Proceedings of the 1998 ACM/SIGAPP Symposium on Applied Computing (SAC'98), February 28 - March 1, 1998, Atlanta, GA, ACM Press, pp. 318-326.

Finding Rural Postman Tours

Clarissa Cook
Math and Computer Sciences
The University of Tulsa
Tulsa, OK, USA

Dale A. Schoenefeld
Math and Computer Sciences
The University of Tulsa
Tulsa, OK, USA
schoend@utulsa.edu

Roger L. Wainwright
Math and Computer Sciences
The University of Tulsa
Tulsa, OK, USA
rogerw@utulsa.edu

Keywords: rural postman tours, routing, combinatorial optimization, graph theory, genetic algorithms.

Abstract: This work defines valuable terminology and develops significant techniques for obtaining optimal or near optimal solutions to the Rural Postman Problem.

1 Introduction

This work defines valuable terminology and develops significant techniques for obtaining optimal or near optimal solutions to the Rural Postman Problem. The *Rural Postman Problem* (RPP) assumes an undirected, connected graph, $G(V, E, R, \omega : E \rightarrow Z_0^+)$. The function ω assigns edge lengths and R is an arbitrary subset of E. The objective of the RPP is to find a minimum length tour in G that includes each edge of R at least once. R is referred to as the *required subset of edges*.

Many practical routing problems involve finding paths that traverse a set of arcs in a graph. Road sanding trucks may service road segments belonging to their governmental jurisdiction, while using roads belonging to another jurisdiction for access. The challenge is to find an efficient route.

The difference between the *Rural Postman Problem* (RPP) and the classical *Chinese Postman Problem* (CPP) is that the desired tour in the CPP is required to include all edges of G whereas the desired tour in the RPP is only required to include the edges of R. Finding the optimal tour that solves the RPP is a harder problem than finding the optimal tour that solves the CPP. The RPP is NP-complete. The undirected CPP can be solved in polynomial time.

The classical CPP can be easily solved if all vertices of G have even degree since, then, G is an Euler graph. Any Euler tour in G is a solution to the classical CPP. Each such Euler tour includes each edge of G exactly once and has tour length given by the sum of the edge lengths.

Even when G is not an Euler graph, an optimal tour solving the CPP can still be constructed by carefully selecting edges that are to be repeated during a tour. We generalize the terminology and the techniques from the CPP to develop a basic polynomial-time, deterministic algorithm for finding a near-optimal solution to the RPP. Our techniques find a superset of R, say Q, for which the subgraph of G induced by Q is connected. We define the *Even Parity Completion of Q relative to G*, denoted $Completion_G(Q)$. $Completion_G(Q)$ is an Euler multigraph that naturally determines a tour in G containing each edge of R. We give an example to illustrate that the basic technique does not always produce an optimal RPP tour. However, we note that there is a superset, say S, of R having the property that our near optimal technique, when applied to S rather than R, finds the optimal tour for R. Hence, we apply various strategies, including genetic algorithms, to find a suitable set S.

2 The Rural Postman Problem and the Chinese Postman Problem

We use the graph theory terminology presented by Liu [13]. The *Rural Postman Problem* (RPP) assumes an undirected, connected graph, $G(V, E, R, \omega : E \rightarrow Z_0^+)$. The function ω assigns edge lengths and R is an arbitrary given subset of E. The objective of the RPP is to find a minimum length tour in G that includes each edge of R at least once [8]. R is referred to as the *required* subset of edges. The RPP, is a transformation from the *Hamiltonian Circuit* problem, and, hence, is NP-complete [14]. The RPP remains NP-complete even if $w(e) = 1$ for all e in E [8]. The special case of the RPP that arises when the subset of required edges, R, is the set of all edges,

E, is the classical *Chinese Postman Problem* (CPP). A deterministic, polynomial-time heuristic for solving the CPP is due to Edmonds and Johnson [6] and is included in the book by Evans and Minieka [7]. We will present the heuristic after we define the *Even Parity Completion* of a subset of edges in a graph.

We note that the classical Graphical Steiner Tree Problem (GSP) is a vertex variation of the edge-oriented Rural Postman Problem. Specifically, a solution to the RPP is a *tour* of minimal length that includes all *edges* in a required subset of edges. On the other hand, a solution to the GSP is a *tree* of minimal length that includes all *vertices* in a required subset of vertices. The GSP is also NP-complete and numerous research articles deal with finding near optimal or optimal solutions to this problem [1].

