Proceedings of the 1996 ACM/SIGAPP Symposium on Applied Computing, February 18-20, 1996, Philadelphia, PA., Pages 299-304, ACM Press.

SOLVING THE SUBSET INTERCONNECTION DESIGN PROBLEM USING GENETIC ALGORITHMS *

Faris N. AbuAli
Computer Science Department
Yarmouk University
Irbid, Jordan

Roger L. Wainwright
Dept. of Mathematical and Computer Sciences
The University of Tulsa
600 South College Avenue
Tulsa, OK 74104-3189, USA
rogerw@penguin.mcs.utulsa.edu
http://www.utulsa.edu

Dale A. Schoenefeld
Dept. of Mathematical and Computer Sciences
The University of Tulsa
600 South College Avenue
Tulsa, OK 74104-3189, USA
dschoen@euler.mcs.utulsa.edu
http://www.utulsa.edu

Key Words: Genetic Algorithms, Interconnection Design Problem, Minimum Spanning Tree

ABSTRACT

The genetic algorithm (GA) heuristic is used to find near optimal solutions for an NP-complete variation of the minimum spanning tree problem. Given a set of vertices V, a cost function $c: V \times V \longrightarrow \Re^+$, and a collection of subsets of V, $\{X_1, ..., X_m\}$, a graph G with vertex set V is called *feasible* if every X_i induces a connected subgraph of G. The minimum subset interconnection design problem is to find a feasible graph with a minimum cost. A GA solution is compared to two recently developed approximate solution techniques, and is shown to produce superior results.

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

INTRODUCTION

The minimum subset interconnection design problem is defined as follows. Given a set of vertices V, a cost function $c: V \times V \longrightarrow \Re^+$, and a collection of subsets of V, $\{X_1, ..., X_m\}$, a graph G with vertex set V is called *feasible* if every X_i induces a connected subgraph of G. This means that for any two elements u and v in X_i there is some u-v path in G consisting only of vertices of X_i. The minimum subset interconnection design problem is to find a feasible graph with a minimum cost. In this paper we investigate the genetic algorithm (GA) as a heuristic technique for finding optimal or near optimal solutions to this problem.

A wide range of applications can be thought of as interconnection design problems. An example is the design of vacuum systems as presented by Du and Miller[11]. Another example relates to establishing communications networks where some subset of sites needs to be internally connected. Prisner [15] gives an excellent example of communicating networks between employees.

The Interconnection design problem is NP-complete [11]. Special cases of the problem that are not NP-complete are detailed in[11, 15] and summarized below:

1. When m, the number of given subsets, is equal to 1, the problem reduces to the classical minimum spanning tree problem.

2. *K(u,v)*, for $u, v \in V$ and $u \neq v$, is the number of subsets, X_i, containing both vertices u and v. When the maximum value of *K(u,v)* over all $u, v \in V$ is equal to 1, the problem is solvable in polynomial time.

3. Let P = the maximum value of $|X_i|, i = 1, ..., m$, where $|X_i|$ is the cardinality of X_i. When P=2, the problem is solvable in polynomial time. If $P \geq 3$ the problem becomes NP-complete.

Prisner [15] recently developed two algorithms for approximately solving the subset interconnection design problem. His first algorithm begins with an edgeless graph $G(V, \Phi)$, and adds new edges consecutively. Edges are chosen based on the total cost-benefit ratio. At every stage of the graph G, the edge v-w with maximum $u(vw, G)/c(vw)$ is chosen until this maximum quotient equals zero. The total benefit of edge v-w, denoted by $u(vw, G)$, is given by $\sum_{i=1}^{m} u_i(vw, G)$, where $u_i(vw, G) = 0$ if X_i does not contain both v and w or if both v and w lie in some common connected component of the actual graph $G[X_i]$, and otherwise, $u_i(vw, G) = 1$. The cost of edge v-w is denoted by $c(vw)$. We will refer to Prisner's first algorithm as algorithm P1. The time-complexity of algorithm P1 is $O(n^4 + mn^2)$, where n is the number of vertices.

