# A Genetic Algorithm for the Point to Multipoint Routing Problem with Varying Number of Requests

Liming Zhu, Roger L. Wainwright, and Dale A. Schoenefeld
*Mathematical and Computer Sciences Department*
*The University of Tulsa, 600 South College Avenue*
*Tulsa, OK 74104-3189, USA [rogerw,dschoen]@euler.mcs.utulsa.edu*

## Abstract

***Message scheduling or call request scheduling is the process of finding optimal routing for a set of circuit connection requests through a communications network. Each request has a single source with one or more destinations. Furthermore, different requests may have different source and different destination nodes. Finding optimal routing for a set of requests is called the Point to Multipoint Routing Problem (PMRP). The Current practice in solving the PMRP is to consider each point to multipoint request as a collection of point to point requests, and then solve each point to point request. This is very costly and generally produces poor results. This paper presents an algorithm for the Point to Multipoint Routing Problem that uses a genetic algorithm and a heuristic Steiner tree algorithm. Our genetic algorithm allows the scheduler to find an optimal or near-optimal path through the network for each request. In our algorithm each request is treated as a whole and not as a collection of point to point requests. Furthermore, given a set of Point to Multipoint Routing requests, our algorithm considers various subsets sizes of the original set of requests and produces an optimal or near-optimal ordering of the requests for the specified subset size. We ran our PMRP algorithm on several test cases with excellent results.***

## 1. Introduction

Message scheduling or call request scheduling is the process of finding optimal routing for a set of circuit connection requests through a communications network. Each request has a single source with one or more destinations. Furthermore, different requests may have different source and different destination nodes. Finding optimal routing for a set of requests is called the Point to Multipoint Routing Problem (PMRP). Current practice takes each point to multipoint request and treats it as a collection of point to point requests. This method incurs considerable costs as it is common for multiple copies of the request to appear on the same link or on parallel links. This paper presents an algorithm for the Point to Multipoint Routing Problem that uses a genetic algorithm and a heuristic Steiner tree algorithm. Using both of these techniques in conjunction allows the scheduler to find an optimal or near-optimal path through the network for each request without having to consider the request as several point to point messages. In a communications network each edge has a cost (of traversal) associated with it. In our algorithm the Steiner tree heuristic is used to find a minimal cost or near minimal cost tree in the network for a given request. Hence, each request is treated as a whole. The genetic algorithm is used to determine the optimal or near optimal ordering of the requests through the network.

## 2. Point to Point Routing Problem

Consider a scheduler for a telecommunications network that schedules multiple simultaneous requests through a network. Each request has a single source node and a single destination node. The source and destination nodes may be any nodes in the network. The entire signal represented by a request must be routed along the same path. Each link in the network has an assigned capacity and a cost for using the link. This technique incurs considerable costs as it is common for multiple copies of the signal to appear on the same link or on parallel links. This is called the Point To Point Routing Problem (PPRP), since each request routes a signal from one point to another point in the network. The PPRP is known to be NP-complete [7].

Cox ***et al.*** have studied the PPRP extensively [7]. Cox describes a scheduler for a circuit switching telecommunications network on which call requests arrive randomly at any of the nodes of the network. In his model, each call request is specified by six attributes: source node, destination node, requested start time, requested duration, bandwidth requirement or capacity, and priority class. For each request, the scheduler assigns a sin-

0-7803-4869-9/98 $10.0001998 IEEE

gle dedicated path through the network from the source node to the destination node during the entire requested time interval. That is, the connection is ***circuit switched.*** The scheduler must take into consideration the capacities of requests that might concurrently use a link. A feasible schedule for a set of requests is one for which the sum of the capacities of the requests assigned to a link, at any given time, is never greater than the link's capacity. If a call request cannot be accommodated by a particular schedule, the request is said to be blocked and the schedule is infeasible.