3 Even Parity Completion of a Subset of Edges

Motivated by the well known theorem stating that an undirected graph has an Euler tour if and only if it is connected and its vertices are all of even degree, our research defines, by construction, the *Even Parity Completion of Q relative to G* where Q is any given subset of edges in G. We denote such a completion by $Completion_G(Q)$. In the special case that Q is E, the set of all edges, we abbreviate the terminology to the *Even Parity Completion of G* and the notation to *Completion(G)*.

We first let $G_Q(V_Q, Q)$ be the subgraph of G induced by Q. That is, for a given subset, say Q, of edges in G, V_Q is the set of all vertices incident to any edge in Q. If each vertex in G_Q has even degree, then the $Completion_G(Q)$ is G_Q. Otherwise, we

Step 1 Construct a complete graph G'_Q, derived from G, on the odd degree vertices of G_Q.

Step 2 Find a minimum weight matching on G'_Q.

Step 3 Construct $Completion_G(Q)$ by augmenting G_Q to obtain the multigraph that results from including, or replicating if already included, the edges in G that correspond to the minimum weight matching on G'_Q.

In Step 1, G'_Q is a complete graph on the set of odd degree vertices of G_Q. For each pair of odd-degree vertices in G_Q, say x'_1 and x'_2, the associated edge weight in the complete graph, G'_Q, is assigned to be the length of a shortest path in G between x'_1 and x'_2.

Edmonds and Johnson and others have presented polynomial algorithms for finding a minimum weight matching on a graph as required in Step 2 above. [5, 7].

In the third step of the construction, we let $Completion_G(Q)$ be the multigraph obtained from G_Q by including, replicating if previously included, an edge (and its length) from G each time the edge is part of a selected path in G that corresponds to an edge in the minimum weight matching on G'_Q.

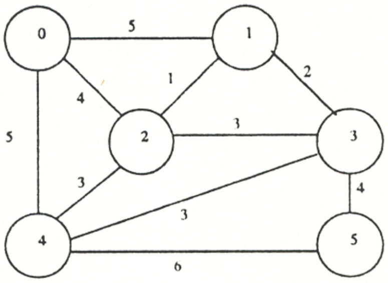


Figure 1: A Graph, $G(V, E, \omega)$

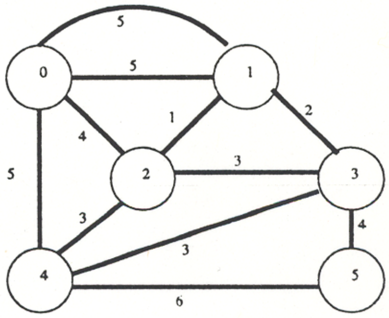


Figure 2: The Completion of $G(V, E, \omega)$

The *Completion of G* for the graph in Figure 1 is easily determined and is indicated in Figure 2. That is, for Figure 1, the complete graph, G', derived from G on the odd degree vertices of G, is a graph with a single edge from v_0 to v_1. The weight of the edge is 5. The minimum weight matching on G' is, of course, this single edge. Replicating the edge $\{v_0, v_1\}$ in G, we obtain the *Completion of G* as shown in Figure 2.

A *Completion of Q relative to G* is shown in bold line style as part of a later figure, Figure 4(c). The graph G is shown in Figure 4(a). G_Q, including the set of five edges given by $Q = \{\{0,1\},\{1,2\},\{1,3\},\{2,4\},\{3,5\}\}$, is shown with bold line style in Figure 4(a). The complete graph G'_Q, derived from G, on the odd degree vertices of G_Q is indicated in Figure 4(b). Edge weights in Figure 4(b) are determined by computing shortest paths in Figure 4(a). The minimum weight matching of G'_Q consists of the edge {0,1} and the edge {4,5}. We obtain $Completion_G(Q)$, by augmenting G_Q with the edge {4,5} and a copy of the edge {0,1}. $Completion_G(Q)$ is shown in bold line style in Figure 4(c).

$Completion_G(Q)$ is not necessarily unique since there may be different, minimum-weight matchings of G'_Q, and since there may be different minimum-length paths selected in G that correspond to an edge in G'_Q. For example, depending on how ties are broken for the graph in Figure 1, *Completion*(G) is either the graph G with edge $\{v_0, v_1\}$ replicated or the graph G with two edges, $\{v_0, v_2\}$ and $\{v_1, v_2\}$, replicated. Figure 2 illustrates the selection of the single edge $\{v_0, v_1\}$. One can show that the sum of the edge weights in any two *Completions of Q relative to G* is the same.

