.1
IX	86.4	73.7*	100.0	14.7
X	78.9	65.8*	66.1	16.7

Table 5: The Expected Active Cost for the PMST problem, for 20 nodes (node probability is 0.1), and edge connectivity probability 0.5.

Test Cases	Algorithms			Percent Improvement
	Greedy	Det. (GA)	Cycle basis (GA)	
I	226.7	210.5	207.3*	8.5
II	235.3	NF	185.4*	21.2
III	215.9	NF	212.4*	1.7
IV	200.5	NF	187.0*	6.7
V	252.1	206.8	197.8*	21.5
VI	273.7	NF	212.4*	22.4
VII	234.9	269.7	205.5*	12.5
VIII	227.7	213.4	207.3*	9.0
IX	245.1	NF	222.6*	9.2
X	210.6	NF	191.7*	9.0

Table 6: The Expected Active Cost for the PMST problem, for 30 nodes (node probability is 0.1), and edge connectivity probability 0.1. NF means no feasible chromosomes were generated.

Test Cases	Algorithms			Percent Improvement
	Greedy	Det. (GA)	Cycle basis (GA)	
I	196.2	198.3	172.4*	12.1
II	146.4	128.2*	130.3	12.4
III	214.6	151.2	150.2*	30.0
IV	167.0	146.6*	152.8	12.2
V	170.2	144.2*	149.9	15.3
VI	175.1	143.1*	156.1	18.3
VII	229.3	184.4*	188.0	19.6
VIII	179.8	153.1*	153.1*	14.8
IX	204.3	178.0*	186.6	12.9
X	160.3*	177.0	167.6	-4.6

Table 7: The Expected Active Cost for the PMST problem, for 30 nodes (node probability is 0.1), and edge connectivity probability 0.2.

Test Cases	Algorithms			Percent Improvement
	Greedy	Det. (GA)	Cycle basis (GA)	
I	157.0	124.1*	133.9	21.0
II	139.5	117.1*	126.4	16.1
III	171.0	120.6*	128.5	29.5
IV	164.8	122.5*	123.1	25.6
V	127.5	119.2*	125.8	6.5
VI	126.8	109.5*	112.2	13.7
VII	171.7	140.4*	140.4	18.2
VIII	153.5	127.3	122.7*	20.1
IX	207.3	138.1*	162.6	33.4
X	122.7	116.8*	122.4	4.8

Table 8: The Expected Active Cost for the PMST problem, for 30 nodes (node probability is 0.1), and edge connectivity probability 0.3.

Test Cases	Algorithms			Percent Improvement
	Greedy	Det. (GA)	Cycle basis (GA)	
I	141.1	121.1*	139.9	14.1
II	129.7	113.2*	115.7	12.7
III	152.9	108.1*	121.2	29.3
IV	164.6	103.1*	140.2	37.4
V	138.6	100.6*	100.7	27.4
VI	117.7	99.1*	106.8	15.8
VII	137.4	135.6	125.5*	8.6
VIII	138.8	124.0	120.9*	12.9
IX	140.9	117.8*	118.4	16.4
X	116.3	111.3*	130.8	4.3

Table 9: The Expected Active Cost for the PMST problem, for 30 nodes (node probability is 0.1), and edge connectivity probability 0.4.

Test Cases	Algorithms			Percent Improvement
	Greedy	Det. (GA)	Cycle basis (GA)	
I	109.9	96.8*	114.8	11.9
II	128.9	111.2*	125.7	13.8
III	112.0	110.1	104.7*	6.5
IV	137.9	98.2*	107.2	28.7
V	122.5	95.5*	99.2	22.1
VI	114.2	95.6*	102.6	16.3
VII	138.3	115.8*	125.3	16.3
VIII	135.8	111.2*	118.3	18.2
IX	116.6	106.6*	136.2	8.6
X	122.1	110.8*	112.8	9.3

Table 10: The Expected Active Cost for the PMST problem, for 30 nodes (node probability is 0.1), and edge connectivity probability 0.5.

