CHAPTER 10

COMPUTATIONAL STRATEGIES FOR OPTIMIZATION OF MAIL DISTRIBUTION PATH

10.0 INTRODUCTION

The optimal path for delivery of postal mail along with the distribution schedule can be generated by processing the GIS data representation of the 'beat' of the postman and the list of the points to which mail is to be delivered. The computational strategies for generation of the optimal path for distribution should make use of the knowledge about the road network of the postman beat and the geographic spread of the active mail delivery points (delivery points to which mail is to be delivered on the given day). The problem of finding an optimal path for distribution of postal mail to a set of mail delivery points has been described as Rural Postman Problem: Delivery Points (RPP:DP) in Chapter 9 and is supposed to be an NP-Hard problem.

Some softwares are available for generation of optimal path for distribution of postal mail using GIS representation for structured localities of countries available in countries such as Canada, Finland etc [W-16;W-17;W-18]. No technical details of the implementation are available in public domain. Also there is no reported work for generation of optimal path for distribution of postal mail in the Indian context as discussed in chapter 9. This motivated us to develop some novel intelligent computational strategies for optimization of mail distribution path and generation of mail distribution schedule considering the peculiarities of the Indian System. Although solution to RPP:DP is not found in literature, the corresponding conventional variant of the problem, namely the Rural Postman Problem (RPP) has been thoroughly investigated and is found to be NP hard.

A paper entitled “Algorithms for Finding the Optimal Path for Distribution of Postal Mail” is under preparation
My Research Guide Dr. P Nagabhushan has been invited to give a talk on “Graph Theory Application to Optimization of Distribution Path for Postal Mail” at RVCE, Bangalore
Exact and approximate solutions are proposed for some instances of Rural Postman Problem (RPP). Christofides et al, (1981) have proposed an exact algorithm using the integer programming formulation to solve the RPP. This algorithm is computationally inefficient and has an exponential time complexity and can be used for graphs having very small number of vertices. Christofides et al, (1981) have also proposed a heuristic procedure for solving the rural postman problem. The heuristic procedure provides an approximate solution to the RPP and consists of three phases; the first phase referred to as Graph Transformation phase, converts the graph representing the road network where the mail is to be distributed into a complete graph (every vertex is connected to every other vertex), which is further reduced to remove duplicate edges. The second phase referred to as the Minimal Spanning Tree phase, generates a minimum spanning tree connecting all the components of the road network where mail is to be distributed. The third stage referred to as Minimal Cost Matching phase, uses the EDMONDS (Edmonds and Johnson, 1973) minimal cost-matching algorithm for pairing of odd degree vertices in the graph containing edges where the mail is to be distributed and the edges of the minimum spanning tree. This pairing implies addition of an imaginary edge between the pair of odd degree vertices to convert the odd degree vertex to even degree. In practice the imaginary edge is replaced by the edges in the shortest path between the pair of odd degree vertices. This process of addition of the edges for every pair of odd degree vertices converts the graph where mail is to be distributed to an Euler graph/ Unicursal graph (Christofides et al, 1981). The Euler graph thus produced is processed to find the minimal cost closed walk for distribution of mail using Fleury’s algorithm (Ahr, 2000), which is described in Chapter 9.

Pearn and Wu (1994) propose modifications to the Christofides heuristic algorithm to obtain a more efficient procedure for finding the optimal path for distribution of postal mail. Pearn and Wu (1994) suggest the use of a static penalty value for evaluation of the minimum spanning tree that connects the components of the graph and different ordering of the phases in Christofides et al, algorithm to obtain an efficient solution. There are other heuristic solutions proposed to solve the RPP. Groves and VanVuuren (2005) present a local search framework for solving the Undirected Rural Postman Problem (URPP). The local search framework is based on the two OPT and three OPT procedure for solving the
Traveling Salesperson Problem (TSP). An approximate heuristic procedure to solve the rural postman problem by Memetic algorithms is found in (Rodrigues and Ferreira, 2001). The procedure makes use of recombination and local search to improve the current solutions by organizing the possible solutions in a ternary tree like structure. Kang and Hee, (1998) propose a genetic algorithm and graph transformation approach to solve the RPP. The methodology works by converting the graph representing the RPP to a Hamiltonian graph, and solves the problem as a TSP.

Most of the heuristic procedures found in literature employ different strategies to implement the three phases of the Christofides et al, algorithm. Some of the procedures combine these phases in generation of optimal solution to RPP. In summary any solution to RPP involves converting an unconnected graph representing the roads/edges, which have mail to be delivered, to a connected graph. The connected graph is further transformed to Euler graph by minimal cost matching of odd degree vertices. A closed walk representing a Euler trail starting and ending on the node representing the Mail Delivery Post Office (MDPO) gives the optimal path for distribution of postal mail.

The problem of finding the optimal route for distribution of mail to a list of points to which mail is to be delivered, namely RPP:DP requires combining strategies to solve the edge covering problems such as RPP and the strategies for solving the node covering problems such as Traveling Sales Person Problem (TSP). In this chapter we present intelligent computational strategies to process the list of points to which mail is to be delivered with respect to the GIS representation of the postman beat to generate the optimal path for distribution of postal mail and the mail distribution schedule. The solution methodology is broadly based on the solution outlined in algorithm 9.3.

The solution to RPP:DP is found by devising novel computational strategies to implement the modules M1 to M4 depicted in Figure 9.3, which provides the overview of the solution. The functionality of module M1 namely, generation of the mail distribution graph is implemented by scanning the edgewise delivery point representation of the postman beat, for the list of points to which mail is to be delivered and generating a list of edges which are to be traversed completely (from one end point to the other) and a list of edges that are
to be partially traversed (starting and returning to the same end point of the road). The functionality of the module M2, i.e., conversion of mail distribution graph to a connected graph is implemented by using a newly devised graph transformation technique employing the Spatial Data Structure (SDS-PB) representing the beat of the postman where mail is distributed. The functionality of the module M3 namely converting non-eulerian graph to eulerian graph is implemented either by using EDMONDS matching algorithm or by making use of a steady state genetic algorithm using newly devised chromosome representation and operators. The functionality of module M4 namely generation of the optimal route and the distribution schedule is implemented using a Fleury’s algorithm and an intelligent computational strategy using active mail delivery points and the list of edges that need partial traversal. The computational strategies have been tested on practical beats of postman with different mail delivery loads.