It is easy to show that each vertex in $Completion_G(Q)$ has even degree (that was the objective!). It is also easy to show that if G_Q is connected then $Completion_G(Q)$ is connected. Whenever $Completion_G(Q)$ is connected, then $Completion_G(Q)$ has an Euler tour. Any Euler tour of $Completion_G(Q)$ naturally determines a postman tour in G that contains each edge in Q and, likely, many edges not in Q. Edges in the postman tour, whether or not they are in Q, may be repeated one or more times. However, if G_Q is not connected, then $Completion_G(Q)$ may or may not be connected.

4 Solving the Chinese Postman Problem

A polynomial time deterministic heuristic for the CPP is presented in [7]. If all vertices of G have even degree, an Euler tour of G exists and is a solution to the CPP. In this case, several Euler tours exist; but, they differ from each other only in their routing and each Euler tour includes each edge of G exactly once. Hence, any Euler tour is minimal and, since it traverses each edge one and only one time, has length given by the sum of the edge weights.

When some vertices of G have odd degree and, consequently, G does not have an Euler tour, an optimal tour for the CPP can still be easily constructed by carefully selecting edges that are to be repeated [7]. The construction is done as follows:

Step 1 Construct an *Even Parity Completion* of G.

Step 2 Find a postman tour of G corresponding to an Euler tour of the *Even Parity Completion* of G.

5 Our Basic Rural Postman Problem Heuristic

Our basic heuristic for solving the RPP for the graph $G(V, E, R, \omega)$, where R is the set of required edges and ω is the set of edge weights, is outlined in the following steps.

Step 1 Find a "good" Q such that $R \subseteq Q \subseteq E$ and G_Q is connected. G_Q is the subgraph induced by Q.

Step 2 Construct the *Even Parity Completion of Q relative to G*, $Completion_G(Q)$. Each vertex in $Completion_G(Q)$ has even degree and $Completion_G(Q)$ is connected.

Step 3 Find an Euler tour of $Completion_G(Q)$.

Step 4 Find the postman tour in G determined by the Euler tour of $Completion_G(Q)$.

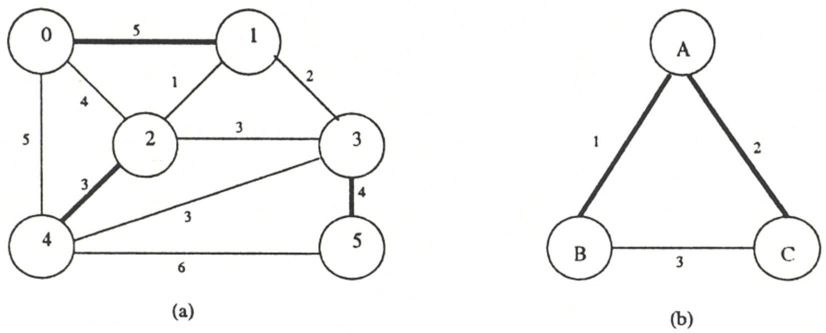
We illustrate each of the four steps in Figures 3, 4, 5, and 6, respectively. In Step 1, shown in Figure 3, R is $\{\{0,1\},\{2,4\},\{3,5\}\}$ and we determine that Q is $R \cup \{\{1,2\},\{1,3\}\}$ as follows. We start with the graph $G(V, E, R, \omega)$ in Figure 3(a). The required set of edges, R, is illustrated with a bold line style and the edge weights are also indicated. The component of R consisting of the edge $\{0,1\}$ is contracted to a vertex v_A. Similarly, the component of R consisting of the edge $\{2,4\}$ is contracted to a vertex v_B and the component corresponding to the edge $\{3,5\}$ is contracted to a vertex v_C. A complete contraction graph, say H, with vertex set $\{v_A, v_B, v_C\}$ is constructed. The weight of an edge in H is assigned to be the shortest distance between the corresponding components in G. Finally, we find the minimum spanning tree of H to be the two edges $\{v_A, v_B\}$ and $\{v_A, v_C\}$. The graph H is shown in Figure 3(b) where the minimum spanning tree is shown with a bold line style.

Continuing with Step 1, the path edges in G corresponding to the edges of the minimum spanning tree of H are used to augment the edges of R and to obtain Q. In our example, the single edge path from component A to component B is $\{\{1,2\}\}$ and the single edge path from component A to component C is $\{\{1,3\}\}$. Hence, $Q = R \cup \{\{1,2\},\{1,3\}\}$. Edges in Q are those indicated with a bold line style in Figure 3(c).