Prisner's second algorithm also constructs the edges, one at a time [15]. First, the algorithm computes the function *K(u,v)* and the "modified costs" $c(uv)/K(u, v)$ for $u, v \in V$ and $u \neq v$. Next, the algorithm finds a (c/K)-minimum spanning tree of X_1. Then, it completes the already existing edges in X_2, to obtain a connected spanning subgraph. This is repeated for X_3, and so on. We will refer to Prisner's second algorithm as algorithm P2. The time-complexity of algorithm P2 is $O(n^2(\log n + m\alpha(n^2, n)))$, where α denotes the inverse of the *Ackerman function*. According to Prisner [15], algorithm P2 seems to be faster than algorithm P1, however algorithm P1 generally gives better results.

For example, in Figure 1, four feasible graphs are shown for a set of 10 vertices and a collection of 10 subsets. The cost matrix and the subsets are shown in Figure 1(a) and Figure 1(b), respectively. Solutions shown in Figure 1(c) and Figure 1(d) were produced using our implementation of Prisner's Algorithms, P1 and P2, respectively. Figure 1(e) and Figure 1(f) show superior solutions from our genetic algorithm, which we describe below.

GENETIC ALGORITHMS

Genetic Algorithms (GA) are based on the principles of natural genetics and survival of the fittest. Genetic algorithms search for solutions by emulating biological selection and reproduction. In a GA the parameters of the model to be optimized are encoded into a finite length string, usually a string of bits. Each parameter is represented by a portion of the string. The string is called a chromosome, and each bit is called a gene. Each string is given a measure of "fitness" by a fitness function, sometimes called the objective or evaluation function. The fitness of a chromosome determines its ability to survive and reproduce offspring. The "least fit" or weakest chromosomes of the population are displaced by more fit chromosomes. Genetic Algorithms are applicable to a wide variety of problems. In particular, genetic algorithms have been very successful in obtaining near-optimal solutions to many different combinatorial optimization problems [2, 3, 4, 6, 7, 9, 14, 19].

Genetic algorithm packages for a single processor have been available for several years. The research reported here made use of LibGA [5], a GA package developed in house. Davis, Goldberg and Rawlins provide an excellent in depth study of genetic algorithms [6, 7, 12, 16]. It is assumed the reader is generally familiar with the fundamentals of genetic algorithms.

PROBLEM ENCODING AND EVALUATION FUNCTION

A graph G=(V,E) can be represented by an adjacency matrix A, with entries that are either 0 or 1. The value of a_{ij} is 1 if there is an edge between v_i and v_j, otherwise a_{ij} is 0. Our chromosome representation of G is a bit-string corresponding to the upper triangular part of the adjacency matrix in row major order. The length of the chromosome is fixed at $n(n-1)/2$, where n is the number of vertices in the graph.

To evaluate the fitness of a chromosome, the graph the chromosome represents is checked for *feasibility* with respect to all of the subsets X_i, $1 \leq i \leq m$, and then the cost of the edges is summed. One difficult feature of the subset interconnection design problem is that the *feasibility* constraint is a *hard* constraint rather than a *soft* constraint. That is, breaking this constraint renders a solution infeasible. Several researchers have developed solutions for the infeasible chromosome problem. The chromosome repair technique used by Liepins *et al.* [13] and later by Davis *et al.* [8] is one example. Sekharan *et al.* [18] developed a technique where the infeasible chromosomes are kept in a separate pool. He probabilistically "mated" chromosomes from the feasible pool with chromosomes from the infeasible pool.

Our algorithm uses the chromosome repair technique when the chromosome is *infeasible*. This algorithm is described below. The following steps are performed until the chromosome is *feasible*.

1. Select a subset X_i for $1 \leq i \leq m$.
2. While the chromosome is *infeasible* with respect to subset X_i do steps 3-7.
3. Select a vertex $u \in X_i$ at random.
4. Find the closure C of u in G with respect to X_i. The closure C of u in G with respect to X_i is the set of vertices in X_i that can be reached in G from u using only vertices in X_i.
5. Select a vertex $v \in X_i - C$ at random.
6. Add the edge u-v to the graph G.
7. Add the cost of the edge u-v to the cost of the graph.

We developed three strategies for the selection of the subsets described in step 1 above:

a) Sequential selection: for i=1,..., m select subset X_i.

b) Sorted selection: sort the subsets in increasing order according to cardinality and then do sequential selection.

c) Random selection: randomly pick a permutation P, of the m subsets and then select subset X_i where i=$P[j]$ for $1 \leq j \leq m$.