The computational strategies developed for optimization of mail distribution path and generation of the distribution schedules are explained in this chapter. The remaining part of the chapter is divided into seven sections. Section 10.1 describes the computational strategy for generation of the Mail Distribution Graph. Section 10.2 describes the graph transformation approach to generation of the connected graph from the mail distribution graph. Computational strategies based on Genetic Algorithm and EDMONDS algorithm, to generate an Euler graph from a non-Eulerian connected graph are described in section 10.3. Section 10.4 describes the computational strategy for generation of the optimal path for distribution of mail from the Euler graph and also the mail distribution schedule. The complete procedure of generation of the optimal path for distribution of postal mail and distribution schedule is illustrated in section 10.5 using a case study. Section 10.6 provides experimental analysis and section 10.7 gives the conclusions.

10.1 GENERATION OF MAIL DISTRIBUTION GRAPH

A Mail Distribution Graph (MDG) is a sub graph of the graph representing the road network of the locality where the mail is to be distributed. The MDG is a collection of edges and vertices that are to be necessarily visited to distribute the mail on a given day. The sub graph is generated by processing the list of delivery points to which the mail is to
be distributed (active mail delivery points). The procedure selects all the edges / roads which have mail to be distributed on the given day, from the edges of the graph representing the road network of the locality. Each of the edges so selected is checked to find whether it is to be completely traversed or partially traversed. This is accomplished by comparing the cost of complete traversal or partial traversal and selecting the one with the least cost. The MDG is represented as a list of edges (along with the corresponding end vertices) and a list of vertices from which a partial edge traversal is to be carried out to distribute the mail. The complete procedure employed for generation of MDG is outlined in algorithm 10.1. The MDG consists of a list of edges that require complete traversal (eMDG) and a list of vertices and edges that require partial traversal (peMDG).

Algorithm 10.1: Generation of Mail Distribution Graph (MDG)

Input: List of the delivery points, L to which mail is distributed and GIS data representation of the locality where the mail is to be distributed

Output: Mail Distribution Graph: A set of edges which are to be completely traversed to distribute mail (eMDG) and a set of vertices along with the edges which are to be visited to distribute mail by partial traversal of edge (peMDG). peMDG also consists of the active mail delivery points for each partial edge

1. For every delivery point in L find the edge on which it occurs and find the non repeating list of edges which have active delivery points (delivery points to which mail is to be distributed). Let E be the number of non-repetitive edges which need to be traversed in the partial Mail Distribution Graph (pMDG)

2. For i=1 to E do
   a. For i\textsuperscript{th} edge in pMDG find the cost of complete traversal (C\textsubscript{i}), partial traversal from end point\textsubscript{1} (C\textsubscript{21}) and partial traversal from endpoint2 (C\textsubscript{22}).
   b. If C\textsubscript{i} is less than C\textsubscript{21} and C\textsubscript{22} then the edge is to be completely traversed, process the next edge
   c. If C\textsubscript{21} is less than C\textsubscript{i} and C\textsubscript{22} then the edge i\textsuperscript{th} is to be partially traversed from end point\textsubscript{1}, remove the i\textsuperscript{th} edge from pMDG and add end point\textsubscript{1}, i\textsuperscript{th} edge and mail delivery points to be visited to a list of vertices with partial edges, peMDG.
   d. If C\textsubscript{22} is less than C\textsubscript{i} and C\textsubscript{21} then the edge i\textsuperscript{th} is to be partially traversed from end point2, remove the i\textsuperscript{th} edge from pMDG and add end point2, i\textsuperscript{th} edge and mail delivery points to be visited to a list of vertices with partial edges, peMDG

3. End For

4. Copy the remaining edge in pMDG to eMDG, this gives the list of edges that are to be completely traversed

5. peMDG consists of list of vertices in the graph which are to be visited necessarily so that partial edge is traversed
The algorithm produces a list of edges and vertices, which are to be necessarily visited to distribute the mail on the given day. These lists describe a sub graph of the graph representing the road network of the locality. To generate an optimal path for distribution of mail this graph is to be a connected graph including the node representing the mail delivery post office (MDPO). A new graph transformation procedure for generating a connected graph from MDG is described in section 10.2.

10.2 GENERATION OF CONNECTED GRAPH

The mail distribution graph generated using the algorithm 10.1 for distribution of mail to a list of delivery points may be unconnected having more than one connected component. Generation of a closed walk starting from the MDPO node and visiting all the mail delivery points and returning to MDPO node requires the graph to be connected. Further this connected graph is to be generated by adding edges with least possible cost. In this work a new computational procedure that employs the spatial data structure for postman beat (SDS-PB) as the basis for generation of such a connected graph is devised. The SDS-PB is a spatial tree data structure that enlists the minimum cost path from MDPO to all the end points/junction points in the postman beat. The process of connected graph generation is described in the following.

All the vertices/junction points of the graph representing the postman beat and the edges of the MDG that need complete traversal (eMDG) are processed using Depth First Search (DFS) technique to find the connected components in the graph. Every vertex in the graph is labeled according to the component to which it belongs. Further the edges of the mail distribution graph that need complete traversal (eMDG) and the vertices that need to be visited to traverse the partial edges (peMDG), for distribution of mail are processed to find vertices that are required to be visited to distribute the mail. The required vertices and the components of the graph are processed to find the components that are to be necessarily visited to distribute the mail. These required components are to be merged into a single component by adding edges from the original graph for generation of the optimal path for distribution of postal mail.
The merging of the required components into a single component is done in two stages. In the first stage the system checks to find whether addition of an edge from spatial data structure SDS-PB joins two components. If so the corresponding edge is added to the graph to merge the components. All the edges of SDS-PB are checked in non-decreasing order of their length for inclusion. If during this process all the required components merge into one then further checking is abandoned and the second stage is not needed as we have a connected graph. In many practical mail distribution graphs the connected graph is generated using the first stage only. But if more than one component that requires to be visited remains after considering all the edges in SDS-PB, then the merging of the required components is done using the second stage.

In the second stage a shortest path between all pairs of remaining components with the characteristic that all the edges in the shortest path are from SDS-PB is found if it exists and the edges of the shortest path are added to merge the remaining components. The edges that are added in the first and second stage are processed to remove duplicates if any. This list of added edges generates a connected graph such that each component is connected to the MDPO node through the shortest path.

This strategy may pick a sub optimal path in joining components but as the connected graph so generated has the least path to every component from the MDPO the Euler trail that is generated subsequently will necessarily have the least cost path. The complete procedure for generation of the connected graph from the mail distribution graph is outlined in algorithm 10.2.