In Step 2, we construct $Completion_G(Q)$. Figure 4(a) is the same figure as Figure 3(c) where the edges in Q are indicated with a bold line style. Edges in $Completion_G(Q)$ are those edges indicated with a bold line style in Figure 4(c). Construction of $CompletionG(Q)$ using the example from Figure 4 was discussed previously in the section on the *Even Parity Completion of a Subset of Edges*.

Finding an Euler tour of $Completion_G(Q)$ in Step 3 is a well known construction since $Completion_G(Q)$ is connected and each vertex has even degree. An Euler tour is illustrated in Figure 5.

Converting an Euler tour of $Completion_G(Q)$ to a postman tour in G is trivial since any edge that is replicated in $Completion_G(Q)$ is simply traversed multiple times in G. The rural postman tour of G that is determined by the Euler tour in Figure 5 and includes all edges of $R = \{\{0,1\},\{2,4\},\{3,5\}\}$ is illustrated in Figure 6. The length of the tour constructed by our basic RPP heuristic is 26.



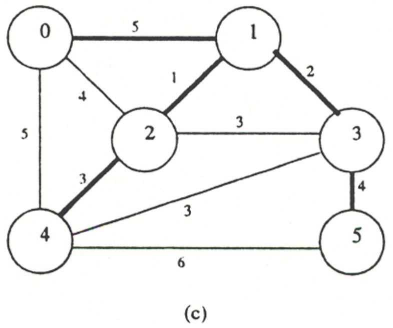


Figure 3: The graph $G(V, E, R, \omega)$ and finding Q so that the subgraph induced by Q is connected.

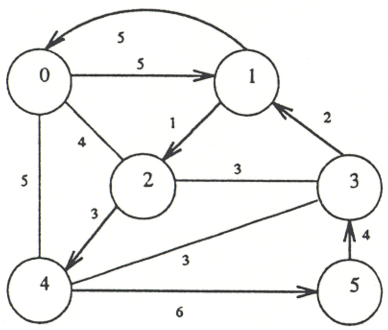


Figure 5: An Euler tour of the *Completion of Q relative to G.*

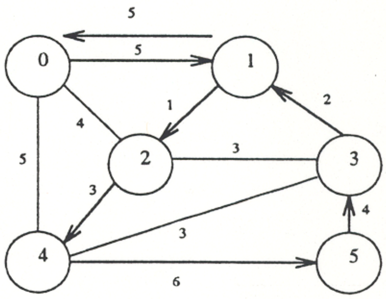


Figure 6: A postman tour in G including all edges of R.

6 Opportunities for Improving Basic RPP Heuristic

Our basic Rural Postman Problem heuristic does not always produce the optimal postman tour for a required subset of edges. Figure 7(a) illustrates a graph G with $R = \{\{0,1\},\{1,2\},\{3,4\},\{4,5\}\}$. Our Step 1 will append the single edge {1,4} to R to obtain the connected Q as illustrated with a bold line style in Figure 7(b). Our Step 2 will then construct $Completion_G(Q)$ as the multiset $R \cup \{\{0,3\},\{1,4\},\{2,5\}\}$ where the edge {1,4} is repeated. $Completion_G(Q)$ is illustrated in Figure 7(c). The postman tour in G, determined by the Euler tour of $Completion_G(Q)$, traverses the edge {1,4} twice while traversing all other edges once. The postman tour has length 10. Clearly, a postman tour that simply traverses the perimeter of the graph, and does not traverse the edge {1,4} at all, will contain all of the required edges and will have length of 8.

In Figure 8, we illustrate that our basic RPP heuristic may actually find a better postman tour for a required edge subset, R, if we artificially force R to include one or more other edges from $E - R$. To illustrate, we again use the graph G in Figure 7(a) with $R = \{\{0,1\},\{1,2\},\{3,4\},\{4,5\}\}$ and we obtain $R' = R \cup \{\{0,3\}\}$ by forcing the "required" set to also include {0,3} as illustrated by Figure 8(a). Step 1 of our basic RPP heuristic determines that