Notice that the while loop (steps 3-7) can be rather expensive. However, this allows the algorithm to consider a larger number of potential solutions.

TEST PROBLEMS AND RESULTS

The generational genetic algorithm model was executed using four combinations of single point and uniform crossovers, each with fixed and adaptive mutation rates. The crossover rate is 0.6. The mutation rate is 0.1 and the selection strategy for the generational GA was roulette (fitness proportionate).

Table 1 shows the results of running algorithms P1 and P2, as developed by Prisner[15], and our genetic algorithm for 10 different random distributions of vertices and subsets. The number of vertices is 30 and the number of subsets is 140. For all 10 data sets, one random distribution of vertices is generated, and the Euclidean distance between the vertices is used as the cost of connecting the vertices. Then, for each data set, a random distribution of subsets was produced. The size of each subset was randomly selected in the range from 2 to the number of vertices.

In each of the 10 data sets tested, the generational genetic algorithm with uniform crossover and adaptive mutation rate outperformed algorithms P1 and P2. Notice that the GA using adaptive mutation consistently performed better than fixed mutation regardless of the crossover method used.

Table 2 shows the results for a total of 34 runs on eight different sets of vertices. The cardinalities of the eight sets of vertices range from n=5 to n=40 in increments of 5. For each set V of vertices, four different collections of subsets of V were generated. The first collection of subsets contained 1*$|V|$ subsets of V. Similarly, the second, third and fourth collections of subsets of V contained 2*$|V|$, 3*$|V|$, and 4*$|V|$ subsets of V, respectively. In addition, for the case where $|V|$=35, a collection of 8*35=280 subsets of V was generated and tested. Similarly, for the case where $|V|$=40, a collection of 8*40=320 subsets of V was generated and tested. In each case, the size of each subset was randomly generated in the range from 2 to $|V|$, the number of vertices.

In all 34 cases tested, at least one of our genetic algorithms outperformed algorithm P2. In 28 of the 34 cases, at least one of our genetic algorithms produced the same or better results than algorithm P1. In the remaining 6 cases, all occurring when n is large and the number of subsets is relatively small, the GA did not do as well as algorithm P1. On the other hand, when the number of subsets is increased, the GA consistently produced better results.

CONCLUSION

To the authors' knowledge a genetic algorithm implementation for the subset interconnection design problem has not been implemented before. Prisner [15] has recently developed two near optimal algorithms for this problem. We developed a genetic algorithm for the subset interconnection design problem and tested it on a variety of problems comparing our results to Prisner's algorithms. Our GA exhibited superior results in nearly all of the cases tested. We determined that our best GA to solve this problem uses uniform crossover with adaptive mutation. Our three selection strategies to sequence the selection of the subsets in order to convert an infeasible chromosome to a feasible chromosome were indistinguishable in their results. No one algorithm appeared to be any better than the others. Prisner's algorithms run in the order of minutes and our GA runs in the order of hours. Because the GA yields superior results in most cases, we recommend that researchers run the GA as well as Prisner's algorithms to obtain the best result possible.

Algorithm		Random data sets									
		I	II	III	IV	V	VI	VII	VIII	IX	X
Algorithm P1		3706	4452	4247	4093	4010	4179	4212	4216	3621	4233
Algorithm P2		4703	5162	5206	4817	4880	5018	4820	4858	4337	5069
Genetic Alg.											
Crossover	Mut.										
Single pt	Adapt	3749	4188*	4482	3846*	4070	3874	4178	4186	3393*	4428
Uniform	Adapt	3623*	4190	4104*	3851	3749*	3826*	3968*	4160*	3559	4003*
Single pt	Fixed	4107	4628	4755	4624	4608	4855	4771	4757	4450	4994
Uniform	Fixed	3986	4535	4502	4089	4156	4251	4627	4340	3787	4481

Table 1: Subset Interconnection Design problem for 30 nodes and 140 subsets. A "*" indicates the best.

Acknowledgements

This research has been supported by OCAST Grant AR2-004. The authors would like to thank Referee No. 4 for his thorough reading of the manuscript, which resulted in numerous improvements and clarifications. We gratefully acknowledge the support of Sun Microsystems, Inc.