The scheduler calculates the cost of each feasible schedule. The cost is the sum of the network edge costs of carrying scheduled requests through their assigned paths. For an infeasible schedule, one generally includes penalties associated with requests that are blocked. The scheduler's task is to choose a set of paths for all call requests that minimizes total cost while taking into account the time varying nature of the call requests. Cox addresses this problem through the use of a statistical approach that makes rapid initial assignments of incoming call requests to paths, based upon statistical information about the current network traffic. He then uses evolutionary programming techniques to find alternate routing assignments that are more nearly optimal or that can accommodate additional requests [7].

## 3. Point to Multipoint Routing Problem

Little research has been devoted to the Point to Multipoint Routing Problem (PMRP) where each request originates at a single source node and may have several destination nodes. The difference between the PPRP and the PMRP is that several destination nodes may occur for a given request in the case of the PMRP.

Our investigation into the PMRP is a result of our collaboration with LDDS WorldCom (formally WilTel), a locally based internationally prominent telecommunications vendor. It is routine for a telecommunications company to transmit the same signal to many different destinations over a telecommunications network backbone. Thus an optimal or near optimal algorithm to solve the PMRP is extremely practical to this rapidly expanding and highly competitive business.

Our technique for the PMRP finds a circuit switched minimal spanning tree (MST) assignment for each of several point to multipoint requests so that the largest possible number of requests can be accommodated or such that the routings incur the least possible cost. The MST determines a point to multipoint circuit in the network. Instead of several copies of the signal on a single link, a single copy would be sent. When this signal reaches an intermediate node that is linked to two or more destination nodes or other intermediate nodes, the intermediate node duplicates the signal as determines by the MST. Compared to treating each point to multipoint request as several point to point requests, our method reduces the network traffic caused by each request, and allows for more messages to be sent simultaneously.

## 4. Genetic Algorithms

Genetic algorithms (GA) search for solutions by emulating biological selection and reproduction. Genetic algorithms are based on the principles of natural genetics and survival of the fittest. The genetic algorithm operates on a fixed size population of chromosomes. A chromosome is a string of genes that represents an encoding of a candidate solution. Associated with each chromosome is a fitness value. The fitness of a chromosome corresponds to its ability to survive and reproduce offspring. Several researchers have investigated the benefits of solving combinatorial problems using genetic algorithms. Davis [8], Goldberg [10], Whitley [12], Belew and Vose [3], Chambers [4] and Mitchell [11] provide excellent in-depth studies of genetic algorithms. This research uses an order based genetic algorithm where a chromosome is a permutation of integers. The genetic algorithm implementation used in this research is LibGA [6].

## 5. Steiner Tree Problem in a Graph

The Steiner Problem in a Graph (SPG) is a classic combinatorial optimization problem. The SPG has been shown to be NP-complete. Given a graph and a required subset of vertices from the graph, $W$, any subtree containing all vertices of $W$, and possibly additional vertices from the graph, is called a Steiner tree. A minimal Steiner tree is a Steiner tree of minimal cost. Thus, given a weighted graph G = $(V,E)$, and a subset $W \subseteq V$, the SPG is to find a minimum cost spanning tree $T = (V', E')$ with $W \subseteq V' \subseteq V$ and $E' \subseteq E$. An example Steiner tree where the required vertices are W = $\{A, E, I\}$ is shown in Figure 1. There are several techniques for constructing near-optimal Steiner trees in the literature [1, 9].

## 6. Methodology and Implementation

We consider only the special case of the PMRP where all requests have the same start time, duration, and priority. The first step to finding a solution is to model the network as a graph. Each edge is assigned a cost and a capacity. Each request is identified by the source node, the destination node(s), and the capacity of the request.

