Heuristics for the Stochastic Eulerian Tour Problem

Srimathy Mohan
Michel Gendreau
Jean-Marc Rousseau

October 2007

CIRRELT-2007-46

Heuristics for the Stochastic Eulerian Tour Problem

Srimathy Mohan[1], Michel Gendreau[2,3,*], Jean-Marc Rousseau[2]

1. School of Global Management and Leadership, MC 2451, Arizona State University, P.O. Box 37100, Phoenix, AZ 85069-7100, USA
2. Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Université de Montréal, P.O. Box 6128, Station Centre-ville, Montréal, Canada H3C 3J7
3. Department of Computer Science and Operations Research, Université de Montréal, P.O. Box 6128, Station Centre-ville, Montréal, Canada H3C 3J7

Abstract. The Stochastic Eulerian Tour Problem (SETP) seeks the Eulerian tour of minimum expected length on an undirected Eulerian graph, when demand on the arcs that have to be serviced is probabilistic. In an earlier paper, we have shown that the SETP is NP-hard and derived several properties of the optimal tour. In this paper, we develop three constructive heuristics for the SETP. The first two are greedy tour construction heuristics while the third is a sub-tour concatenation heuristic. We tested the performance of the three heuristics and a post-optimization procedure on grid networks and on Euclidean graphs. Our results indicate that the sub-tour construction heuristic performs well when the probability of occurrence of each edge in the graph, $p > 0.1$ for the grid networks. For the Euclidean networks, as the number of edges increases, the second heuristic performs the best among the three, though the margin of improvement is small.

Keywords. Arc routing, Eulerian tour problem, stochastic demand.

Acknowledgements. Financial support for this work was provided by the Natural Sciences and Engineering Research Council of Canada (NSERC) and by the Fonds FCAR (renamed Fonds québécois de la recherche sur la nature et les technologies, FQRNT). This support is gratefully acknowledged.

---

* Corresponding author: michel.gendreau@cirrelt.ca

This document is also published as Publication #1309 by the Department of Computer Science and Operations Research of the Université de Montréal.

Dépôt légal – Bibliothèque nationale du Québec,
Bibliothèque nationale du Canada, 2007

1. Introduction

This paper presents three heuristic procedures for the Stochastic Eulerian Tour Problem (SETP). We defined the SETP in a companion paper as follows. We are given an undirected Eulerian graph $G = (V, E)$, a set $R\left(R \subseteq E, |R| = n\right)$ of edges that require service, and a distance $d(v_i, v_j)$ between every pair of directly connected nodes v_i and v_j. On any instance of the problem, only a subset of the *n* service edges is present, and hence, requires a visit. The number of present edges follows a specified probability distribution. The objective is to determine an a priori Eulerian tour that visits all the *n* edges and minimizes the expected length of the tour. On any given instance, one visits and services the present edges in the same order as in the a priori tour, while skipping the ones that are absent.

Since we have shown that the SETP belongs to a class of hard problems, we concentrate on developing heuristic algorithms that would provide good solutions. In this paper, we present three different tour construction heuristics for the SETP. The first heuristic is a simple greedy heuristic that determines the next service edge to add to the tour as the one that results in the least increase in the expected length when appended at the end of the tour. The second heuristic also greedily selects the next edge of the tour. However, this heuristic selects the next service edge from the set of edges incident to the current node rather than from the set of all available service edges. The third heuristic constructs several small sub-tours and then concatenates these sub-tours by considering the expected savings in concatenating them. We have also incorporated an adaptation of a post-optimization procedure introduced by Gendreau, Hertz, and Laporte (1992) for the TSP. Hertz, Laporte, and Nanchen (1999) call this adaptation as DROP-ADD and have used it for the undirected Rural Postman Problem.

Our heuristics are partly based on the heuristics for the stochastic node routing problem first introduced by Jaillet (1985) as the *Probabilistic Traveling Salesman Problem* (PTSP). He has proposed a number of heuristics by suitably modifying several well-known TSP heuristics such as the Clarke-Wright algorithm and nearest neighbor algorithm. Rossi and Gavioli (1988) present computational results after testing three of Jaillet's heuristics. Their computational results indicate that the probabilistic Clarke-

Wright algorithm produces TSP tours with lower expected lengths than the corresponding deterministic TSP heuristics when the probabilities are low (between 0 and 0.6). We first present the theoretical preliminaries that we use to design our heuristics in Section 2. Section 3 presents detailed descriptions of the three tour construction heuristics and the post optimization procedure for the SETP. We present detailed computational results in Section 4 and provide the conclusion and directions for future research in Section 5.

2. Theoretical Preliminaries

The given undirected graph G has a designated depot where the Eulerian tour starts and ends. In order to facilitate the representation and analysis, we duplicate the depot and represent the duplicated node as v_0, which now serves as the depot and is connected to the original depot by two edges of length 0, one of which is a service edge.

Given an Eulerian tour t for the graph G, we have an ordering of the nodes and edges, and thus, a direction of traversal (and service) for each of the n service edges. If we traverse edge e_i from node v_k to v_l, we define v_k as the in-node for edge e_i $\left(v_i^{in}\right)$ and v_l as the out-node for edge e_i $\left(v_i^{out}\right)$. Thus, given the in-node and the out-node for each edge in R, we represent an Eulerian tour t as $t=\left(v_0, v_1^{in}, e_1, v_1^{out}, v_2^{in}, e_2, v_2^{out}, \ldots, v_n^{in}, e_n, v_n^{out}, v_0\right)$, where the edges $e_1, e_2, \cdots, e_n$ are numbered in their order of appearance in tour t. When the number of present service edges follows a binomial distribution with parameter p, the expected length of a given tour t is

$$E[L_t] = p^2\left[\sum_{r=0}^{n-2}(1-p)^r\ L_t^r\right] + p\,(1-p)^{n-1}\ L_t^{n-1} + p\left[\sum_{i=1}^{n} l(e_i)\right] \tag{1}$$

where $$L_t^r = \sum_{j=0}^{n} d\left(v_j^{out}, v_{\overline{j+r+1}}^{in}\right) \qquad \forall\ r \in \{0,\cdots,n-1\}$$

with $$v_0^{out} = v_{n+1}^{in} = v_0$$

$\overline{j+r+1} = (j+r) \bmod (n+1) + 1$, and

$$d\left(v_j^{out}, v_{\overline{j+r+1}}^{in}\right) = \begin{cases} d\left(v_j^{out}, v_{j+r+1}^{in}\right) & \text{if } 0 \leq j \leq n-r \\ d\left(v_j^{out}, v_0\right) + d\left(v_0, v_{j-n+r}^{in}\right) & \text{if } n-r < j \leq n \end{cases}$$

Note that if nodes v_i and v_j are not directly connected, then $d(v_i, v_j)$ is the shortest distance between v_i and v_j.