References

[1] Hal Berghel, Ed Deaton, George Hedrick, David Roach, and Roger Wainwright, editors. *Proceedings of the 1992 ACM/SIGAPP Symposium on Applied Computing*, New York, 1992. ACM Press.

[2] J. L. Blanton and R. L. Wainwright. Vehicle routing with time windows using genetic algorithms. In William A. Coberly, editor, *Proceedings of the Sixth Oklahoma Symposium on Artificial Intelligence*, pages 242–251, Tulsa, Oklahoma, November 1992.

[3] D. E. Brown, C. L. Huntley, and A. R. Spillane. A parallel genetic heuristic for the quadratic assignment problem. In Schaffer [17].

[4] A. L. Corcoran and R. L. Wainwright. A genetic algorithm for packing in three dimensions. In Berghel et al. [1], pages 1021–1030.

[5] A. L. Corcoran and R. L. Wainwright. LibGA: A user-friendly workbench for order-based genetic algorithm research. In Deaton et al. [10], pages 111–118.

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

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

[8] Lawrence Davis, David Orvosh, Anthony Cox, and Yuping Qiu. A genetic algorithm for survivable network design. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, Urbana-Champaign, Illinois, July 1993. Morgan Kaufmann.

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

[10] Ed Deaton, K. M. George, Hal Berghel, and George Hedrick, editors. *Proceedings of the 1993 ACM/SIGAPP Symposium on Applied Computing*, New York, 1993. ACM Press.

[11] D-Z. Du and Z. Miller. Matroids and subset interconnection design. *SIAM J. Discrete Math*, 1(4):416–424, 1988.

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

[13] G. E. Liepins, M. R. Hilliard, J. Richardson, and M. Palmer. Genetic algorithm applications to set covering and traveling salesman problems. In Brown, editor, *OR/AI: The Integration of Problem Solving Strategies*. 1990.

[14] P. P. Mutalik, L. R. Knight, J. L. Blanton, and R. L. Wainwright. Solving combinatorial optimization problems using parallel simulated annealing and parallel genetic algorithms. In Berghel et al. [1], pages 1031–1038.

[15] Erich Prisner. Two algorithms for the subset interconnection design problem. *Networks*, 22:385–395, 1992.

[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] D. Ansa Sekharan and Roger L. Wainwright. Manipulating subpopulations of feasible and infeasible solutions in genetic algorithms. In Deaton et al. [10], pages 118–125.

Number of Vertices	Number of Subsets	Algorithm P1	Algorithm P2	Genetic Algorithm		
				Random	Sequential	Sorted
5	5	196*	196*	196*	196*	196*
	10	241*	241*	241*	241*	241*
	15	281*	281*	281*	281*	281*
	20	332*	334	332*	332*	332*
10	10	534	525	486*	486*	503
	20	799	887	725*	736	725*
	30	954	1103	904	896*	896*
	40	976	1165	925*	925*	925*
15	15	785	766	678*	678*	703
	30	1106	1201	955	964	946*
	45	1417	1600	1295	1285	1269*
	60	1651	1898	1525*	1566	1528
20	20	1396	1480	1239	1177*	1265
	40	1901	2032	1792	1720*	1853
	60	2406	2811	2187	2309	2155*
	80	2891	3169	2650	2580*	2620
25	25	1590	1819	1590	1615	1524*
	50	2141	2535	2031*	2188	2098
	75	2493	3036	2504	2460	2393*
	100	2848	3706	2770	2696*	2711
30	30	2056	2303	2094	2115	1905*
	60	2891	3268	2627*	2639	2671
	90	3382	3712	3182	2971*	3073
	120	3891	4883	3509*	3630	3569
35	35	2294*	2869	2423	2485	2418
	70	3202*	3853	3324	3264	3265
	105	3889	4887	3894	4091	3807*
	140	4346*	5178	4401	4444	4577
	280	6013	8421	5906*	6028	6020
40	40	2558*	2984	2930	3294	2978
	80	3461*	4643	3952	3887	3810
	120	4246*	5526	4851	4759	4773
	160	5143	6413	4908*	5417	5113
	320	7121	10189	7143	6890*	6941

Table 2: Subset Interconnection Design problem using Genetic Algorithms with uniform crossover, crossover rate 0.6, and adaptive mutation. A "*" indicates the best.