As an example of the PMRP, consider a communications network consisting of 8 nodes as shown in Figure 2. The network can be modeled as a graph consisting of eight vertices representing the eight nodes and eleven

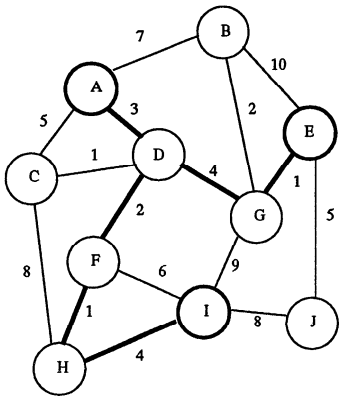


Figure 1: An SPG Example, W = *{A, E, I}*. The bold lines indicate the Minimum Cost Steiner Tree

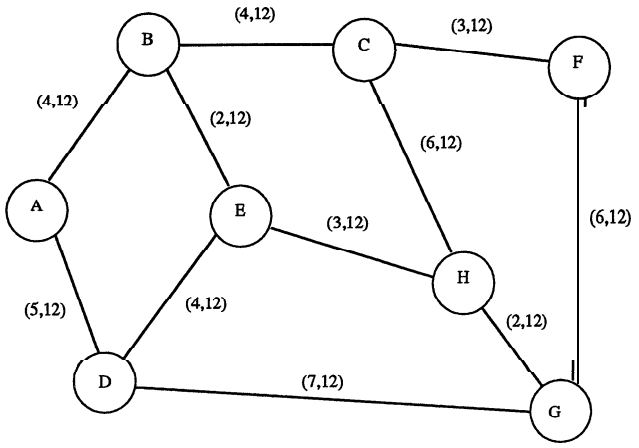


Figure 2: A Communications Network with (Cost, Capacity) on each edge

edges representing the available transmission lines between them. Each edge has associated with it a cost for use and an available capacity. For simplicity, in this example the capacity of every edge is the same, **12.** However, edges could have different capacities. Note the cost is per unit of capacity. For example, using one unit of capacity on link EH in Figure 2 costs three, and using its full capacity costs 36.

Now consider four message requests. Each request contains a source node, one or more destination nodes, and a message size or capacity. An example set of requests is shown in Table 1. To route these four messages using a point to multipoint routing technique, the scheduler must find, for each message, a tree in the network that contains the source and each destination. In this case, the scheduler must find four different trees (one tree for each source and its destinations in Table 1). For the routing to be optimal, the trees must cost as little as possible and no request can be blocked. The requests can be routed in any order. Note the order one chooses to route the requests may greatly effect the solution.

Table 1: Four Example Message Requests

| Request | Source | Destination(s) | Capacity |
|---|---|---|---|
| RI | G | A, C, F | 4 |
| R2 | A | E, F, G | 6 |
| R3 | H | B, D, F, G | 3 |
| R4 | C | E | 4 |

In our previous work on the PMRP [5], requests were given a priority based on the order in which they arrived. For example, suppose a routing was determined for the first four requests that arrived, but when request five was added, no permutation of the five requests produced a solution. Thus a solution was obtained routing just requests 1 through 4. Any later requests could not be considered. That is, if later requests six and seven showed up they could not be considered due to the priority constraint. This constraint limited the solution space and made the PMRP much easier to solve. However, this model does not reflect the real world business approach to this problem. Our current implementation of the PMRP presented in this paper does not assign priorities to the requests as they come in, thus mimicing industry practices. Hence, in the above example our new algorithm might be able to find a better solution (considering all seven requests) routing say all of the requests but request three. To further clarify this problem from a simple business perspective, consider a communications network company selling bandwidth on their network for a specified half hour period during the next business day. Requests come in periodically during the current business day, and a commitment is not required until 5:00 p.m. Thus, no priority is assigned to the requests, and servicing the maximum number of requests becomes the goal. Furthermore, an algorithm (like the one we are presenting) is running throughout the day incorporating new requests as they arrive. The best solution by the close of the business day is used, and commitments are made at that time to the customers for the next day's broadcast.