2.1. Addition of a Service Edge to a Path

Consider a path P starting at the depot v_0 and servicing $i\,(i < n)$ service edges. Let us now add service edge e_{i+1} to the path. We can calculate the expected increase in the length of the path by adding edge e_{i+1} by considering the following two cases:

(i) the service edges $e_1, e_2, \ldots, e_i$ already in the path are not present; and

(ii) k of the i service edges in the path are not present.

The expected increase in the length of path P by adding service edge e_{i+1} can now be expressed as

$$E\left[I\left(L_P, e_{i+1}\right)\right] = p\,(1-p)^i\; d\left(v_0, v_{i+1}^{in}\right) + p^2 \sum_{k=0}^{i-1} (1-p)^k\; d\left(v_{i-k}^{out}, v_{i+1}^{in}\right) + p\, l\left(e_{i+1}\right) \tag{2}$$

The first term of expression (2) corresponds to the situation where edge e_{i+1} is the first edge present on the tour and the previous *i* edges are absent. Hence, we need to consider the direct distance between the depot and the in-node of edge e_{i+1}. The probability associated with this event is $p(1-p)^{i-1}$. Similarly, the second term corresponds to the scenario where exactly *k* edges $(k = 0,1,2,\ldots,i-1)$ are absent before edge e_{i+1}. In this case, edges e_{i-k} and e_{i+1} are present and edges e_{i-k+1} through e_i are absent. Hence, we need to consider the distance between the out-node of edge e_{i-k} and the in-node of edge e_{i+1}, and the associated probability is $p^2(1-p)^k$. The last term corresponds to the probability that edge e_{i+1} is present.

Note that in order to calculate $E\left[I\left(L_p, e_{i+1}\right)\right]$ using (2), we need to designate one of the ends of edge e_{i+1} as the in-node, i.e., we need to fix v_{i+1}^{in}. If $e_{i+1} = \left(v_k, v_l\right)$, we calculate $E\left[I\left(L_P, e_{i+1}\right)\right]$ for both orientations of e_{i+1}, i.e., by fixing v_{i+1}^{in} as v_k once and as v_l once, and pick the orientation with the minimum expected increase.

2.2. Merging of Sub-tours

The probabilistic Clarke-Wright algorithm proposed by Jaillet (1985) for the PTSP serves as a motivation for the results in this sub-section. In our case, a sub-tour for the given graph G starts at the depot, services $i\ (i<n)$ service edges, and returns to the depot. Let us consider two sub-tours ST_1 and ST_2 which do not service any common service edge. Sub-tours ST_1 and ST_2 service n_1 and n_2 edges, respectively.

$$ST_1 = \left(v_0, v_{1,1}^{in}, e_{1,1}, v_{1,1}^{out}, v_{1,2}^{in}, e_{1,2}, v_{1,2}^{out}, \ldots, v_{1,n_1}^{in}, e_{1,n_1}, v_{1,n_1}^{out}, v_0\right)$$

$$ST_2 = \left(v_0, v_{2,1}^{in}, e_{2,1}, v_{2,1}^{out}, v_{2,2}^{in}, e_{2,2}, v_{2,2}^{out}, \ldots, v_{2,n_2}^{in}, e_{2,n_2}, v_{2,n_2}^{out}, v_0\right)$$

Let $E\left\lfloor L_{ST_1}\right\rfloor$ and $E\left\lfloor L_{ST_2}\right\rfloor$ be the expected length of the sub-tours ST_1 and ST_2. When we merge ST_1 and ST_2 by edges e_{1,n_1} and $e_{2,1}$ (i.e., $v_{2,1}^{in}$ directly follows v_{1,n_1}^{out} on the merged sub-tour), the expected length of the merged sub-tour could be less than or equal to the total expected lengths of ST_1 and ST_2. This expected savings in the length of the merged sub-tour could be calculated by considering the following events and the associated probabilities.

- Edge $e_{1,i}\ \left(1 \le i \le n_1\right)$ is the last present service edge on ST_1 --

  probability: $p\left(1-p\right)^{n_1-i}$

- Edge $e_{2,j}$ $(1 \leq j \leq n_2)$ is the first present service edge on ST_2 --

  probability: $(1-p)^{j-1}\,p$

When $e_{1,i}$ is the last present edge on ST_1 and $e_{2,j}$ is the first present edge on ST_2, the savings incurred in the distance traversed is

$$S(e_{1,i}, e_{2,j}) = d(v_{1,i}^{out}, v_0) + d(v_0, v_{2,j}^{in}) - d(v_{1,i}^{out}, v_{2,j}^{in}) \tag{3}$$

The expected savings when we merge ST_1 and ST_2 through edges e_{1,n_1} and $\mathrm{e}_{2,1}$ can be represented as

$$E\left[S(\mathrm{ST}_1, ST_2 : e_{1,n_1}, e_{2,1})\right] = \sum_{i=1}^{\mathrm{n}_1} \sum_{\mathrm{j}=1}^{n_2} S(e_{1,i}, e_{2,j})\, p_{ij} \tag{4}$$

where, $\quad p_{ij} = p^2 (1-p)^{(n_1-i)+(j-1)}$

and $\quad S(e_{1,i}, e_{2,j})$ is given by (3)

Note that we have four different ways to merge ST_1 and ST_2. We can merge the sub-tours through edges e_{1,n_1} and $e_{2,1}$, or e_{1,n_1} and e_{2,n_2}, or $e_{1,1}$ and $e_{2,1}$, or $e_{1,1}$ and e_{2,n_2}. The best way to merge ST_1 and ST_2 is through the edge concatenation that yields the maximum expected savings. Thus, the expected savings from merging ST_1 and ST_2 is

$$E\left[S(\mathrm{ST}_1, ST_2)\right] = \max_{h \in \{1, n_1\}, k \in \{1, n_2\}} \left[E\left[S(ST_1, ST_2 : e_{1,h}, e_{2,k})\right]\right] \tag{5}$$

3. Heuristic Procedures

In this section, we present detailed descriptions of the three heuristics that we have developed for the SETP. All three are tour construction procedures. The first and the second heuristics construct the tour by adding one service edge at a time, while the third heuristic constructs several small sub-tours and then concatenates these to form the Eulerian tour. Finally, we also describe the post-optimization procedure DROP-ADD that is analogous to the US procedure (Gendreau et al., 1992) for the TSP.