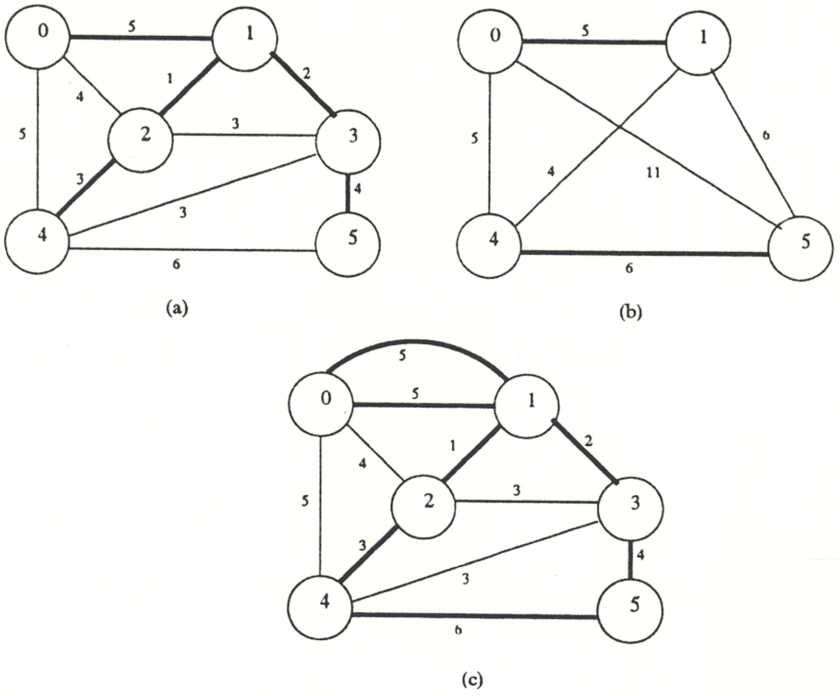


Figure 4: *Constructing the Even Parity Completion of Q relative to G.*

the R' is already connected and so Q is R'. In Step 2, $Completion_G(Q)$ augments R' by including edge $\{2,5\}$. $Completion_G(Q)$ is the graph consisting of each edge in the perimeter included exactly one time. Hence, the postman tour resulting from application of our basic RPP heuristic will traverse the perimeter of G without ever using the edge $\{1,4\}$. As indicated before, the tour has length 8, an improvement over a tour of length 10.

Since forcing additional non-required edges may improve the tour produced by our basic RPP heuristic, we are motivated to develop various strategies to select additional edges to augment R.

It is important to note that, for any required subset of edges, R, there is a set R' for which $Completion_G(R')$ corresponds to the optimal RPP tour. However, it may or may not be true that the "good" Q determined by our connection process in Step 1 of the basic RPP heuristic is such an R'.

Our improved RPP heuristic algorithms are Force One Edge, Force Two Edges, Iterated Force One Edge, and The Genetic Algorithm.

In the following sections, we describe the improved algorithms. It is interesting and useful to note that any of the four improvements could be used to improve any underlying RPP heuristic. We, in particular, use the improvements with the basic RPP heuristic outlined previously.

6.1 Force One Edge

For the RPP problem indicated by $G(V, E, R, \omega)$, the *Force One Edge* improvement to the basic RPP heuristic consists of the following three steps.

Step 1 Apply the basic RPP heuristic using R as the required edge set.

Step 2 For each $e \in E - R$, apply the basic RPP heuristic using the required edge set given by $Q = R \cup \{e\}$.

Step 3 Choose the best tour resulting from Step 1 and Step 2.

Clearly, application of the *Force One Edge* heuristic will do no worse than the basic RPP heuristic. The order of the *Force One Edge* heuristic is $n * \vartheta(m)$ where n is the number of elements in $E - R$ and $\vartheta(m)$ is the order of the basic RPP heuristic.

6.2 Force Two Edges

For the RPP problem indicated by $G(V, E, R, \omega)$, our *Force Two Edges* improvement to the basic RPP heuristic consists of the following three steps.

Step 1 Apply the basic RPP heuristic using R as the required edge set.

Step 2 For each $e_1, e_2 \in E - R$, apply the basic RPP heuristic using the required edge set given by $Q = R \cup \{e_1, e_2\}$.

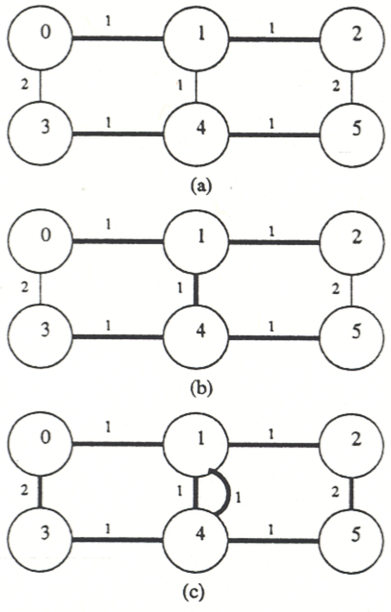


Figure 7: An suboptimal tour produced by the basic RPP heuristic.

Step 3 Choose the best tour resulting from Step 1 and Step 2.