In our GA, a chromosome is a permutation of the number of requests, $n$ that have arrived thus far. For example, suppose $n$ = 20, then the chromosome is a permutation of the integers 1..20. In practice, it is rarely possible to schedule all of the requests, regardless of the order we use. Thus, some subset $k$ of the $n$ requests is selected ($1 \leq k \leq n$), and the GA determines the best order to route $k$ requests from among $n$. This is accomplished as follows. First a fixed value for $k$ is determined,

say, 12. Then the GA is run on a population of chromosomes each of size ***n***. However, the evaluation function for a given chromosome is determined by routing only the first ***k*** requests in the chromosome. The last ***n-k*** alleles in each chromosome are ignored in the evaluation. Note all of the ***n*** alleles participate in crossover and mutation thus mixing up the requests. When our algorithm finishes we have the "best" routing for ***k*** requests. Since the goal is to route as many requests as possible, we rerun the GA with larger ***k*** values until a solution cannot be found or $k=n$. The best chromosome for the largest ***k*** is the solution to the given problem. Note in our model we assume all requests yield the same profit thus maximizing ***k*** is the goal. It is a simple matter to assign different profit margins for each request and alter the goal to maximize profit.

The genetic algorithm initializes the population with random permutations. During execution, the genetic algorithm calls an evaluation function to assign fitness values to the chromosomes. Each message is routed through the network. For a given chromosome, a non-GA (deterministic) method selects a Steiner tree for each request in the order in which the requests appear in the chromosome,

To route a given request, the algorithm first inspects the edges of the entire graph for any edges with a capacity less than the capacity of the request and temporarily changes the cost on each such edge to an artificially high cost value. Then, a near-optimal or optimal Steiner tree is found for the current request. After the Steiner tree has been found, the capacities of the edges used in the Steiner tree are decremented by the capacity of the request. This is repeated for each request, in the order they are found in the chromosome, until every request has been routed. The evaluation function then assigns the fitness value to the chromosome. The fitness value of a chromosome is the cost of using the network. That is, the sum of the capacity used on each link times its cost. The object is to minimize cost for the maximum subset size, I.

Consider the network in Figure 2 and the requests in Table 1. Assume that the genetic algorithm is ready to evaluate a chromosome with an allele pattern of 1,2,3,4. In this case $k = n = 4$. First, a Steiner tree is found to route R1 (request 1). Then, Steiner trees will be determined for R2, R3, and R4. For each request the algorithm verifies that capacities on all edges are not less than the capacity of the request. Consider the route for Rl. The algorithm finds a Steiner tree for Rl. The result is shown in Figure 3. The Steiner tree is illustrated with the dashed lines and the changes to the available capacities are shown. The algorithm now finds a routing for R2. Because the capacity of R2 is six and all

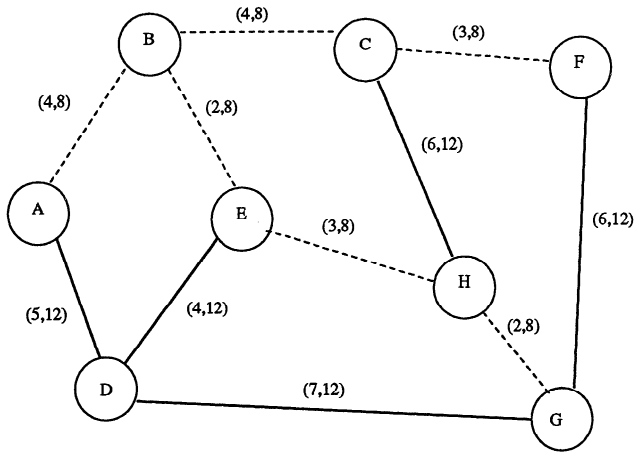


Figure 3: The Steiner Tree for Rl (dashed lines) and the changes in available capacities. Steiner Tree = 18.