3.1. Heuristic 1: Global Greedy

This heuristic starts with an empty path and successively adds one edge at a time to the path. The edge added to the path is the one that results in the least expected increase in the length of the resulting path. Once all the n edges are added, we complete the tour by returning to the depot from the out-node of the last added service edge.

Two components contribute to the expected increase in the path length as computed by (2). The first is the inter-edge traversal distances between the service edges on the path and the candidate edge, and the second is the length of the candidate edge. Hence, the heuristic tends to select shorter edges closer to the out-node of the last edge in the path. As a result, in certain situations, the tours produced by this heuristic could be longer than necessary when all the service edges are present. For example, consider the graph in Figure 1 with 12 service edges. Node 1 is the depot. The length of all the edges on the three outer triangles is 5 and the length of the edges on the connecting inner triangle is 1.

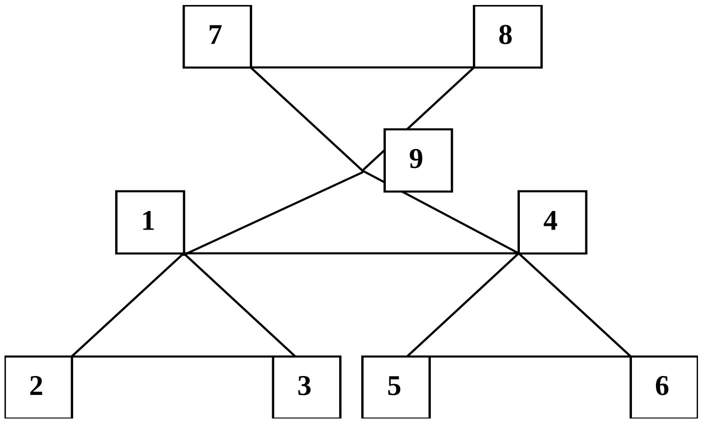

Figure 1. Example graph for heuristic 1

When each edge in the graph is present with a probability $p = 1.0$, the global greedy heuristic produces the tour (1-4-9-1-2-3-1-4-5-6-4-9-7-8-9-1) of length 51. But, it is obvious that we can reduce the total length to 48 since the edges (1,4), (4,9), and (9,1) are traversed twice. In this case, the expected length will also be reduced since $p = 1.0$. Using this as a motivation, we apply the procedure SHORTEN (Hertz et al., 1999) to the tour produced by the global greedy heuristic. Basically, this procedure starts with node r (= 1), and moves all the service edges as far to the right as possible. It then replaces the path from node r to the in-node of the first service edge by the shortest chain. If the length of the resulting tour is smaller, we renumber all the nodes on the new tour and start the procedure again at node r (= 1). If not, we increment r by 1 and continue until no more reduction in the tour length is possible.

If we apply procedure SHORTEN to the above tour, we get the tour (1-2-3-1-4-5-6-4-9-7-8-9-1) of length 48. For values of p less than 1.0, the expected length of the shortened tour could be greater than the expected length of the original tour since the ordering and the orientations of the service edges could be changed during the SHORTEN procedure. Hence, we calculate the expected length of the shortened tour and retain this tour as the final result only if its expected length is less than or equal to the expected length of the original tour.

Algorithm GLOBAL GREEDY

```
Step 0:  Initialize $P \leftarrow (v_0)$ and $B \leftarrow E$.
Step 1:  For every edge $e_i$ in $B$, calculate $E[I(L_P, e_i)]$ as described in Section 2.1.
Step 2:  Set $k \leftarrow \text{argmin}\{E[I(L_P, e_i)]\}$. Append the tour segment $(v_k^{in}, e_k, v_k^{out})$ to $P$.
         Set $B \leftarrow B - \{e_i\}$.
Step 3:  If $B = \varnothing$, append the return path from the out-node of the last added service edge to the
         depot to $P$ to get tour $t$ and go to Step 4. If not, go to Step 1.
Step 4:  Apply procedure SHORTEN to tour $t$ to obtain tour $t'$. If $E[L_{t'}] \le E[L_t]$, $t'$ is the final
         tour, else $t$ is the final tour.
```

3.2. Heuristic 2: Local Greedy

This heuristic is a modification of the global greedy heuristic. At each iteration, instead of selecting the next service edge from among all the available service edges, we choose from only among the service edges incident to the node we are at (current node). Thus, this heuristic starts at the depot, adds the service edge with the least expected increase in length from among all the service edges incident to the depot, and repeats the process at the out-node of the added service edge. If at any point, there are no service edges incident to the current node, there may be a matching edge incident to this node, that was added to the graph while solving the augmentation problem. If this is the case, we traverse the matching edge and continue the process, until all service edges are added and we have a complete tour.

This heuristic ensures that when p = 1.0, the length of the tour generated using this heuristic is equal to the length of a random Eulerian tour.

<u>Algorithm LOCAL GREEDY</u>

Step 0: Initialize $P \leftarrow (v_0)$ and $B \leftarrow E$. Calculate the degree of all the nodes in the graph. Let *current_node* $\leftarrow$ v_0.

Step 1: Let *W* = set of service edges incident to *current_node.* If $W \neq \varnothing$, calculate $E\left[I\left(L_P, e_i\right)\right]$ as described in Section 2.1. for every edge e_i in *W*, and go to Step 2. If $W = \varnothing$, go to Step 3.

Step 2: Set $k \leftarrow \underset{e_i \in W}{\text{argmin}} \left\{E\left[I\left(L_P, e_i\right)\right]\right\}$. Append the tour segment $\left(v_k^{in}, e_k, v_k^{out}\right)$ to P. Set $B \leftarrow B - \{e_i\}$. Decrease degree of *current_node* and v_k^{out} by 1. Set *current_node* $\leftarrow$ v_k^{out}, and go to Step 4.

Step 3: If degree[*current_node*] > 0, there is a matching edge incident to *current_node*. Traverse this matching edge. Decrease degree of *current_node* and the out-node of the matching edge by 1. Set *current_node* $\leftarrow$ out_node of the matching edge, and go to Step 4.

If degree[*current_node*] = 0, backtrack on the partial tour just developed, to a node v_i with degree > 0. Set *current_node* $\leftarrow v_i$, and go to step 4.

Step 4: If $B = \varnothing$, append the return path from the out-node of the last added service edge to the depot to P to get tour t and go to Step 5. If not, go to Step 1.

Step 5: Apply procedure SHORTEN to tour t to obtain tour t'. If $E\left[L_{t'}\right] \leq E\left[L_t\right]$, t' is the final tour, else t is the final tour.