Clearly, as with *Force One Edge*, application of the *Force Two Edges* heuristic will do no worse than the basic RPP heuristic. The order of the *Force Two Edges* heuristic is $n^2 * \vartheta(m)$ where n is the number of elements in $E - R$ and $\vartheta(m)$ is the order of the basic RPP heuristic.

6.3 Iterated Force One Edge

Motivated by the work of Alexander and Robbins for improving near optimal solutions to the Graphical Steiner Tree Problem [1], we describe our *Iterated Force One Edge* heuristic for the RPP. The heuristic consists of the following steps.

```
Step 1 Let $Q = R$.
Step 2 Apply the basic RPP heuristic using $Q$ as the
       required edge set.
Step 3 Repeat
       For each edge $e \in E - Q$, apply the basic RPP
       heuristic using $S = Q \cup \{e\}$ as the required edge
       set.
       Determine the best $e \in E - Q$ and let $Q = Q \cup \{e\}$.
       Until (no improvement or $Q = E$).
```

Again, the *Iterated Force One Edge* heuristic will do no worse than the basic RPP heuristic. In the special case that the *Iterated Force One Edge* heuristic stops after one iteration of the repeat loop, the *Iterated Force One Edge* produces the same result as the *Force One Edge* heuristic. When the *Iterated Force One Edge* heuristic stops after two iterations of the repeat loop, the two successful edges chosen by the first and second iterations of the repeat loop may or may not correspond to a pair of successful edges chosen by the *Force Two Edges* heuristic. The order of the *Iterated Force One Edge* heuristic is $n^2 * \vartheta(m)$ where n is the number of elements in $E - R$ and $\vartheta(m)$ is the order of the basic RPP heurisitic.

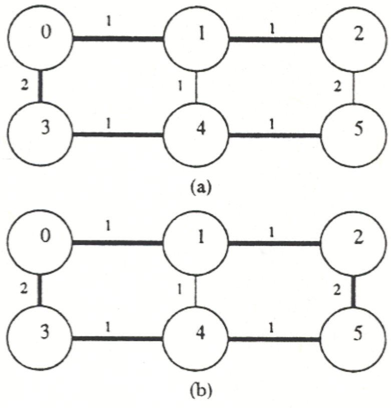


Figure 8: An improved tour produced by the basic RPP heuristic after forcing an additional edge.

6.4 Genetic Algorithms

The GA works by selection, recombination, and mutation on the fixed length character strings [9, 10, 11, 12, 16]. Several researchers have investigated the benefits of solving combinatorial problems using genetic algorithms. Davis [4], Goldberg [9], Rawlins [16], Corcoran and Wainwright [3], and Blanton and Wainwright [2] provide excellent examples. The genetic algorithm implementation used in this research is libGA [3].

We use a bit string chromosome to represent a RPP tour for $G(V, E, R, \omega)$. The chromosome is a bit string with length given by *cardinality(E)* - *cardinality(R)*. The bit positions in the chromosome are indexed by the edges of E that are not in R. If an allele is 1, we force the inclusion of the corresponding edge into the subset of required edges. We apply the basic RPP heuristic to the augmented set of "required" edges, say R'.

As an example, Figure 9(a) illustrates the same graph and the same required set that we have used in other illustrations. $R = \{\{0,1\}, \{2,4\}, \{3,5\}\}$ is the subset of required edges as indicated by a bold line style. The edges in $E - R$ are labeled by $e_0, e_1, ..., e_6$. Figure 9(b) illustrates the augmented set of required edges that is determined by the chromosome (0 1 0 1 0 0 0). That is, we force the inclusion of the edge e_1 and e_3 and the augmented set of edges is $R' = R \cup \{\{0,2\}, \{1,3\}\}$. R'

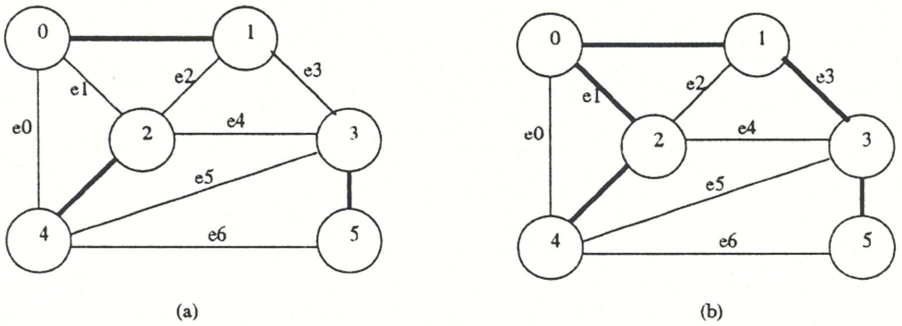


Figure 9: Additional required edges for the chromosome (0 1 0 1 0 0 0).