**Algorithm 10.2 : Generation of Connected Graph**

**Input:** Mail Distribution Graph (eMDG, peMDG), SDS-PB

**Output:** The Connected graph as a list of edges (CGRAPH), modified peMDG

1. Generate a list of vertices required to be visited (RV) by considering MDPO, vertices in peMDG and the endpoints of the edges in eMDG.
2. Generate the components of the mail distribution graph considering edges in eMDG and all the vertices and store the vertices belonging to each component as a separate set in COMPS.
3. Find the components which are required to be visited to distribute the given days mail by comparing RV and COMPS.
Optimization Of Mail Distribution Path 279

4. Arrange the edges of SDS-PB in non descending order

5. While all the edges in SDS-PB are not considered and all the required components are not merged into one do
   a. Take the first/next edge in SDS-PB and check whether it joins two required components, if so include the edge in Additional_Edges list and merge the components.

6. EndWhile

7. If there are more than one remaining components which are required to be visited
   a. Enlist the required components which are to be merged to produce a single component
   b. Find the shortest path using only the edges in SDS-PB between all possible pairs of remaining required components if it exists and add such edges to a temporary variable Auxiliary_Edges

8. endif

9. Combine the edges in eMDG, Additional_Edges and Auxiliary_Edges into a list CGRAPH of non repeating edges by removing duplicates if any. (CGRAPH consists of the edges of the connected graph)

10. If CGRAPH consists of any edges present in peMDG then remove such edges from peMDG as these edges will be completely traversed in the distribution path and there is now no need for their partial traversal.

The algorithm 10.2 generates a connected graph CGRAPH and also verifies whether some of the edges that needed partial traversal in the mail distribution graph now are being traversed completely because of addition of edges, if so such edges are removed from the list of edges that need partial traversal for delivery of mail.

The transformation of the mail distribution graph to a connected graph is the most critical step in generation of optimal path for distribution of mail and depends on the structure of the mail distribution graph. The use of the spatial data structure and the process of checking the insertion of single edge to join the required components facilitates a quick generation of connected graph. Even the worst-case scenario when the components are joined using the shortest distance path between two components with all the edges in SDS-PB is a faster technique as compared to the complete enumeration and finding the least cost connected graph.

This strategy will always yield a connected graph as the SDS-PB itself is connected and any component can be connected to any other component using only the edges in SDS-PB. Further as the edges added are only from SDS-PB only minimum cost edges get added resulting in a minimum cost connected graph for generation of the optimal path for distribution of mail. The connected graph generated using this technique is more suitable
for finding the least cost closed walk that starts and ends on the MDPO node, which is the root node of SDS-PB.

10.3 GENERATION OF EULER GRAPH

The connected graph generated using algorithm 10.2 is used to find the optimal path for distribution of mail. This requires finding the Euler circuit that starts and ends on the MDPO node. An Euler circuit can be found in a graph if the graph is Eulerian i.e., all the nodes of the graph are of even degree. Hence, if the graph is non-Eulerian there is a need to convert it to an Eulerian graph.

The non-Eulerian graph is converted to an Euler graph by converting the odd degree vertices to even degree vertices. The conversion involves optimal pairing of odd degree vertices and adding an imaginary edge between each pair of vertices. The imaginary edge stands for the shortest path between the endpoints using the edges in the original graph.

The complete computational procedure employed in converting the non-Eulerian graph to Eulerian graph is described in algorithm 10.3.

Algorithm 10.3: Generation of Euler Graph

Input: \textit{peMDG}, Connected graph (CGRAFI) and GIS data representation of the beat of the postman where the mail is distributed (PGIS-DATA)

Output: Euler graph (EGRAFI) containing edges which are required to be visited to distribute mail, with the property that every vertex is of even degree, modified \textit{peMDG}

1. Generate a list of edges/roads of the graph representing the beat of the postman where mail is distributed from PGIS-DATA, also list the distance/cost of each edge.
2. Find the shortest distance between every pair of vertices in the graph.
3. Find the degree of each vertex of the connected graph (CGRAFI) and list all the odd degree vertices.
4. Enlist all the possible pairing of odd degree vertices along with the shortest path/distance between each pair.
5. Find minimal cost match for the odd degree vertices using a Genetic Algorithm/EDMONDS exact algorithm.
6. Find the edges which are to be added (corresponding to the shortest path between paired odd degree vertices) in \textit{Additional_Euler_Edges}.
7. Combine the edges in CGRAFI and \textit{Additional_Euler_Edges} to generate a list of edges representing an Euler graph (EGRAFI) which is to be traversed to find the optimal path for distribution of postal mail.
8. If EGRAFI consists of any edges present in \textit{peMDG} then remove such edges from \textit{peMDG}. 
The most important step of the algorithm 10.3 is the step 5, which requires finding the minimal cost matching of odd degree vertices so as to convert them to even degree vertices. The matching can be carried out using the EDMONDS graph matching algorithm (Edmonds and Johnson, 1973) or the newly devised steady state genetic algorithm (GA). The steady state GA employed for minimal cost matching of odd degree vertices is described in section 10.3.1 and the EDMONDS algorithm is described in section 10.3.2.

10.3.1 Genetic Algorithm for Matching Odd degree vertices

Genetic Algorithm (GA) is an evolutionary computational strategy that imitates the evolution of living beings. Surprisingly the emulation of the evolutionary process gives very good results in solving many computational tasks. Genetic algorithms use two processes for evolution of a population of possible solutions (or individuals) namely Inheritance and Competition or Survival of the Fittest. The process of evolution results in preserving of the best features in subsequent generations and weeding out of unfit individuals. The fittest individual at the termination of the evolution by some predefined criterion is the solution offered by the GA. Genetic Algorithms have been devised to solve many optimization problems. Mousayama et al, (2006), present a genetic algorithm using an orthogonal crossover operation to solve the mixed Chinese Postman Problem and report that the GA provides near optimal solutions for problems of small size. Bráysy (2001) gives a review of the use of genetic algorithms in vehicle routing problems.

The genetic algorithms work on a population or a collection of several alternative solutions to the given problem. Each individual in the population or a possible solution is called a string or chromosome. Individual characters or symbols in the strings are referred to as ‘Genes’. Every GA solution to a problem makes use of a problem dependant chromosome structure along with the selection, crossover and mutation operators. There are two types of genetic algorithms namely, Simple Genetic Algorithm and Steady State Genetic Algorithm. In simple GA during the evolution of the new generation the newly generated individuals replace all the individuals of the previous generation, whereas in Steady State GA only the worst fit individuals are replaced by the two offspring if they have better fitness. Goldberg, (2000) states that steady state GA is better when the optimization
problem has more constraints. Hence a steady state GA is devised in this work for minimal cost matching of odd degree vertices as the constraints on the feasible solutions of matching odd degree vertices makes the use of simple GA infeasible.