3.3. Heuristic 3: Sub-Tour Construction

The third heuristic is also a tour construction heuristic, but is quite different from the global and local greedy heuristics. This heuristic first constructs a single giant sub-tour using a set *C* of eligible edges, and another outer tour using the edges in *E\C*. The sub-tour construction heuristic then breaks up the giant sub-tour into as many small separate sub-tours as possible. Finally, the heuristic inserts these small sub-tours at appropriate insertion points on the outer tour to obtain the Eulerian tour. We first describe each of the procedures of the algorithm and then provide a detailed description of the overall heuristic.

Determination of the set of eligible edges

This procedure forms a set *C* of the edges that can be used to form the giant sub-tour. If the degree of a node is greater than 2, then this node occurs more than once in the final Eulerian tour, and hence there is a sub-tour out of this node. This fact motivates the idea behind forming the set *C*. In order to determine *C*, we need to consider only edges whose both end points are of degree greater than 2.

Step 1: Set $C \leftarrow \varnothing$. Calculate the degree of all the nodes in the graph.

Step 2: If the two end points of an edge $e_i \in E$ are both of degree > 2, then add edge $e_i \in E$ to the set *C*.

Breaking up a single sub-tour into smaller sub-tours

This procedure breaks a sub-tour that starts at the in-node of a service edge in *C* and ends at the out-node of another service edge in *C* into as many small sub-tours as possible. Given a giant sub-tour, we have an ordering of the nodes. The procedure uses this ordering and results in more than one sub-tour if one or more nodes are visited more than once in the giant sub-tour. Let the given sub-tour containing *k* nodes be represented as $ST = (v_{s,1}, v_{s,2}, \ldots, v_{s,k})$. Since the in-node of the first edge on a given sub-tour and the out-node of the last service edge are the same, we do not store the out-node of the last service edge in *ST* for notational simplicity. However, when we merge a given sub-tour with another sub-tour or the outer tour, we always ensure that the last service edge on the given sub-tour is traversed, by visiting the out-node of the last service edge. We store the resulting smaller sub-tours in the array *sub_tour*.

Step 0: Set $sub\_tour[1] \leftarrow ST$. For each node $v_i \in V$, set $count[v_i] \leftarrow 0$.

Step 1: Set $count[v_{s,i}] \leftarrow count[v_{s,i}] + 1$ for all $i = 1, \ldots, k$, *num_tours* ← 1, and $i \leftarrow 1$.

Step 2: If $count[v_{s,i}] > 1$, let j be the position of node $v_{s,i}$ the second time it occurs on the given sub-tour. If not, set $i \leftarrow i+1$, and go to Step 4.

Step 3: Remove the sub-tour $(v_{s,i}, v_{s,i+1}, \ldots, v_{s,j-1})$ from *ST*. For $l = i, \ldots, j-1$, set $count[v_{s,l}] \leftarrow count[v_{s,l}] - 1$. Set $num\_tours \leftarrow num\_tours + 1$, and $sub\_tour[num\_tours] \leftarrow (v_{s,i}, v_{s,i+1}, \ldots, v_{s,j-1})$.

Step 4: If $i < k$, go to Step 2.

If not, if *num_tours* > 1, use this procedure to break up $sub\_tour[2]$ to $sub\_tour[num\_tours]$. If *num_tours* = 1, the given tour has no sub-tours, STOP.

Inserting the sub-tours into the outer tour

This procedure inserts the *num_tours* sub-tours into an outer tour. Each sub-tour ST_i is represented as $ST_i = (v_{s_i,1}, v_{s_i,2}, \ldots, v_{s_i,k_i})$. This procedure determines a common node between the outer tour and each one of the sub-tours, if one exists, and inserts the sub-tours starting at this common node. Note that there is at least one sub-tour having a common node with the outer tour during the first iteration. While inserting the sub-tours, we make sure that we insert the sub-tour directly into the outer tour and not into another sub-tour. We repeat the procedure until all the sub-tours are inserted into the outer tour. We present below a detailed description of the procedure. Note that we store the position on the outer tour at which a common node exists between sub-tour ST_i and the outer tour in $pos\_outer[i]$ and the corresponding position on the sub-tour ST_i in $pos\_subtour[i]$.

Step 0: Let $t_{outer} \leftarrow$ outer tour, and $A \leftarrow$ set containing the *num_tours* sub-tours. Set $pos\_outer[i] \leftarrow 0$ for $i = 1,\ldots,num\_tours$.

Step 1: For $i = 1,\ldots,num\_tours$, if a common node exists between the outer tour and sub-tour ST_i, determine its position on t_{outer} and ST_i, and update $pos\_outer[i]$ and $pos\_subtour[i]$.

Step 2: Let B be the set of all sub-tours with $pos\_outer[i]$ > 0. Set $m \leftarrow |B|$ and $j \leftarrow 1$.

Step 3: Let $k \leftarrow \underset{ST_i \in B}{\text{argmax}} \{pos\_outer(i)\}$. Start sub-tour ST_k at node $pos\_subtour[k]$ and insert it into the outer tour starting at $pos\_outer[k]$. Set $j \leftarrow j+1$, $pos\_outer[k] \leftarrow -1$.

Step 4: If $j \le m$, go to step 3. If not, go to Step 5.

Step 5: If there is one or more sub-tour ST_i with $pos\_outer[i]$ = 0, go to Step 2. If not, t_{outer} contains the final Eulerian tour, stop.

Algorithm SUB-TOUR CONSTRUCTION

Step 1: Construct the set *C* of the eligible edges for sub-tour construction using the procedure described earlier.

Step 2: For each edge $e_i \in C$, construct a sub-tour $ST_i = \left(v_0, v_i^{in}, e_i, v_i^{out}, v_0\right)$. Let *A* $\leftarrow$ set of all sub-tours.

Step 3: Calculate $E\left\lfloor S\left(ST_i, ST_j\right)\right\rfloor$ using (5) as described in Section 3.2. for every pair of sub-tours in *A*.

Step 4: Let $k \leftarrow arg\,max\left\{E\left\lfloor S\left(ST_i, ST_j\right)\right\rfloor\right\}$ and i' and j' be the indices yielding *k*.

Step 5: Concatenate sub-tours i' and j' using the appropriate orientation of the sub-tours. Add the concatenated tour to *A*. Set $A \leftarrow A - \left\{ST_{i'}, ST_{j'}\right\}$, and $|A| \leftarrow |A| - 1$.

Step 6: If $|A| = 1$, SHORTEN the giant sub-tour in set *A* and go to Step 7. If not, go to Step 3.

Step 7: Break up the giant sub-tour in set *A* into as many smaller sub-tours as possible using the procedure described earlier.

Step 8: If $E \setminus C \neq \varnothing$, construct a tour t_{outer} with the edges in $E \setminus C$ using the global greedy heuristic and SHORTEN the tour. If not, set $t_{outer} \leftarrow \{v_0\}$.