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

Cost matrix

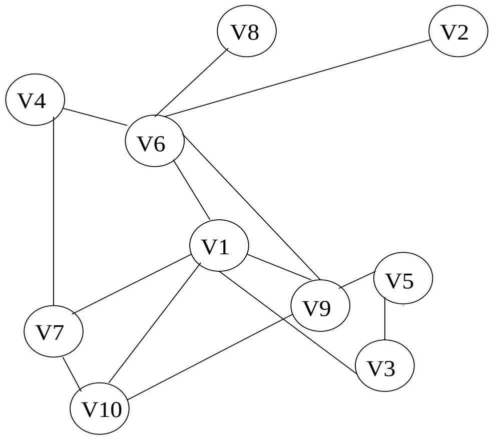
	1	2	3	4	5	6	7	8	9	10
1	*	71	45	46	43	26	43	49	31	54
2		*	73	97	59	66	114	56	69	125
3			*	91	14	68	74	84	14	75
4				*	88	32	46	47	77	63
5					*	61	78	74	16	82
6						*	56	24	54	71
7							*	80	62	17
8								*	71	96
9									*	66
10										*

(a)

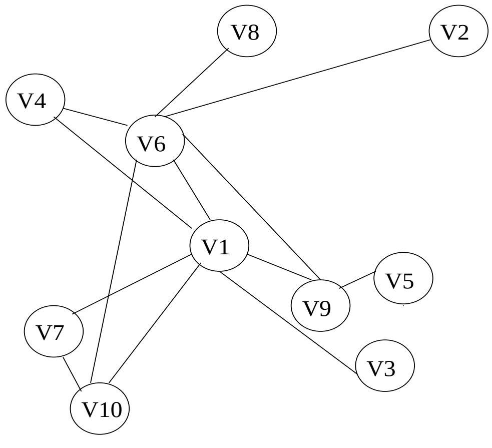
Subsets of V

i	X_i						
1	1	4	5	6	9	10	
2	1	6	10				
3	2	4	5	6	7	9	10
4	1	2	5	6	8	9	10
5	1	3	6	8			
6	1	2	4	6	8	9	10
7	1	3	7				
8	1	10					
9	6	9	10				
10	1	3	4	5	7	9	

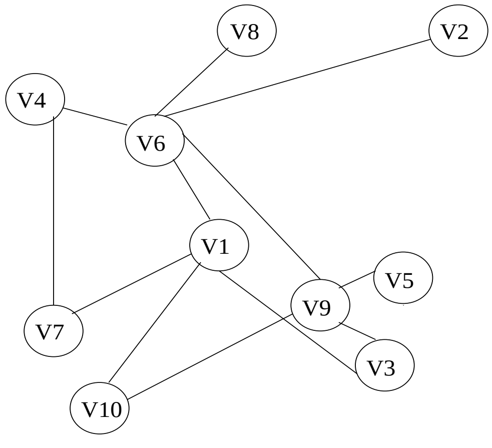
(b)



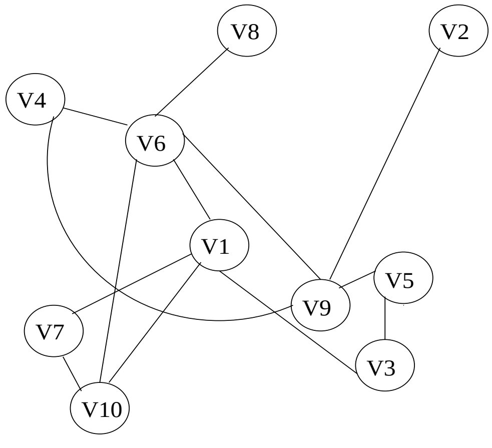
(c)



(d)



(e)



(f)

Figure 1: Subset Interconnection Design problem for 10 nodes and 10 subsets. (a) The cost matrix, (b) The subsets of V, (c) Algorithm P1 with a cost of 534, (d) Algorithm P2 with a cost of 525, (e) Genetic Algorithm with uniform crossover with a cost of 486, (f) Genetic Algorithm with single point crossover with a cost of 510.