The steady state GA has been used for solving many highly constrained problems of optimization. Chafekar et al, (2003) propose two novel approaches based on concurrent and sequential execution of GA’s to solve highly constrained multi-objective optimization problems. Many problem dependant structures for representing probable solutions (chromosomes) and the corresponding operators have been proposed. Ko et al, (1996) propose an adaptive cross over operator that restricts the location of the crossover position as generations progress, further it has been shown that the performance of this operator is better than the simple GA. Mesquita et al, (2002) propose a adjacency matrix representation of the chromosome for representing analog and digital circuits in electronic circuit synthesis application. Seuranen (2003) discusses the use of genetic algorithms for solving the Traveling Salesman Problem. The paper presents chromosome representation and many crossover and mutation operators.

Though various applications of GA to routing problems are found in literature, it seems that the problems such as CPP/ RPP have not received much attention from GA practitioners. A few works that solve the Chinese Postman Problem by converting it to a Hamiltonian circuit problem are found in literature, one such work is described in (Kang and Hee, 1998).

In this work a solution to RPP:DP is proposed by employing a newly devised steady state GA for minimal cost matching of odd degree vertices. The constraints of the minimal cost matching of odd degree vertices are: every odd degree vertex is to be included in one and only one pairing and the pairing of all odd degree vertices should yield the minimum cost. Incorporation of these constraints in the generation of new individuals leads to searching only the feasible space for the optimal solution, which is the characteristic of many steady state GA’s. The working of the steady state GA for finding the minimal cost matching of odd degree vertices is described in the following paragraphs.
A randomly generated population of chromosomes initiates the GA solution for minimal cost matching of odd degree vertices. The chromosome structure employed in this work is a collection of randomly matched odd degree vertices, satisfying the required constraints along with the shortest path between them. A matched pair of odd degree vertices along with the distance is referred to as a gene. Each chromosome has \( \frac{ODV}{2} \) genes, where \( ODV \) is the number of odd degree vertices in the connected graph. The structure of the chromosome is illustrated in Figure 10.1.

\[
\text{Gene} = \left\{ (V_{11}, D_{11}), (V_{12}, D_{12}), \ldots, (V_{p1}, V_{p2}, D_{p}) \right\}
\]

Where,
\[
p = \frac{ODV}{2} \quad \text{is the number of genes in a chromosome / odd degree vertex pairs}
\]
\[
V_{p1} \text{ and } V_{p2} \text{ are the pair of vertices in } p^\text{th} \text{ pairing of odd degree vertices}
\]
\[
D_p \text{ is the shortest distance between } V_{p1} \text{ and } V_{p2}
\]

**Figure 10.1: Chromosome Structure for Representing Solution of Mapping Odd Degree Vertices**

The steady state GA progresses by evolving the population making use of the roulette wheel selection (Goldberg, 2000; Mazumder and Rudnick, 1999) and newly devised crossover and mutation operators. The roulette wheel selection is implemented using a uniform random number generator and a cumulative probability, indicating the proportion of the individuals in the population based on their fitness. The fitness of the individual is the reciprocal of the cumulative distance of all the shortest paths in the chromosome/individual. The Fitness value of the individual chromosome is computed as depicted in equation 10.1.

\[
\text{FitnessOfChromosome} = \frac{1}{\sum_{i=1}^{p} D_i} \quad \ldots(10.1)
\]
where,

\[ D_i \text{ is the distance of the shortest path joining the odd degree vertices in } i^{th} \text{ gene} \]

\[ P = \frac{ODV}{2} \]

The cross over operator is devised to generate new chromosomes or alternate solutions by combining the genes of the parents selected using the selection operator. The cross over operation is performed on the selected parents with a pre assigned high probability. An empirically selected high probability of 0.9 is used in this work. The cross over operator first generates a random cross over location and uses it to generate two offspring from two parents. The first offspring is generated, by copying the genes of the second parent from cross over position to the last gene. The genes from first location to cross over position are computed by copying from the first parent if there are no overlaps. If there are overlaps and the possible genes in the first parent are exhausted then genes are taken from the second parent to complete the first offspring. The second offspring is generated, by copying the genes of the first parent from cross over position to the last gene. Further the second offspring is generated in a manner similar to the first offspring. The complete cross over operation is illustrated in Figure 10.2 for a chromosome of length 5 genes.

**Figure 10.2: The Cross-Over Operator for Matching of Odd Degree Vertices**

The steady state genetic algorithm also employs a low probability mutation operator. An empirical value of 0.3 is employed in this work. The mutation is carried out on a single
Optimization Of Mail Distribution Path

chromosome. One of the offspring is randomly selected with uniform probability for the mutation process. Further a gene of the chromosome is selected at random for affecting the mutation. A randomly chosen odd degree vertex then replaces the vertex at the selected position. If the newly introduced random vertex was previously present in this or other genes of the chromosome then to meet the single occurrence constraint of any odd degree vertex this previous occurrence has to be removed. The previous occurrence is replaced by the vertex, which was just removed from the mutation position. The mutation process also involves computing the shortest distance for the genes whose pairing has been modified.

The mutation process for the randomly selected values is illustrated in Figure 10.3.

<table>
<thead>
<tr>
<th>Parameters for Mutation</th>
<th>Mutation Individual: Offspring 1</th>
<th>Mutation Position: 3</th>
<th>Mutation feature: 1st vertex</th>
<th>Mutation feature value: V_5^2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Individual Selected for Mutation</td>
<td>(V_1^1, V_1^2, D_1^1)</td>
<td>(V_2^2, V_2^1, D_2^1)</td>
<td>(V_3^2, V_3^1, D_3^2)</td>
<td>(V_4^2, V_4^1, D_4^2)</td>
</tr>
<tr>
<td>Changes in genes due to mutation</td>
<td>(V_5^2, V_5^3, D_5^2)</td>
<td>(V_3^2, V_3^3, D_3^2)</td>
<td>(V_4^2, V_4^3, D_4^2)</td>
<td>(V_5^2, V_5^3, D_5^2)</td>
</tr>
<tr>
<td>The mutated chromosome</td>
<td>(V_1^1, V_1^2, D_1^1)</td>
<td>(V_2^2, V_2^1, D_2^1)</td>
<td>(V_3^2, V_3^1, D_3^2)</td>
<td>(V_4^2, V_4^1, D_4^2)</td>
</tr>
</tbody>
</table>

**Figure 10.3: The Mutation Operator for Matching of Odd Degree Vertices**

These crossover and mutation operators are applied repeatedly along with the selection of the parents for reproduction until the GA terminates. The GA can terminate due to one of the following conditions.