Step 9: Insert the smaller sub-tours from Step 7 into the outer tour from Step 8 using the procedure described above.

3.4. Post-Optimization: DROP-ADD

The post-optimization procedure that we use is an adaptation of the Unstringing-Stringing (US) procedure developed by Gendreau et al. (1992) for the TSP. Hertz et al. (1996) refer to this adaptation as DROP-ADD and use it for the undirected rural Postman Problem. Given an Eulerian tour t with expected length $E\left[L_t\right]$, procedure DROP-ADD attempts to find a tour t' with $E\left[L_{t'}\right] \leq E\left[L_t\right]$ by successively removing service edges from the solution and then adding them at the best possible position. This procedure uses the ADD procedure as described in Hertz et al. (1996) to insert edges into a tour. We provide a step-by-step description of the DROP-ADD procedure below.

Step 1: Let the given Eulerian tour be t with expected length $z^* = E[L_t]$. The n service edges of the tour are numbered in their order of appearance in the tour. Let $i \leftarrow 1$.

Step 2: Remove edge e_i from the tour t and SHORTEN the tour to obtain tour $\tilde{t}$.

Step 3: Add edge e_i to tour $\tilde{t}$ using procedure ADD as described in Hertz et al. (1996).

Step 4: Set $t \leftarrow \tilde{t}$. If $E[L_t] < z^*$, set $t' \leftarrow t$, $z^* \leftarrow E[L_t]$, $i \leftarrow 1$; go to Step 2.

Step 5: If $i = n$, stop. If not, set $i \leftarrow i + 1$ and go to Step 2.

4. Computational Results

We coded all the heuristics and the post-optimization procedure in C and tested them on two classes on randomly generated problems. The first class of problems consists of grid networks of varying sizes. We generated grids of sizes 4x4, 5x5, 6x6, 7x7, 8x8, and 9x9 for our computations. For each one of the problems the lengths of the horizontal edges was randomly selected in the interval [5,10] and the length of the vertical edges in the interval [4,8]. All the edges of the grid are service edges. The location of the depot was randomly selected from all the vertices. For each of the grid sizes, we generated 10 instances, thus generating 60 grid networks in all. For all the instances, we solved a matching problem to make the given grid Eulerian.

For the second class of problems, we generated a specified number of vertices (8, 10, 15, and 20 in our computations) in the $[0,10]^2$ square. We generated a first set of edges by constructing a random Hamiltonian cycle on these vertices. This ensures that the graph is connected. We then added more edges to the graph randomly until a pre-specified graph density was reached. For our computations, we generated graphs of density 0.3, 0.5 and 0.7 for each value of the number of vertices. We chose the depot as the median of all the vertices of the graph. Finally, for each combination of number of vertices and graph density, we generated 10 problem instances, thus generating 120 Euclidean graph instances in all. Here again, we solved the matching problem of each instance to make it Eulerian.

The biggest of the grid networks contains 174 service edges and the biggest Euclidean network contains 153 service edges. In a real world scenario, such as a mail delivery or a meter reading application, the SETP has to be solved for each mail carrier or meter reader separately. Under this condition, we feel that the problem sizes that we have considered are quite realistic.

For all the 180 problem instances, we obtained Eulerian tours when the probability of occurrence of a service edge, p, ranges from 0.1 to 1.0 (in steps of 0.1). For each instance, we also generated a random Eulerian tour, and calculated the expected length of this tour for the different values of p. Tables 1-6 present the results for the grid networks. For the Euclidean networks, we have presented the results for the networks with 20 vertices in Tables 7-9. Each cell in the tables contains two numbers. The first number represents the average over 10 instances of the ratio of the expected length of the tour obtained using a particular heuristic and the expected length of the random Eulerian tour. The second number gives the average over 10 instances of the time taken in seconds for that heuristic on a Sun Sparc work station.

Each row of the tables presents the average results over 10 instances for a given probability, for all three heuristics with and without the post-optimization procedure. The number in bold (for each row) indicates the heuristic with the best average result over 10 instances for that probability. For example, in Table 1, for a probability of 0.1 the first heuristic along with the DROP-ADD procedure has the best average result over 10 instances. The last row of the tables present the best result over all the 100 runs (10 instances X 10 probabilities of occurrence) for each heuristic. The following describes the contents of each column in the tables.

Column 1: Value of p

Column 2: Expected length of tour by global greedy heuristic/expected length of random Eulerian tour

Column 3: Expected length of tour by global greedy heuristic + DROP-ADD/ expected length of random Eulerian tour

Column 4: Expected length of tour by local greedy heuristic/expected length of random Eulerian tour

Column 5: Expected length of tour by local greedy heuristic + DROP-ADD/ expected length of random Eulerian tour

Column 6: Expected length of tour by sub-tour construction heuristic / expected length of random Eulerian tour

Column 7: Expected length of tour by sub-tour construction heuristic + DROP-ADD / expected length of random Eulerian tour

4.1. Effect of Procedure DROP-ADD

Our computational results indicate that the post-optimization procedure DROP-ADD is quite effective in producing new Eulerian tours with lower expected lengths. For the grid networks, the expected length of the tours produced by the global and local greedy heuristics drops by 2-4% on average (for the various values of p) after using the DROP-ADD post-optimization procedure. This decrease in the expected length is a little higher for the sub-tour construction heuristic. On a tour produced by the sub-tour construction heuristic, when a service edge is removed from the tour and has to be re-inserted during the DROP-ADD phase, there is typically more options for points of insertion of this edge into the tour. Hence, the post-optimization procedure is more effective on tours produced by the third heuristic. The improvement is the least on tours produced by the second heuristic. For the Euclidean networks, the decrease in the expected length is between 1% and 3% for all three heuristics. The tours produced by the three heuristics for the Euclidean networks are quite similar and hence the effect of DROP-ADD is similar too.

4.2. Performance of the Heuristics

Our results indicate that when the probability of occurrence of the service edges is very low ($p = 0.1$), the global greedy heuristic along with DROP-ADD seems to perform better than the other two heuristics. For other values of p, the sub-tour construction heuristic along with DROP-ADD clearly seems to perform better than the other two heuristics for grid networks. For the Euclidean networks, as the number of edges increases, the local greedy heuristic along with DROP-ADD seems to perform the best among the three, though the margin of improvement is very small. For problems with smaller number of service edges, the best results seem to be spread among the three heuristics for the various values of p .

The results on our computational times indicate that all the three heuristics are quite fast. For the largest problems (9x9 grid networks, and 20 node Euclidean networks; edge density of 0.7), all the heuristics generate tours in about a minute. So we can generate tours using all the three heuristics and choose the tour with the best expected length.