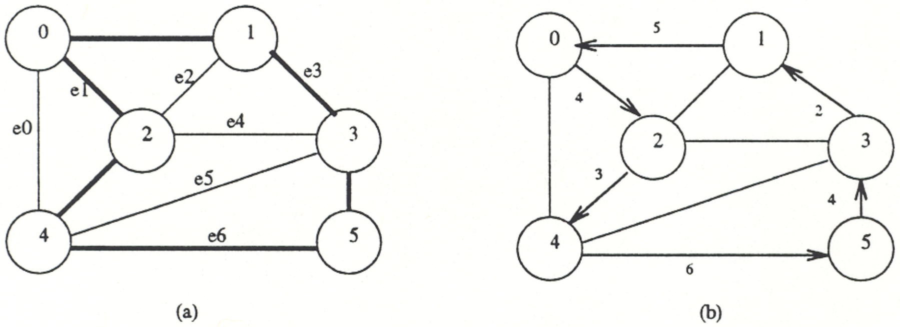


Figure 10: Basic RPP heuristic applied with chromosome (0 1 0 1 0 0 0).

is a connected set of edges as indicated with a bold line style in Figure 9(b). Since R' is already connected, Q is selected to be R' in Step 1 of the basic RPP heuristic. Step 2 of the basic RPP heuristic further includes edge $e_6 = \{4, 5\}$ during the construction of $Completion_G(Q)$. $Completion_G(Q)$ is illustrated with a bold line style in Figure 10(a). The postman tour resulting from Step 3 and Step 4 of the basic RPP heuristic is illustrated in Figure 10(b). The length of the tour resulting from the fitness evaluation of the chromosome (0 1 0 1 0 0 0) is 24. The basic RPP heuristic produced a tour of length 26.

Our generational genetic algorithm uses simple crossover. Our mutation is bit inversion. The crossover rate is 1.0 and the mutation rate is 0.05. The population size was 200 for one set of runs and 500 for another set of runs.

7 Data Sets and Results

The data sets used in our research are modifications of the eighteen benchmark *steinb* data sets that are used for the Graphical Steiner Tree problem. The data sets are available from the *OR Library* at *http://mscmga.ms.ic.ac.uk*. The vertices, edges, and edge weights indicated in the *steinb* data sets are maintained. The set of required vertices is simply ignored and replaced with a set of required edges. The set of required edges for each of our graphs was determined by doing a Bernoulli selection on each edge in the set of all edges. The resulting data sets were fixed and, rather than being named *steinbx*, are named *rppbx*. The eighteen *rppbx* data sets are available from *http://euler.utulsa.edu/~schoend*.

The results of applying our various techniques to the *rppbx* data sets are listed in Table 1. Each row in the table corresponds to one of the data sets. The cardinalities of the vertex set, the edge set, and the required edge subset are indicated. The remaining columns give the length of the rural postman tour that is determined by the *basic RPP* heuristic, the *Force One Edge* heuristic, the *Force Two Edges* heuristic, the *Iterated Force One Edge* heuristic, the *Genetic Algorithm* with population size 200, and the *Genetic Algorithm* with population size 500. The first four heuristics are deterministic. The two genetic algorithms are not deterministic. The best tour for each data set is indicated with a '*' in Table 1.

In general, the *Iterative Force One Edge* heuristic produced the best rural postman tours. The success of the technique can be explained by noting that whenever there are three or four independent edges whose forced inclusion will improve the performance of the basic RPP heuristic, the *Iterative Force One Edge heuristic* can most easily find the edges. The frequent ties among the *Force One Edge*, the *Force Two Edges*,

x	*card* *V*	*card* *E*	*card* *R*	Basic RPP	Force One	Force Two	Iter.	GA 200	GA 500
1	50	63	12	192	188 *	188 *	188 *	188 *	188 *
2	50	63	12	197	197	196 *	197	200	197
3	50	63	12	139	134	130 *	130 *	130 *	142
4	50	100	20	150	146 *	146 *	146 *	146 *	146 *
5	50	100	20	171	167	166	161 *	179	168
6	50	100	20	161	156 *	156 *	156 *	156 *	160
7	75	94	18	268	260	253	250 *	260	253
8	75	94	18	228	211	209 *	209 *	209 *	209 *
9	75	94	18	295	291	291	291	303	290 *
10	75	150	30	286	285	283 *	283 *	294	284
11	75	150	30	249 *	249 *	249 *	249 *	267	262
12	75	150	30	262	257	253 *	253 *	291	267
13	100	125	25	315	303	301 *	301 *	304	306
14	100	125	25	340	337	335	332	347	327 *
15	100	125	25	292	292	284 *	284 *	330	287
16	100	200	40	410	404	400	396 *	437	521
17	100	200	40	350	347	344	336 *	391	452
18	100	200	40	374	370 *	370 *	370 *	411	477