- There is no improvement in the best fitness value of the successive population
- The GA has reached the maximum number of iterations

The complete process of the steady state GA is outlined in algorithm 10.4.
Algorithm 10.4: Genetic Algorithm for finding Minimal Cost Maximal Matching of Odd Degree Vertices

**Input:** Odd degree vertices of the graph, the number of odd degree vertices (ODV) the list of edges in the original graph, the shortest path between every pair of vertices.

**Output:** The best possible match/pairing of odd degree vertices (minimal cost maximal matching, MATCH)

1. Find the number of possible pairings of odd degree vertices, \( NP = \binom{ODV}{2} \)
2. Find the size of the initial population, which is also equal to the number of chromosomes, say \( PS = 0.4 \cdot NP \) (an empirically selected value) (if \( PS > 150 \), \( PS \) is assigned the value of 150)
3. Find the number of required odd degree pairings for converting the graph to Eulerian graph, it is also the number of genes in a chromosome, \( Ng = \frac{ODV}{2} \)
4. Generate an initial population of chromosomes (probable solutions) of size \( PS \). Each chromosome has \( Ng \) number of genes.
5. Evaluate fitness of each individual and store in a Fitness Array
6. Initialize the maximum number of iterations to \( N \) (say 20), minimum number of iteration to \( n \) (say 10) and maximum fitness value to \(-1\)
7. While the terminating conditions are not met do
   a. Find the maximum fitness of the population
   b. If there is no change in maximum fitness from previous iteration then break out of while loop as there is no improvement in the solution, if minimum number of iterations are completed.
   c. If the iteration is greater than the maximum iterations then break out of while loop
   d. Select two parents (\( P1 \) & \( P2 \)) for reproduction from the population pool using roulette wheel selection. Every individual gets selected based on their fitness value
   e. Generate two offsprings (\( O1 \) & \( O2 \)) using the crossover operator
   f. Apply mutation to the offspring
   g. Find the fitness value of the offspring.
   h. Find the two least fit individuals in the current population
   i. If the fitness of the offspring is greater than the least fit individuals replace the least fit individuals with the offspring
   j. Evaluate the fitness of the new population
8. endWhile
9. Find the individual/chromosome with the highest fitness in the population. The genes of this chromosome give the solution to minimal cost maximal matching of odd degree vertices. The genes of the fittest individual give the minimal cost maximum matching, store it in MATCH
The genetic algorithm for finding the minimal cost maximum matching of odd degree vertices provides a good solution to the problem though it may not be the optimal solution. The exact solution to the problem of finding the minimum cost maximal matching of odd degree vertices can be found using the EDMONDS algorithm and is explained in section 10.3.2.

10.3.2 Exact Algorithm for Matching Odd Degree Vertices

The optimal solution to minimal cost maximal matching of odd degree vertices can be obtained by the exact solution proposed by Edmonds and Johnson (1973). The solution is applicable to directed and undirected edges without negative cost. The solution is based on an ingenious reduction of the problem to an integer linear programming problem. The process of finding the minimal cost matching involves enlisting all possible matches of odd degree vertices along with the distance of shortest path joining the pair and constructing a integer linear programming problem for inclusion of each pair in the final optimal match.

The objective of the integer linear programming problem (LPP) is to minimize the total cumulative distance / cost of all the edges, included for matching of the odd degree vertices. The solution of this problem gives the optimal match for the odd degree vertices. The construction of the integer linear programming problem for matching of odd degree vertices is explained using an example in the following paragraphs.

Consider the graph shown in Figure 10.4. All the four vertices of the graph are odd degreeed.

Figure 10.4: A Typical Graph with Odd degree Vertices
The minimum cost maximal matching problem for the graph in Figure 10.4 can be expressed as an integer LPP by first enlisting all the possible matching. The Table 10.1 lists all possible matching of odd degree vertices, the corresponding shortest path distance and the parameters employed in the LPP model.

**Table 10.1: Parameters for Matching of Odd degree Vertices**

<table>
<thead>
<tr>
<th>Sl No</th>
<th>Possible match of odd degree vertices</th>
<th>Shortest distance</th>
<th>The variable in LPP model</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(1,2)</td>
<td>50</td>
<td>X1</td>
</tr>
<tr>
<td>2</td>
<td>(1,3)</td>
<td>30</td>
<td>X2</td>
</tr>
<tr>
<td>3</td>
<td>(1,4)</td>
<td>40</td>
<td>X3</td>
</tr>
<tr>
<td>4</td>
<td>(2,3)</td>
<td>60</td>
<td>X4</td>
</tr>
<tr>
<td>5</td>
<td>(2,4)</td>
<td>55</td>
<td>X5</td>
</tr>
<tr>
<td>6</td>
<td>(3,4)</td>
<td>70</td>
<td>X6</td>
</tr>
</tbody>
</table>

The Edmonds formulation of matching odd degree vertices forms the following integer Linear Programming Problem and solves it to obtain the minimal cost maximal matching by relaxing integer requirement (Edmonds and Johnson, 1973).

Minimize $50X_1 + 30X_2 + 40X_3 + 60X_4 + 55X_5 + 70X_6$ such that the following conditions are satisfied,

\[
\begin{align*}
X_1 + X_2 + X_3 & \leq 1 \\
X_1 + X_4 + X_5 & \leq 1 \\
X_2 + X_4 + X_6 & \leq 1
\end{align*}
\]

This condition implies that there can be only one edge incident on vertex 1.

This condition implies that there can be only one edge incident on vertex 2.

This condition implies that there can be only one edge incident on vertex 3.
The solution to this integer LPP by relaxing the integer requirement and later converting real values to integers, gives the exact solution to the minimal cost maximal matching of odd degree vertices. This algorithm has been shown to have a time complexity of $O(n^3)$ where $n$ is the number of odd degree vertices (Edmonds and Johnson, 1973).