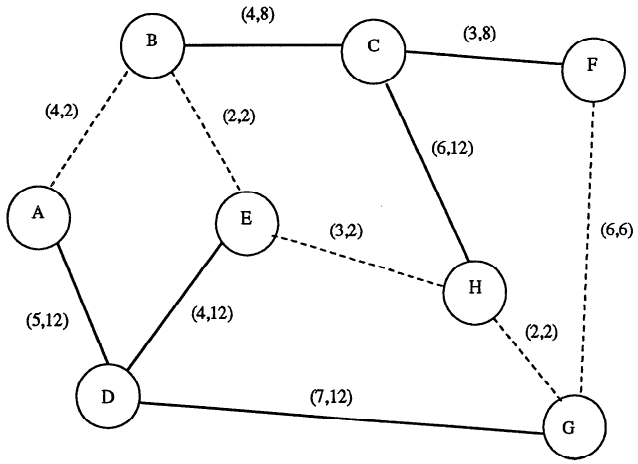


Figure 4: The Steiner Tree for R2 (dashed lines) and the changes in available capacities. Steiner Tree = 17.

links in the network still have a capacity of at least six, all links are declared usable. The resulting Steiner tree is shown in Figure 4.

Next, R3 is routed. Because the capacity of R3 is greater than the remaining capacities on edges AB, BE, EH and GH, those edges have their costs changed temporarily to an extremely high value to minimize the possibility of being used by the Steiner tree algorithm. After the Steiner tree function finishes execution, the cost values of the altered edges are returned to their original value. Figure 5. shows the Steiner tree and resulting capacities after routing R3. Finally, R4 is routed. Notice by inspection of Figure 5 there is no way to route four units from C to E. Thus R4 is "blocked".

For comparison, consider the chromosome (4,3,2,1) routing R4, R3, R2, and finally Rl through the initial network shown in Figure 2. A summary follows: R4 is

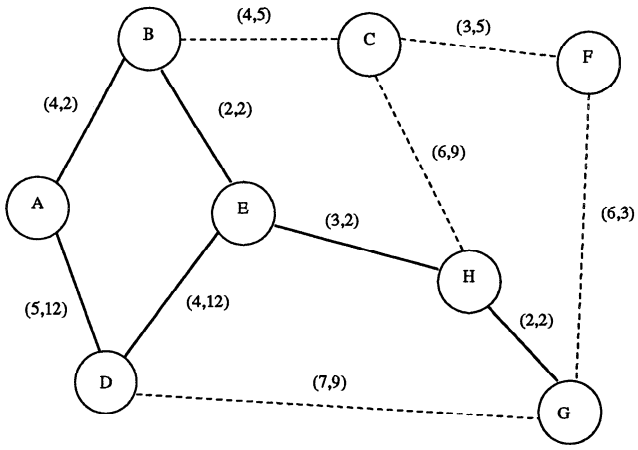


Figure 5: The Steiner Tree for R3 (dashed lines) and the changes in available capacities. Steiner Tree = 26.

routed by using the Steiner tree BC, DE (cost = 6). R3 is routed by using the Steiner tree BE, DE, EH, HG, FG (cost = 17). R2 is routed by using the Steiner tree AD, DE, EH, GH, FG (cost = 20). Rl is routed by using the Steiner tree DG, AD, AB, BC, CF (cost = 23). In this ordering of the requests none of the requests are blocked and a solution is generated. Figure 6 shows the resulting network and capacities after routing R4, R3, R2, Rl. In this case the chromosome (4,3,2,1) yields a feasible solution with a fitness value of 6+17+23+20=66.

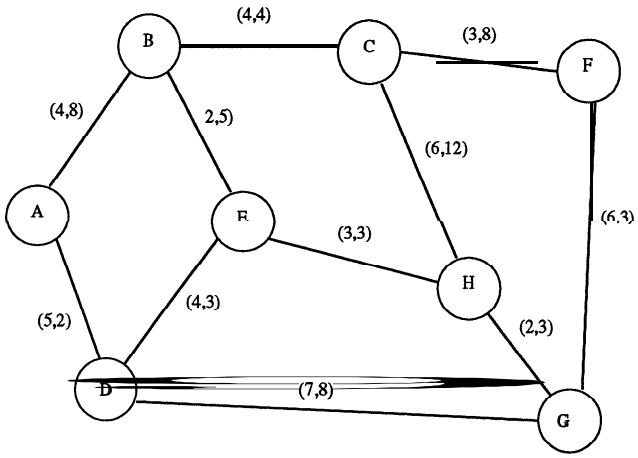


Figure 6: Resulting network and capacities after routing R4, R3, R2, Rl

In this research, a generational genetic algorithm was used. The pool size was 100. The initial population was generated randomly. The PMX crossover method and roulette wheel selection method were used. The GA terminated after 100 iterations or upon convergence. Our program to find Steiner trees was an adaptation of the code developed by Alexander and Robins [1]. It uses the Kou, Markowsky, Berman (KMB) algorithm for solving the Steiner problem in a graph.