4.3. Comparison of Heuristic Tours with Random Tour

We also compared the expected lengths of the tours produced by the three heuristics with the expected length of a random Eulerian tour. Table 10 contains average results for the grid and Euclidean networks. For a given probability, we pick the results produced by the heuristic with the best average result over 10 instances for each problem size, and compute the overall average best result over all the problem sizes using these results. Our results show that for the grid networks, the expected length of our overall average best solution is lower than the expected length of a random tour by 10% on average, for lower values of p . As p increases to 1.0, this average reduces to 6% for $p = 0.5$ and 1% for $p = 0.9$. In one particular 9x9 instance, the expected length of the tour generated by the sub-tour construction heuristic is 25% lower than the expected length of the random Eulerian tour. For the Euclidean networks, the gaps are not as dramatic. The expected length of our overall average best solution is lower than the expected length of a random tour by 2% on average, for low values of p. These results clearly show that it is advantageous to use heuristics designed specifically for the SETP rather than generate a random Eulerian tour.

5. Conclusion

In this paper, we have presented three different heuristics for the SETP and the DROP-ADD post-optimization procedure. We have tested the performance of the three heuristics on grid networks and on Euclidean graphs of various sizes. Our results indicate that the sub-tour construction heuristic performs well when $p > 0.1$ for the grid networks. For the Euclidean networks, as the number of edges increases, the second heuristic performs the best among the three, though the margin of improvement is small.

We also compared the expected lengths of the tours produced by the three heuristics with the expected length of a random Eulerian tour. Our results show that for the grid networks, the expected length of our overall average best solution is lower than the expected length of a random tour by 10% on average, for lower values of p. As p increases to 1.0, this average reduces to 6% for $p = 0.5$ and 1% for $p = 0.9$. For the Euclidean networks, the expected length of our overall average best solution is lower than the expected length of a random tour by 2% on average, for low values of p. This paper serves as a good start to the methodological contribution to the SETP and the results can definitely be used as a starting point for the development of meta-heuristics for the SETP.

References

Bertsimas, D. J. 1988. *Probabilistic combinatorial optimization problems*. Ph. D. Thesis, Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA.

Bertsimas, D. J. and L. H. Howell. 1993. Further results on the probabilistic traveling salesman problem. *Eur. J. Oper. Res.* **65** 68-95.

Gendreau, M. A. Hertz and G. Laporte. 1992. New Insertion and Postoptimisation procedures for the traveling salesman problem. *Oper. Res.* **40** 1086-1094.

Hertz, A., G. Laporte, and P. Nanchen. 1999. Improvement Procedures for the Undirected Rural Postman Problem. *INFORMS J. on Computing* **11** 53-62.

Jaillet, P. 1985. *Probabilistic traveling salesman problem*. Ph. D. Thesis, Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA.

Jaillet, P. 1988. A priori solution of a traveling salesman problem in which a random subset of customers are visited. *Oper. Res.* **36** 929-936.

Laporte, G., F. V. Louveaux, and H. Mercure. 1989. Models and exact solutions for a class of stochastic location-routing problems. *Eur. J. Oper. Res.* **39** 71-78.

Rossi, F. A. and I. Gavioli. 1988. Aspects of heuristic methods in the probabilistic traveling salesman problem. G. Andreatta, F. Mason and P.Serafini Eds. *Advanced School on Stochastics in Combinatorial Optimization*. World Scientific, Singapore.

Prob.	Heur 1	H1+DA	Heur 2	H2+DA	Heur 3	H3+DA
0.1	1.0079	**0.9799**	1.0051	0.9917	1.0133	0.9868
	0.01	0.24	0.00	0.08	0.01	0.12
0.2	1.0128	**0.9524**	1.0108	0.9799	1.0200	0.9735
	0.01	0.20	0.00	0.09	0.01	0.11
0.3	1.0142	**0.9543**	1.0130	0.9725	1.0097	0.9707
	0.01	0.15	0.00	0.08	0.01	0.10
0.4	1.0039	0.9570	1.0133	0.9703	1.0055	**0.9514**
	0.01	0.17	0.00	0.09	0.01	0.11
0.5	1.0485	0.9768	1.0131	**0.9673**	1.0233	0.9676
	0.01	0.13	0.00	0.08	0.01	0.09
0.6	1.0761	0.9848	1.0123	0.9718	1.0282	**0.9619**
	0.01	0.18	0.00	0.09	0.01	0.10
0.7	1.0759	1.0014	1.0105	0.9779	1.0243	**0.9690**
	0.01	0.13	0.00	0.09	0.01	0.10
0.8	1.0831	1.0015	1.0077	0.9847	1.0200	**0.9787**
	0.01	0.15	0.00	0.08	0.01	0.10
0.9	1.0872	0.9935	1.0040	0.9921	1.0172	**0.9878**
	0.01	0.12	0.00	0.08	0.01	0.10
1.0	1.0722	**1.0000**	**1.0000**	**1.0000**	1.0058	**1.0000**
	0.00	0.09	0.00	0.06	0.01	0.06
Best	0.92	0.89	0.91	0.88	0.93	0.88

Table 1. Results for the 4x4 grid network

Prob.	Heur 1	H1+DA	Heur 2	H2+DA	Heur 3	H3+DA
0.1	0.9506	**0.9395**	0.9836	0.9551	1.0045	0.9542
	0.04	1.04	0.00	0.74	0.07	0.78
0.2	0.9524	0.9117	0.9767	0.9341	0.9776	**0.9074**
	0.03	0.81	0.00	0.77	0.08	0.79
0.3	0.9916	0.9117	0.9793	0.9305	1.0123	**0.8939**
	0.03	0.69	0.00	0.72	0.08	0.94
0.4	1.0385	0.9412	0.9856	0.9419	0.9846	**0.9280**
	0.04	0.75	0.00	0.70	0.08	0.89
0.5	1.0383	0.9465	0.9922	0.9515	1.0150	**0.9437**
	0.03	0.95	0.00	0.68	0.08	0.85
0.6	1.0401	0.9565	0.9973	0.9600	1.0243	**0.9466**
	0.03	0.78	0.00	0.68	0.08	0.93
0.7	1.0458	0.9755	1.0004	0.9717	1.0315	**0.9583**
	0.03	0.78	0.01	0.66	0.08	0.70
0.8	1.0632	0.9832	1.0016	0.9804	1.0179	**0.9657**
	0.03	0.75	0.00	0.74	0.10	0.79
0.9	1.0635	0.9892	1.0012	0.9905	1.0075	**0.9796**
	0.03	0.81	0.00	0.71	0.10	0.77
1.0	1.0473	**1.0000**	**1.0000**	**1.0000**	1.0307	**1.0000**
	0.02	0.40	0.00	0.26	0.06	0.41
Best	0.91	0.85	0.84	0.77	0.84	0.77