Test Cases	Algorithms			Percent Improvement
	Greedy	Det. (GA)	Cycle basis (GA)	
I	380.7	NF	281.2*	26.1
II	319.0	261.7*	266.7	18.0
III	441.2	NF	367.8*	16.6
IV	350.1	NF	296.2*	15.4
V	336.9	NF	279.3*	17.1
VI	309.2	286.0	251.6*	18.6
VII	270.8	NF	248.5*	8.2
VIII	321.3	NF	278.3*	13.4
IX	333.9	261.3*	261.9	21.7
X	333.4	NF	243.3*	27.0

Table 11: The Expected Active Cost for the PMST problem, for 40 nodes (node probability is 0.1), and edge connectivity probability 0.1. NF means no feasible chromosomes were generated.

Test Cases	Algorithms			Percent Improvement
	Greedy	Det. (GA)	Cycle basis (GA)	
I	232.2	181.0*	249.2	22.0
II	265.1	207.6*	239.5	21.7
III	295.1	206.1	196.0*	33.6
IV	238.2	456.2	199.9*	16.1
V	202.0*	234.5	230.6	-14.2
VI	244.7	186.9*	217.2	23.6
VII	253.9	224.2	213.5*	15.9
VIII	242.7	273.5	225.0*	7.3
IX	275.3	NF	269.7*	2.0
X	244.6	200.8*	222.7	17.9

Table 12: The Expected Active Cost for the PMST problem, for 40 nodes (node probability is 0.1), and edge connectivity probability 0.2. NF means no feasible chromosomes were generated.

Test Cases	Algorithms			Percent Improvement
	Greedy	Det. (GA)	Cycle basis (GA)	
I	170.6	152.0*	184.1	10.9
II	224.2	188.1	181.7*	19.0
III	256.0	175.8	172.6*	32.6
IV	205.9	165.4*	185.0	19.7
V	219.1	203.9	194.0*	11.5
VI	182.8	170.3*	184.1	6.8
VII	236.3	187.2*	196.0	20.8
VIII	257.2	192.4*	205.3	25.2
IX	241.2	196.1*	212.6	18.7
X	208.9	169.7*	176.9	18.8

Table 13: The Expected Active Cost for the PMST problem, for 40 nodes (node probability is 0.1), and edge connectivity probability 0.3.

Test Cases	Algorithms			Percent Improvement
	Greedy	Det. (GA)	Cycle basis (GA)	
I	182.4	132.0*	160.0	27.6
II	197.3	167.5*	180.0	15.1
III	194.9	153.9*	155.7	21.0
IV	186.9	162.6	161.3*	13.7
V	186.9	172.5*	231.3	7.7
VI	171.9	143.9*	160.9	16.3
VII	254.3	170.8*	174.1	32.8
VIII	196.2	153.5*	162.4	21.8
IX	172.7*	185.3	232.7	-7.3
X	193.5	148.6*	190.7	23.2

Table 14: The Expected Active Cost for the PMST problem, for 40 nodes (node probability is 0.1), and edge connectivity probability 0.4.

Test Cases	Algorithms			Percent Improvement
	Greedy	Det. (GA)	Cycle basis (GA)	
I	173.4	125.9*	178.8	27.4
II	191.1	155.6*	192.7	18.6
III	183.3	155.6	145.0*	20.9
IV	160.8	146.9*	149.3	8.6
V	180.1	151.8*	189.4	15.7
VI	167.5	132.4*	151.2	20.9
VII	221.3	154.1*	203.9	30.4
VIII	198.2	145.4*	161.9	26.6
IX	177.5	141.9*	162.8	20.0
X	184.9	149.8*	162.8	18.9

Table 15: The Expected Active Cost for the PMST problem, for 40 nodes (node probability is 0.1), and edge connectivity probability 0.5.