### 10.4 OPTIMAL PATH FOR DISTRIBUTION OF POSTAL MAIL AND DISTRIBUTION SCHEDULE

The Euler graph generated by the use of algorithm 10.3 is processed further using the Fleury's algorithm described in chapter 9 to find the closed walk / Euler trail that starts and ends on the node representing the Mail Distribution Post Office. The Euler trail selects the remaining edge that has the least cost to go out of any vertex. The Euler trail is further processed using the computational strategy described in algorithm 10.5 for generating the distribution schedule. The distribution schedule is generated based on the premise that the mail is delivered to the delivery point when the postman passes through the corresponding road for the first time. Also when the postman first reaches a vertex from where a partial edge is to be served, the postman immediately completes the task.
**Algorithm 10.5**: Generation of the distribution schedule for the delivery of mail

**Input**: List of Mail Delivery Points, Edge list of the road network where the mail is to be distributed, Mail Distribution Graph(eMDG,peMDG), Euler Trail (A closed walk), Number of edges in euler trail say ETN, number of MDPs (MDN), GIS data for mail delivery points, MDPO node

**Output**: Sequence of distribution of mail along with time of delivery

1. Rearrange the mail delivery points in non-decreasing order of the edge.ID. The mail delivery points on one road are collected at one location
2. Sort the mail delivery points lying on each edge/road in the non-decreasing order of the distance of the MDP from end point 1 of the road
3. Initialize edgeindex to point to first edge in euler trail and vertex to the MDPO node
4. While there are more edges in euler trail do
   a. Find the initial vertex
   b. Check whether there is a partial edge to be traversed from this vertex, if so generate the distribution schedule for the MDPs in the edge to be partially traversed from peMDG and store the schedule in DIST_SCHED, Mark the delivery points to which distribution schedule has been generated
   c. Find the first/next edge in Euler Trail
   d. Find if there are mail to be delivered on the edge/road
   e. If there are mails to be delivered then
      i. Find the direction of traversal of the edge and assign the distribution schedule for the delivery points on the road according to traversal direction and store it in DIST_SCHED. Also store the distribution time for each
      ii. Mark the delivery points to which distribution schedule has been generated
   f. EndIf
   g. If the delivery schedule of all the mail delivery points is generated break out of the while loop
5. EndWhile

6. DIST_SCHED consists of the distribution schedule for delivery of the mail on the given day

The mail distribution schedule along with the printed map of the beat where the mail is to be distributed can be provided to the postman for delivery of mail. This helps the postman/delivery agent in efficient delivery of mail and also enhances the image of postal service. The application of the computational strategies to an actual road network representing the postman beat is illustrated in section 10.5.
10.5 OPTIMAL PATH FOR DISTRIBUTION OF POSTAL MAIL: A CASE STUDY

The computational strategies presented in the previous sections can be employed to generate an optimal route for distribution of mail by the postman. The computational strategies described in this chapter also generate a mail distribution schedule. This section presents a case study for generation of optimal route and distribution schedule for delivery of mail. A beat of the postman belonging to Market Branch Post Office of Bagalkot, a district place in the state of Karnataka of India is considered for generation of optimal path. The road network of the said postman beat is given in Figure 10.5.

![Figure 10.5: Road Network Representing the beat of a postman (G1)](image)

The mail delivery points and the other required entities are stored in GIS representation of the postman beat, in the format described in chapter 9. The mail delivery points to which the mail is to be distributed on a typical day is listed in Figure 10.6. The list consists of 38 unique delivery points to be visited.

![Figure 10.6: The list of Mail Delivery Point (IDs) to which mail is to be delivered](image)
The application of algorithm 10.1 for processing the list of mail delivery points generates the mail distribution graph (MDG) depicted in Figure 10.7.

The mail distribution graph (MDG) depicts the edges and vertices that are to be visited to distribute the mail. The filled vertices are to be compulsorily visited. The unfilled vertices or road end points need not be visited for distribution of the given days mail. The self loops in the graph such as e1', e2' are the edges/roads which are to be partially traversed from the indicated vertex or road end point.

To visit the edges and vertices of MDG from MDPO and return back to MDPO the road network should be a connected graph. Hence to generate the optimal route the MDG should be converted to a connected graph. The MDG is converted to a connected graph by employing the algorithm 10.2. Some of the edges, which were required to be partially traversed, are now converted to complete traversal as these edges are necessarily traversed to have a connected graph. The connected graph generated for the MDG in Figure 10.7 and delivery points of Figure 10.6 is illustrated in Figure 10.8. The self loops (Deo, 1974) again indicate the edges requiring partial traversal.
The Connected Graph shows the edges of the connected component that are to be traversed to distribute mail starting from MDPO. The edges, which need partial traversal, are also depicted. The closed walk that starts and ends on MDPO can be generated if and only if the graph is Eulerian. The graph is Eulerian if and only if every node of the connected graph is of even degree. In most of the cases as in the present case the connected graph is not Eulerian. The non-Eulerian connected graph is then converted to Euler graph by appropriate matching of odd degree vertices (algorithm 10.3). The odd degree vertices of the connected graph are matched such that the sum of distance/cost of the additional edges added (which may also include retracing of some of the edges) is the minimum. Two methodologies are proposed in this work for matching of odd degree vertices. One is matching using the Genetic Algorithm and the other is matching using the EDMONDS algorithm as described in sections 10.3.1 and 10.3.2. In this case both the algorithms yield the same matching. It is to be observed that the EDMONDS algorithm always yields the optimal matching whereas the genetic algorithm yields a good solution that can be obtained quickly. The matching obtained for the connected graph of Figure 10.8 is listed in Table 10.2 for Genetic Algorithm and EDMONDS algorithm. The edges are added to generate an Euler graph. If these added edges correspond to any of the edge needing partial traversal, then such edge is removed from partial traversal list. The Euler graph generated by applying algorithm 10.3 is depicted in Figure 10.9.
### Table 10.2: Matching of Odd Degree Vertices

<table>
<thead>
<tr>
<th>SI No</th>
<th>Odd Degree Vertices</th>
<th>Genetic Algorithm</th>
<th>Edmonds Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Optimal Matching</td>
<td>Edges to be added</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2,5) (7,14)</td>
<td>7,5,17,14,11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(7,14)(2,5)</td>
</tr>
<tr>
<td>1</td>
<td>{2,5,7,14}</td>
<td>7,5,17,14,11</td>
<td>17,14,11,7,5</td>
</tr>
</tbody>
</table>

![Figure 10.9: Euler Graph for Connected Graph in Figure 10.8](image)

The Euler graph in Figure 10.9 is used to generate the distribution path and distribution schedule for the mail on the given day. The Euler graph is processed using the Fleury’s algorithm to generate a closed walk starting and ending at MDPO. This walk is referred to as the Euler trail. The Euler trail is further processed along with the list of mail delivery points that receive the mail and the GIS data to obtain a distribution schedule. The application of Fleury’s algorithm and the computational strategy described in algorithm 10.5 results in the optimal traversal path and distribution schedule given in Figure 10.10.
Optimization Of Mail Distribution Path