Table 2. Results for the 5x5 grid network

Prob.	Heur 1	H1+DA	Heur 2	H2+DA	Heur 3	H3+DA
0.1	0.9188	**0.9165**	0.9828	0.9712	1.0048	0.9500
	0.11	3.91	0.01	0.77	0.33	2.62
0.2	0.9329	**0.9143**	0.9680	0.9534	0.9918	0.9232
	0.11	3.02	0.01	0.85	0.34	2.38
0.3	1.0003	0.9283	0.9691	0.9546	0.9430	**0.8852**
	0.11	3.23	0.00	0.88	0.37	2.51
0.4	1.0106	0.9479	0.9776	0.9622	0.9796	**0.9179**
	0.11	2.68	0.00	0.89	0.38	2.28
0.5	1.0452	0.9667	0.9865	0.9720	1.0066	**0.9441**
	0.11	2.29	0.00	0.89	0.39	1.98
0.6	1.0545	0.9683	0.9931	0.9805	1.0251	**0.9605**
	0.11	2.84	0.00	0.88	0.39	2.44
0.7	1.0629	0.9809	0.9969	0.9869	1.0326	**0.9787**
	0.11	3.05	0.00	0.89	0.40	2.01
0.8	1.0607	0.9946	0.9986	0.9894	1.0311	**0.9892**
	0.11	3.23	0.01	0.94	0.40	2.05
0.9	1.0646	1.0022	0.9992	**0.9945**	1.0296	0.9952
	0.11	3.09	0.01	0.96	0.41	1.54
1.0	1.0646	1.0191	**1.0000**	**1.0000**	1.0084	**1.0000**
	0.06	0.89	0.00	0.63	0.25	0.78
Best	0.86	0.86	0.91	0.88	0.88	0.84

Table 3. Results for the 6x6 grid network

Prob.	Heur 1	H1+DA	Heur 2	H2+DA	Heur 3	H3+DA
0.1	0.8778	**0.8680**	0.9543	0.9206	0.9746	0.9335
	0.31	9.55	0.02	4.88	1.02	5.81
0.2	0.9224	0.9043	0.9399	0.9056	0.9684	**0.8956**
	0.31	8.43	0.02	4.91	1.10	7.93
0.3	0.9907	0.9145	0.9492	0.9174	0.9760	**0.8942**
	0.30	8.18	0.02	4.74	1.14	7.94
0.4	0.9835	0.9388	0.9654	0.9351	0.9852	**0.9027**
	0.30	6.53	0.02	5.29	1.17	9.69
0.5	1.0172	0.9632	0.9805	0.9520	1.0095	**0.9264**
	0.31	6.73	0.02	5.70	1.21	10.27
0.6	1.0669	0.9750	0.9915	0.9674	1.0279	**0.9558**
	0.30	7.84	0.02	6.21	1.24	9.64
0.7	1.0517	0.9882	0.9982	0.9795	1.0462	**0.9761**
	0.29	6.30	0.02	6.51	1.26	9.61
0.8	1.0482	0.9853	1.0011	0.9891	1.0624	**0.9788**
	0.29	8.16	0.02	5.97	1.29	9.04
0.9	1.0530	0.9929	1.0013	**0.9952**	1.0354	0.9953
	0.30	8.97	0.02	6.04	1.31	8.28
1.0	1.0597	1.0053	**1.0000**	**1.0000**	1.0444	1.0004
	0.16	3.37	0.02	1.73	0.78	3.36
Best	0.79	0.79	0.88	0.83	0.86	0.79

Table 4. Results for the 7x7 grid network

Prob.	Heur 1	H1+DA	Heur 2	H2+DA	Heur 3	H3+DA
0.1	0.8801	**0.8643**	0.9337	0.9216	1.0118	0.9060
	0.72	22.05	0.03	6.13	2.61	19.55
0.2	0.9065	0.8945	0.9169	0.9045	0.9582	**0.8839**
	0.71	16.15	0.03	6.24	2.94	18.17
0.3	0.9395	0.9127	0.9336	0.9218	0.9716	**0.8850**
	0.72	18.12	0.03	6.24	3.02	19.92
0.4	1.0083	0.9462	0.9553	0.9433	0.9652	**0.9133**
	0.71	12.12	0.03	6.31	3.12	15.49
0.5	1.0246	0.9649	0.9733	0.9610	0.9989	**0.9371**
	0.72	13.93	0.03	6.71	3.25	19.43
0.6	1.0338	0.9759	0.9861	0.9756	1.0184	**0.9609**
	0.69	16.07	0.03	6.76	3.26	19.51
0.7	1.0510	0.9930	0.9940	0.9858	1.0264	**0.9752**
	0.69	18.59	0.03	6.96	3.33	15.36
0.8	1.0581	1.0018	0.9998	0.9922	1.0358	**0.9921**
	0.69	16.35	0.03	6.99	3.43	18.38
0.9	1.0484	1.0089	0.9995	**0.9964**	1.0507	1.0059
	0.69	16.76	0.03	6.92	3.49	17.42
1.0	1.0478	1.0212	**1.0000**	**1.0000**	1.0254	1.0048
	0.35	5.21	0.02	3.47	1.95	4.56
Best	0.85	0.84	0.85	0.84	0.84	0.79

Table 5. Results for the 8x8 grid network

Prob.	Heur 1	H1+DA	Heur 2	H2+DA	Heur 3	H3+DA
0.1	0.8396	**0.8387**	0.8892	0.8581	0.9624	0.8637
	1.50	47.98	0.05	35.00	6.37	69.63
0.2	0.8620	0.8539	0.8809	**0.8491**	0.9534	0.8664
	1.49	38.75	0.05	33.56	6.01	42.08
0.3	0.9646	0.8956	0.9137	**0.8780**	0.9743	0.8832
	1.51	38.91	0.04	35.44	7.02	54.01
0.4	0.9957	0.9371	0.9476	**0.9096**	1.0194	0.9352
	1.47	35.34	0.04	44.31	7.13	53.33
0.5	1.0199	0.9560	0.9732	**0.9393**	1.0637	0.9628
	1.51	32.68	0.04	47.02	7.55	57.29
0.6	1.0393	0.9711	0.9901	**0.9606**	1.0688	0.9787
	1.46	54.79	0.06	43.07	7.48	55.16
0.7	1.0440	0.9824	0.9995	**0.9766**	1.0725	0.9816
	1.46	39.21	0.04	57.72	7.67	46.09
0.8	1.0674	0.9959	1.0025	**0.9870**	1.1342	1.0370
	1.46	48.79	0.05	58.07	7.84	47.35
0.9	1.0637	0.9979	1.0019	**0.9940**	1.1345	1.0224
	1.46	49.89	0.05	38.67	8.05	62.26
1.0	1.0675	**1.0000**	**1.0000**	**1.0000**	1.0945	1.0105
	0.72	14.02	0.03	7.54	5.01	30.73
Best	0.79	0.79	0.83	0.77	0.86	0.75