## 7. Test Cases

We used test cases with up to 20 requests on a single network of 61 nodes and 133 edges. The data set was obtained by modifying one of the benchmark *steinb* data sets that are used for the Graphical Steiner Tree Problem [2]. Each of the 133 edges was assigned a random cost from one to ten and a fixed capacity of twelve. Each request contained no more than eight nodes and had a capacity between five and nine. The 20 requests are listed in Table 2. In practice a set of requests is scheduled

Table 2: Test Case Requests

| Request Number | Source Node | Destination Node(s) | Capacity |
|---|---|---|---|
| 1 | 36 | 7,23,25,40 | 8 |
| 2 | 17 | 15,30,31,40,41,46 | 5 |
| 3 | 48 | 3,9 | 9 |
| 4 | 41 | 50,22,27,35,13 | 6 |
| 5 | 2 | 6,14,18,23,27,33,47,49 | 5 |
| 6 | 13 | 28 | 6 |
| 7 | 50 | 5,12,28,31,44,45 | 7 |
| 8 | 24 | 30,29,20 | 5 |
| 9 | 52 | 13,55,22,9 | 4 |
| 10 | 53 | 28,52,13,55,41,14 | 6 |
| 11 | 10 | 31,20,5,40 | 5 |
| 12 | 15 | 30,20,23,22,18 | 4 |
| 13 | 14 | 9,35,16 | 4 |
| 14 | 61 | 15,33,38,20 | 3 |
| 15 | 55 | 4,41,21 | 5 |
| 16 | 14 | 16,43,44,31,9 | 7 |
| 17 | 60 | 28,14 | 2 |
| 18 | 9 | 6,35,30,7,4,31 | 4 |
| 19 | 51 | 54,40,10 | 6 |
| 20 | 51 | 23,10 | 5 |

for the network for a specific time period, then later if another request arrives a new schedule is determined. If another request comes in, a new schedule is determined, and so on until no more requests can be scheduled or the time has arrived to commit to the schedule. In each case the maximum subset of requests is found among all of the requests. In LDDS WorldCom's case, (at the time of this research) their algorithm for scheduling a request is based on a combination of simple point to point routings and hand scheduling based on experience. Our GA implementation automates this process with good results.

## 8. Results

Table 3 contains the resulting ordering of the requests

and their fitness values found by the genetic algorithm for each of the test cases where $k$ = eight to eighteen for this example network. No feasible solution was generated for $k$ = 19 and 20. As a point of reference, the GA implementation for the test case with eight requests took under 10 minutes of CPU time, and the test case with $k$ = 18 took just under one hour running on a Sun Sparc 10 workstation. The authors are not aware of any

Table 3: Results for subsets k = 8 to 18

| Test Case | Number of Requests | Final Ordering | Fitness Value |
|---|---|---|---|
| 1 | 8 | 14,17,9,12,6,13,20,8 | 579 |
| 2 | 9 | 8,12,14,13,9,20,19,6,17 | 705 |
| 3 | 10 | 8,13,14,12,18,20,19,6,9,17 | 833 |
| 4 | 11 | 9,13,15,8,20,12,6,18,19,17, 14 | 968 |
| 5 | 12 | 17,11,19,6,12,13,8,18,14,15, 9,20 | 1148 |
| 6 | 13 | 10,19,9,8,15,13,14,18,12,6, 11,17,20 | 1328 |
| 7 | 14 | 19,6,8,10,13,9,15,4,18,17, 11,14,12,20 | 1519 |
| 8 | 15 | 2,10,11,9,15,19,8,13,18,20, 4,14,17,12,6 | 1785 |
| 9 | 16 | 9,17,19,8,15,11,6,14,5,4,10, 12,20,2,18,13 | 2218 |
| 10 | 17 | 5,2,10,20,9,17,13,16,18,15, 19,4,6,8,14,11,12 | 2513 |
| 11 | 18 | 15,8,9,1,13,5,17,10,4,2,19, 20,14,11,12,16,18,6 | 2971 |