<table>
<thead>
<tr>
<th>Traversal and Distribution Schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Postman Path</strong> (Sequence of vertices visited): 1-3-8-12-15-10-11-7-11-10-14-9-5-6-2-1</td>
</tr>
<tr>
<td><strong>Mail Distribution Schedule</strong> (Mail Delivery Point (Delivery Time in seconds))</td>
</tr>
<tr>
<td>42(15)-43(20)-44(20)-45(20)-46(20)-176(15)-184(15)-188(20)-307(20)-302(20)-296(20)-</td>
</tr>
<tr>
<td>253(20)-252(15)-251(20)-250(15)-249(20)-222(15)-223(20)-224(15)-225(20)-226(20)-117(20)-</td>
</tr>
<tr>
<td>118(20)-119(20)-120(20)-121(20)-287(20)-282(20)-278(15)-93(20)-94(20)-95(20)-96(20)-24(15)-</td>
</tr>
<tr>
<td>-23(15)-22(15)-21(15)-20(20)</td>
</tr>
<tr>
<td><strong>Total distance traversed by postman</strong>: 7260 mts</td>
</tr>
<tr>
<td><strong>Transit time</strong> (@ average transit rate of 10 mts/sec): 726 secs</td>
</tr>
<tr>
<td><strong>Time for crossing Junction Points</strong>: 125 secs (Cross over time of each junction point available in GIS data)</td>
</tr>
<tr>
<td><strong>Time for delivery of Mail</strong>: 700 secs (Delivery time of each MDP available in GIS data)</td>
</tr>
<tr>
<td><strong>Total Time for traversal and delivery</strong>: 1551 secs = 25.85 minutes</td>
</tr>
</tbody>
</table>

**Figure 10.10**: Optimal Traversal Path and Distribution Schedule

The case study presented in this section has illustrated the use of the intelligent computational strategies for generation of optimal route and mail distribution schedule for a set of delivery points to which one or more mail is to be delivered. The computational strategies are tested with various postman beat networks and lists of delivery points. The results are discussed in section 10.6.

10.6 EXPERIMENTAL ANALYSIS

The computational strategies are implemented on a computer with an INTEL CELERON processor @1.3 GHz and 512 MB memory using MATLAB 7.0 software. The intelligent strategies for generation of optimal path for traversal of the postman and the mail distribution schedule are tested on three different postman beats, which are represented by the graphs illustrated in Figures 10.5, 10.11 and 10.12. The graph of Figure 10.5, namely G1 represents the beat of a postman of Market Branch Post Office of Bagalkot. The graphs of Figure 10.11(G2) and Figure 10.12 (G3) represent the beats of postmen belonging to GRBC post office, of Bagalkot. The graphs represent the road network of the postman beats including cross roads. The graphs do not represent the roads to any scale but only show the interconnection of different roads. The direction, size and other attributes of the roads are stored in the GIS representation of the beat of the postman. The mail delivery points and other entities of interest in the generation of optimal path are stored in the GIS.
representation. Each of the postman beat is tested with five different lists of mail delivery points to which mail is to be delivered. The lists of mail delivery points are selected to illustrate the different mail distribution loads. The results are tabulated in Tables 10.3 and 10.4. Table 10.3 enlists the results of optimal path and distribution schedule generated using genetic algorithm for finding the minimal cost matching of odd degree vertices. Whereas Table 10.4 lists the results when EDMONDS algorithm is used for minimal cost matching of odd degree vertices.

Figure 10.11: Road Network Representing the Beat of Postman (G2)
Figure 10.12: Road Network Representing the Beat of Postman (G3)

Table 10.3: Results of Generation of Traversal Path for the Postman and Distribution Schedule using Genetic Algorithm for Minimal Cost Matching of ODD Degree Vertices

<table>
<thead>
<tr>
<th>Si No</th>
<th>Graph Representing Beat of Postman</th>
<th>Number of Delivery Points</th>
<th>Mail Distribution Graph (MDG)</th>
<th>Connected Graph (CG)</th>
<th>Euler Graph (EG)</th>
<th>Total distance for traversal of Mail (meters)</th>
<th>Total time required for Distribution of Mail (seconds)</th>
<th>Minimum Distances (meters)</th>
<th>Maximum Distances (meters)</th>
<th>Average computation Time in seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>G1</td>
<td>15</td>
<td>20</td>
<td>15</td>
<td>15</td>
<td>7260</td>
<td>1551</td>
<td>7260</td>
<td>7320</td>
<td>1.65</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2059</td>
<td>8840</td>
<td>2610</td>
<td>2660</td>
<td>6.04</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9550</td>
<td>13700</td>
<td>9800</td>
<td>10000</td>
<td>7.00</td>
</tr>
<tr>
<td>2</td>
<td>G2</td>
<td>32</td>
<td>47</td>
<td>22</td>
<td>22</td>
<td>14700</td>
<td>2610</td>
<td>13700</td>
<td>15100</td>
<td>4.94</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3255</td>
<td>16000</td>
<td>16000</td>
<td>17900</td>
<td>6.04</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16800</td>
<td>19400</td>
<td>19400</td>
<td>22600</td>
<td>4.41</td>
</tr>
<tr>
<td>3</td>
<td>G3</td>
<td>45</td>
<td>62</td>
<td>36</td>
<td>36</td>
<td>18500</td>
<td>2815</td>
<td>18500</td>
<td>20900</td>
<td>7.84</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>28300</td>
<td>28300</td>
<td>28300</td>
<td>32200</td>
<td>9.25</td>
</tr>
</tbody>
</table>

Note: All distances are in meters
### Table 10.4: Results of Generation of Traversal Path for the Postman and Distribution Schedule using EDMONDS Algorithm for Minimal Cost Matching of ODD Degree Vertices

<table>
<thead>
<tr>
<th>Sl No</th>
<th>Graph Representing Best of Postman</th>
<th>Number of Delivery Points</th>
<th>Mail Distribution Graph (MDG)</th>
<th>Connected Graph (CG)</th>
<th>Euler Graph (EG)</th>
<th>Distance for Traversal of Mail</th>
<th>Total Required for Distribution of Mail</th>
<th>Average computation Time in seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>G1 [V]: 15 [E]: 20</td>
<td></td>
<td>38</td>
<td></td>
<td></td>
<td>4</td>
<td>17</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>60</td>
<td></td>
<td></td>
<td>9</td>
<td>18</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>103</td>
<td></td>
<td></td>
<td>10</td>
<td>19</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>157</td>
<td></td>
<td></td>
<td>15</td>
<td>23</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>190</td>
<td></td>
<td></td>
<td>18</td>
<td>23</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>G2 [V]: 32 [E]: 47</td>
<td></td>
<td>40</td>
<td></td>
<td></td>
<td>11</td>
<td>35</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>63</td>
<td></td>
<td></td>
<td>15</td>
<td>39</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>104</td>
<td></td>
<td></td>
<td>21</td>
<td>42</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>151</td>
<td></td>
<td></td>
<td>26</td>
<td>44</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>192</td>
<td></td>
<td></td>
<td>34</td>
<td>49</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>G3 [V]: 45 [E]: 62</td>
<td></td>
<td>40</td>
<td></td>
<td></td>
<td>10</td>
<td>36</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>60</td>
<td></td>
<td></td>
<td>13</td>
<td>50</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>105</td>
<td></td>
<td></td>
<td>26</td>
<td>55</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>150</td>
<td></td>
<td></td>
<td>39</td>
<td>62</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>204</td>
<td></td>
<td></td>
<td>45</td>
<td>67</td>
<td>8</td>
</tr>
</tbody>
</table>

Note: All distances are in meters

The results in Tables 10.3 and 10.4 bring out several interesting aspects about the optimal path for distribution of mail and the distribution schedule. Some of the facts brought out by the results are discussed below.