Table 6. Results for the 9x9 grid network

Prob.	Heur 1	H1+DA	Heur 2	H2+DA	Heur 3	H3+DA
0.1	0.9959	**0.9676**	0.9991	0.9691	0.9934	0.9699
	0.09	2.30	0.00	2.04	0.63	3.08
0.2	0.9978	0.9664	0.9964	**0.9603**	0.9963	0.9635
	0.09	2.67	0.01	2.29	0.66	2.64
0.3	1.0032	0.9635	0.9931	0.9619	0.9973	**0.9588**
	0.10	3.09	0.01	2.37	0.68	2.91
0.4	1.0069	0.9766	0.9910	**0.9664**	0.9984	0.9734
	0.09	2.52	0.01	2.82	0.69	2.48
0.5	1.0079	0.9724	0.9905	**0.9703**	1.0014	0.9717
	0.10	2.04	0.00	2.46	0.82	2.61
0.6	1.0114	0.9788	0.9914	**0.9733**	1.0071	0.9782
	0.08	2.20	0.01	2.31	0.78	2.13
0.7	1.0083	0.9817	0.9932	**0.9793**	1.0068	0.9828
	0.09	2.00	0.01	1.99	0.81	2.38
0.8	1.0127	0.9867	0.9956	**0.9851**	1.0185	0.9881
	0.10	2.49	0.00	1.76	0.84	2.43
0.9	1.0161	0.9922	0.9980	**0.9917**	1.0166	0.9946
	0.10	2.83	0.01	1.80	0.86	2.35
1.0	1.0166	**1.0000**	**1.0000**	**1.0000**	1.0307	1.0006
	0.05	1.06	0.00	0.72	0.39	1.12
Best	0.97	0.94	0.98	0.94	0.97	0.92

Table 7. Results for 20 node Euclidean network (density – 0.3)

Prob.	Heur 1	H1+DA	Heur 2	H2+DA	Heur 3	H3+DA
0.1	1.0001	0.9788	1.0000	**0.9746**	0.9993	0.9796
	0.40	11.02	0.04	14.16	2.90	13.55
0.2	1.0013	0.9770	0.9980	0.9757	0.9962	**0.9756**
	0.41	12.61	0.03	12.53	2.99	11.26
0.3	1.0036	0.9762	0.9960	**0.9739**	1.0022	0.9749
	0.43	14.81	0.03	16.43	3.14	14.34
0.4	1.0001	0.9796	0.9948	**0.9769**	0.9967	0.9783
	0.40	11.23	0.03	14.63	3.39	14.97
0.5	1.0029	0.9833	0.9948	**0.9818**	1.0020	0.9847
	0.41	11.20	0.04	10.19	3.86	12.31
0.6	1.0038	0.9874	0.9956	**0.9866**	1.0036	0.9870
	0.40	12.46	0.03	9.18	3.66	12.55
0.7	1.0049	0.9914	0.9970	**0.9881**	1.0091	0.9911
	0.41	10.07	0.04	9.85	3.83	11.95
0.8	1.0047	0.9937	0.9986	**0.9919**	1.0095	0.9958
	0.41	11.58	0.03	11.92	3.93	12.26
0.9	1.0069	0.9963	0.9999	**0.9960**	1.0104	0.9986
	0.41	13.91	0.04	11.83	4.02	11.15
1.0	1.0061	1.0006	**1.0000**	**1.0000**	1.0176	1.0021
	0.21	4.32	0.02	3.09	1.76	3.73
Best	0.99	0.96	0.99	0.96	0.98	0.96

Table 8. Results for 20 node Euclidean network (density – 0.5)

Prob.	Heur 1	H1+DA	Heur 2	H2+DA	Heur 3	H3+DA
0.1	1.0008 1.12	0.9813 32.75	0.9917 0.08	**0.9684** 38.08	0.9923 6.30	0.9764 34.27
0.2	1.0000 1.12	0.9761 35.44	0.9845 0.08	**0.9665** 29.97	0.9947 6.31	0.9720 43.79
0.3	0.9936 1.14	0.9718 47.44	0.9810 0.08	**0.9676** 35.85	0.9962 6.54	0.9719 49.35
0.4	0.9962 1.12	0.9771 36.66	0.9807 0.08	**0.9697** 32.38	0.9963 6.68	0.9782 33.26
0.5	0.9969 1.13	0.9806 31.46	0.9826 0.08	**0.9758** 24.82	0.9953 6.97	0.9811 35.93
0.6	0.9985 1.12	0.9877 26.96	0.9861 0.08	**0.9804** 29.14	0.9987 7.66	0.9854 34.41
0.7	0.9996 1.11	0.9899 37.92	0.9906 0.08	**0.9867** 25.19	1.0054 8.05	0.9906 36.07
0.8	1.0013 1.12	0.9941 25.96	0.9951 0.08	**0.9920** 29.53	1.0060 8.46	0.9951 27.70
0.9	1.0032 1.12	0.9978 28.37	0.9988 0.08	**0.9969** 30.65	1.0126 9.14	0.9977 30.58
1.0	1.0024 0.56	**1.0000** 9.89	**1.0000** 0.04	**1.0000** 7.64	1.0170 4.80	1.0004 10.93
Best	0.98	0.96	0.97	0.95	0.98	0.96

Table 9. Results for 20 node Euclidean network (density – 0.7)

Prob.	Grid	Euclidean		
		0.3	**0.5**	**0.7**
0.1	0.9013	0.9775	0.9797	0.9800
0.2	0.9005	0.9706	0.9728	0.9742
0.3	0.8984	0.9676	0.9712	0.9734
0.4	0.9205	0.9701	0.9749	0.9753
0.5	0.9430	0.9699	0.9775	0.9794
0.6	0.9577	0.9731	0.9808	0.9819
0.7	0.9723	0.9770	0.9839	0.9865
0.8	0.9819	0.9842	0.9880	0.9909
0.9	0.9913	0.9917	0.9933	0.9953
1.0	1.0000	1.0000	1.0000	1.0000

Table 10. Overall average of best results for grid and Euclidean networks