other research for solving this version of the PMRP as it applies to a communication network company. Because there is no previous research similar to this project, there is nothing with which to compare our results. We realize Table 3 says nothing about the efficiency or quality of our algorithm or approach for solving this problem. However, the results exceed our expectations based on our industrial experience of this problem, and give an excellent indication of the potential success of future genetic algorithm implementations of the PMRP problem on an industrial scale.

Furthermore, this research successfully demonstrates the potential for a point to multipoint request scheduler. By using a Steiner tree to describe the routing of a signal from the source to the destinations, a near-optimal routing can be found. By using Steiner tree based routing instead of treating the point to multipoint requests as several point to point requests, there will generally be fewer simultaneous copies of a signal on a link, resulting in less network traffic and a potential for more requests to be sent. We did consider using a GA to determine the "best" Steiner tree for each call request (a GA within a GA implementation). However , we determined that the GA driving the order of the requests was the crucial factor, so a deterministic near-optimal Steiner tree implementation was more than adequate. In addition, by using genetic algorithms to find an optimal or near-optimal ordering of the requests, a series of message requests can be routed concurrently at a reasonable cost blocking fewer requests. The CPU time and resources are well within industrial requirements. The result is a network that can accommodate more requests at a lower cost and offer increased profit for industry compared to current methods used for solving the PMRP.

## Acknowledgements

The authors gratefully acknowledge Michael Alexander and Gabriel Robins for providing code for several Steiner tree algorithms that they developed in [1].

# References

[1] **Michael J. Alexander and Gabriel Robins, "New Performance-Driven FPGA Routing Algorithms", Proceedings of ACM/SIGDA Design Automation Conference, June 1995.**

[2] **J.E. Beasley, " OR-Library: distributing test problems by electronic mail," *J. Opl. Res. Soc.*, *vol.* 41, pp 1069-1072, 1990.**

[3] **Richard K. Belew and Michael D. Vose, editors, *Foundations of Genetic Algorithms* 4, Morgan Kaufman, 1993.**

[4] **Lance Chambers, editor, *Practical Handbook of Genetic Algorithms*, Vol. 1, CRC Press, 1995,**

[5] **H.L. Christensen, R.L. Wainwright and D.A. Schoenefeld, "A Hybrid Algorithm for The Point to Multipoint Routing Problem", Proceedings of the 1997 ACM Symposium on Applied Computing, ACM Press, 1997, pp 263-268.**

[6] **A.L. Corcoran and R.L. Wainwright, "Using LibGA to Develop Genetic Algorithms for Solving Combinatorial Optimization Problems", *Practical Handbook of Genetic Algorithms*, Vol. 1, Lance Chambers, ed., CRC Press, 1995, pp 143 - 172.**

[7] **Louis Anthony Cox, Jr., Lawrence Davis and Yuping Qiu, "Dynamic Anticipatory Routing In Circuit-Switched Telecommunications Networks", *Handbook of Genetic Algorithms*, Lawrence Davis, ed., Van Nostrand Reinhold, 1991, pp 124 - 143.**

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

[9] **Henrik Esbensen, "Finding (Near-) Optimal Steiner Trees in Large Graphs", Proceedings of the Sixth International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, Inc., 1995, pp 485 - 492.**

[10] **D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.**

[11] **Melaine Mitchell, *An Introduction to Genetic Algorithms*, The MIT Press, 1996.**

[12] **L. Darrell Whitley, editor, *Foundations of Genetic Algorithms 2*, Morgan Kaufman, 1993.**