Table 1: Rural Postman Tour Lengths, *rppb*x Data Sets

and the *Iterated Force One* edge heuristics indicates that, frequently, there are only one or two edges whose forced inclusion into the required edge subset are key to the improved performance of the basic RPP heuristic. The four instances where the *Force Two Edges* heuristic finds a better tour than the *Iterated Force One* heuristic indicates that the single edge selection made during the first iteration of the *Iterated Force One* heuristic actually detracts from finding a good pair of edges for forced inclusion into the required edge set. Although there are many instances where the genetic algorithm produced a tie tour and two instances where it was the only heuristic that produced the best tour, the overall poor performance of the *genetic algorithm* can be explained by noting that too much time is spent examining chromosomes that contain many 1s. Such a chromosome forces the inclusion of numerous edges that are not particularly helpful for isolating a small number of key edges. However, the other heuristics that we developed as part of this research, *Force One Edge*, *Force Two Edges*, and *Iterated Force One Edge*, performed extremely well.

8 Future Work

We have improved our data structures and code efficiency so that we can examine data sets with a larger number of vertices, edges, and required edges. We are also investigating strategies for biasing the *genetic algorithm* to favor chromosomes with relative few 1s so that the algorithm will make an earlier identification of the key edges that improve the performance of the basic RPP heuristic.

Perhaps more interesting is the exploration of an alternate basic RPP heuristic. One of the useful observations concerning all of our improved RPP heuristics is that they can be used with any underlying 'basic' RPP heuristic.

During the first step, the basic RPP heuristic used for this research finds a good Q so that $R \subseteq Q$ and Q is connected. During the second step of the basic RPP heuristic, we construct the *Even Parity Completion of Q relatively to G.* It may prove to be useful to essentially reverse these two steps. That is, we will examine the tours resulting from, first, constructing the *Even Parity Completion of R relative to G*, which may or may not be connected, followed by inserting additional edges into the *Even Parity Completion* so that the resulting construction is connected. This approach was previously suggested by Orloff [15]. The challenge is to add a small number of short additional edges so that the even parity at each vertex is maintained.

References

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

[2] J. Blanton and R. Wainwright, "Multiple Vehicle routing with time and capacity constraints using genetic algorithms," S. Forrest (ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 452-459, Morgan Kaufman, 1993.

[3] A. Corcoran and R. Wainwright, "Using LibGA to Develop Genetic Algorithms for Solving Com-

binatorial Optimization Problems," *Practical Handbook of Genetic Algorithms, Applications*, Volume 1, Editor Lance Chambers, CRC, 1995.

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

[5] J. Edmonds and E. L. Johnson, "Matching: A Well Solved Class of Integer Linear Programs," *Combinatorial Structures and their Applications.* Gordon and Breach, New York, pages 89-92, 1970.

[6] J. Edmonds and E. L. Johnson, "Matching, Euler Tours, and the Chinese Postman," *Mathematical Programming* 5, pages 88-124, 1973.

[7] J. R. Evans and E. Minieka, *Optimization Algorithms for Networks and Graphs*, Second Edition, Dekker, 1992.

[8] M. R. Garey and D. S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-Completeness*, Freeman, 1979.

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

[10] Kenneth E. Kinnear, *Advances in Genetic Programming*, The MIT Press, 1994.

[11] John R. Koza, *Genetic Programming*, The MIT Press, 1992.

[12] John R. Koza, *Genetic Programming II*, The MIT Press, 1994.

[13] C. L. Liu, *Elements of Discrete Mathematics*, Second Edition, McGraw Hill, 1985.

[14] J. K. Lenstra and A. H. G. Rinnooy Kan, "On general routing problems," *Networks* 6, pages 273-280, 1976.

[15] C. S. Orloff, "A Fundamental Problem in Vehicle Routing," *Networks* 4,: pages 35-64, 1974

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

[17] C. H. Papadimitriou, "On the Complexity of Edge Traversing," *Journal of the Association of Computing Machinery* 23, pages 544-554, 1976.