- The computational strategies to generate optimal path for distribution of mail use either the exact EDMONDS algorithm or the heuristic Genetic Algorithm for minimal cost matching of odd degree vertices. The EDMONDS algorithm always finds the minimal cost, but the first odd degree vertex pairing, which achieves the minimum value, even if there are more than one such pairings, is always selected and used for matching of odd degree vertices and further the generation of Euler graph. Hence the cost of the optimal route for distribution of mail using EDMONDS algorithm is always constant.
• The Genetic Algorithm being a heuristic or approximate search strategy will always yield a ‘good’ solution that is near to the optimal or equal to the optimal when finding the minimum cost matching of odd degree vertices. The GA sometimes yields the optimal solution but may pick alternate pairings than the one selected by EDMONDS methodology hence this may lead to a better overall optimal path in some cases. The total distance of the optimal path generated for distribution of mail to 190 MDP’s in graph G1 brings out this aspect (Table 10.3 and Table 10.4). The computational strategy using EDMONDS algorithm has yielded a total traversal distance of 9800 meters whereas a particular iteration of GA has given a total traversal distance of 9600 meters, which is less than the distance of the optimal path obtained by the strategy using EDMONDS matching algorithm.

• The minimum and maximum distances of the optimal route obtained by testing the methodology for fifty iterations is listed in the Table 10.3. The maximum or the worst-case distance obtained using GA is less than 1.25 times the corresponding optimal path obtained using EDMONDS strategy. A relative plot of the optimal distance computed using EDMONDS strategy and worst-case distance computed using Genetic Algorithm is illustrated in Figure 10.13, 10.14 and 10.15 for graphs G1, G2 and G3 respectively.

![Optimal Path Distances](image)

**Figure 10.13:** Optimal Distances using EDMONDS strategy and GA for the Beat of the Postman Represented by Graph G1
The Genetic Algorithm computes the optimal path in less time as compared to the EDMONDS methodology. Further the difference in computation time is more pronounced for graphs representing road networks that have more nodes and edges. The time profile of the two methodologies for each of the beats is depicted in Figures 10.16, 10.17 and 10.18.
Figure 10.16: Time Profile of Computational Strategies to Find Optimal Path for Distribution of Postal Mail for Graph G1

Figure 10.17: Time Profile of Computational Strategies to Find Optimal Path for Distribution of Postal Mail for Graph G2
The observation of the time profile of computational strategies indicates that the computational time is less for methodology based on GA. Also the computational time does not monotonically increase with the number of mail delivery points. The computation time is dependant on the geographical spread of the delivery points to which mail is to be distributed.

The geographical spread of the delivery points has an impact on the structure of the mail distribution graph (MDG). The transformation of MDG to a connected graph is a critical step in the process of generation of the optimal path for distribution of mail and occupies a major part of the computation. The newly devised algorithm based on graph transformation using the spatial data structure (SDS-PB), proposed in this work generates the connected graph for the beat of the postman. The connected graph generation not only depends on the number of vertices and edges but also on the geographical spread of the mail delivery points. The Figure 10.18 brings out this aspect. The computational time for generation of optimal path for 40 and 60 mail delivery points is greater than the time required for generation of optimal paths for 105,150 and 204 mail delivery points. This is because the 40 and 60 delivery points generate a mail distribution graph which is
spatially spread out requiring the usage of the second stage for generation of the connected graph.

- Another aspect that stands out after observing the results is that the distance of the optimal path will not increase in direct proportion to the number of mail delivery points and again it is dependant on the geographical spread of the mail delivery points. This fact is illustrated by the optimal distance for distribution of mail to 103 mail delivery points (7840 meters) and 60 mail delivery points (7890 meters) in graph G1 as depicted in Table 10.4.

These features of the optimal route for distribution of postal mail indicate that use of computational tools is a necessity and results in proper utilization of resources. The postman can make use of the motored or un-motored vehicles for distribution of postal mail. The computational strategies described have optimized the traversal distance. The strategies can be adopted to optimize other measures such as the product of mail weight and distance.

10.7 CONCLUSION

This chapter has presented intelligent computational strategies based on graph theoretic and genetic algorithm approaches for generation of optimal path for distribution of postal mail and the mail distribution schedule. The methodology outlined here converts the problem of finding the optimal route for visiting a list of mail delivery points to a problem of finding the Euler trail in a graph by first generating a mail distribution graph (MDG) and then a connected Graph. A closed walk is generated in this connected graph by employing appropriate graph transformation strategies. It is shown that GA based strategy is computationally efficient as compared to the alternative based on EDMONDS strategy for matching of odd degree vertices but may sometimes generate sub optimal solutions. Hence the choice GA v/s EDMONDS strategy for matching of odd degree vertices is to be made based on design constraints in the implementation phase.

The computational strategies are to be implemented at a distribution post office/ sorting office, which has the GIS data for each postman beat. The list of MDP'S to which mail is
to be distributed on a given day is generated using the strategies discussed in part II. The list can then be used for generation of optimal route and also the distribution schedule using the strategies outlined in this chapter. The techniques presented here can be adapted to various other applications such as robot motion planning, wave soldering in electronic component manufacture, generation of optimal route for distribution of milk, generation of optimal route for distribution of newspaper and other utility problems.

The optimal mail distribution route generated using the computational strategies presented in this chapter assumed that the postman moves from one mail delivery point (MDP) to another, while delivering the mail. In practical situations if there is a cluster of nearby points to which mail is to be delivered then the postman parks his vehicle at a central place, distributes the mail in the cluster and then proceeds to the next delivery point or the cluster. Chapter 11 presents computational strategies for finding practically best route for distribution of mail in such a scenario.