DISTRIBUTED MOTION PLANNING ALGORITHMS FOR A COLLECTION OF VEHICLES

A Thesis

by

SUDHIR PARGAONKAR

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2003

Major Subject: Mechanical Engineering

DISTRIBUTED MOTION PLANNING ALGORITHMS FOR

A COLLECTION OF VEHICLES

A Thesis

by

SUDHIR PARGAONKAR

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Approved as to style and content by:

Darbha Swaroop (Chair of Committee)	Craig Smith (member)
A.L.N. Reddy (member)	Dennis L. O'Neal (Head of Department)

December 2003

Major Subject: Mechanical Engineering

ABSTRACT

Distributed Motion Planning Algorithms for
a Collection of Vehicles. (December 2003)
Sudhir Pargaonkar, B.E., Karnataka Regional Engineering College, Surathkal, India
Chair of Advisory Committee: Dr. Darbha Swaroop

Unmanned Vehicles (UVs) currently perform a variety of tasks critical to a military mission. In future, they are envisioned to have the ability to accomplish a mission co-operatively and effectively with limited fuel onboard. In particular, they must search for targets, classify the potential targets detected, attack the classified targets and perform an assessment of the damage done to the targets. In some cases, UVs are themselves munitions. The targets considered in this thesis are stationary. The problem considered in this thesis, referred to as the UV problem, is the allotment of tasks to each UV along with the sequence in which they must be performed so that a maximum number of tasks are accomplished collectively.

The maneuverability constraints on the UV are accounted for by treating them as Dubin's vehicles. Since the UVs considered are disposable with life spans governed by their fuel capacity, it is imperative to use their life as efficiently as possible. Thus, we need to develop a fuel-optimal (equivalently, distance optimal) motion plan for the collection of UVs.

As the number of tasks to be performed and the number of vehicles performing these tasks grow, the number of ways in which the set of tasks can be distributed among the UVs increases combinatorially. The tasks a UV is required to perform are also subject to timing constraints. A UV cannot perform certain tasks before completing others.

We consider a simplified version of the UV problem and do not take into account the timing constraints on the tasks to be performed on targets.

We use linear programming and graph theory to find a solution to this simplified UV problem; in the graph theory approach, we develop an algorithm which is a generalization of the solution procedures available to solve the Traveling Salesman Problem (TSP). We provide an example UV problem illustrating the solution procedure developed in this thesis.

ACKNOWLEDGMENTS

I thank Dr. Swaroop for being the Chair of my advisory committee and helping me throughout my thesis with his valuable advise. I also thank Dr. Craig Smith and Dr. A.L.N. Reddy for their support and giving their time to be on the advisory committee.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION AND PROBLEM STATEMENT

Advances in material, sensing, actuation, information processing and communication technologies are enabling the development of a low-cost, flexible, unmanned miniaturized munitions that can perform a variety of tasks critical to a military mission.

In an uncertain, adversarial environment, these munitions, henceforth referred to as Unmanned Vehicles (UVs) or even simply vehicles, must search for targets, classify the potential targets detected, attack the classified targets and perform an assessment of the damage done to the targets. The classification of potential targets detected is important in ensuring that collateral damage is minimal. The assessment of damage is necessary to ensure that a target is sufficiently destroyed.

The UVs considered for such applications are about 3 feet wide and 4 feet long; they are expected to last for a few hours (30 min to 2 hours). They carry the sensors to detect targets, devices to communicate with other UVs and perhaps with ground controllers, and the requisite hardware to perform on-board navigation and decision making. However, the UVs are constrained by yaw rate constraints. Furthermore, they are use-and-throw weapons and are destroyed as soon as they attack a target or are out of fuel.

The targets considered in this research are stationary. Targets could be valued differently. We will refer to a target as having a higher value to mean that the damage it can inflict is higher; hence, it is a higher priority task to search for, classify and destroy as many higher valued targets of the enemy as possible and in the shortest possible time.

The journal model is *IEEE Transactions on Automatic Control*.

Since the UVs considered in this thesis are of use-and-throw variety and their life spans are limited, it is imperative that they utilize their life as efficiently as possible. It is for this reason that attacking a low valued target can be delayed until a UV is running out of fuel and the classified low valued target is within its reach.

The problem considered in this thesis, referred to as the UV problem, is the allotment of tasks to each UV along with the sequence in which they must be performed so that a maximum number of tasks are accomplished collectively.

A. Static 1-1 Assignment

One way to allocate resources is to perform a static 1-1 assignment of resources to tasks at every instant a potential target is detected or at every instant a task is accomplished. The static 1-1 assignment is myopic as it does not take into account the future tasks that must be performed on a target. For example, consider two targets that are close to one another, but are farther away from a set of UVs; a 1-1 assignment would require two UVs to travel to these targets when a significant amount of fuel could be saved if only one UV serves both targets. Thus, with a static 1-1 assignment, a significant inefficiency in the utilization of fuel occurs.

Reference [11] describes a method of teaming UVs to perform different tasks as a group and to overcome the drawback of a static 1-1 assignment. This paper suggests that if worked in teams, UVs can utilize their life more efficiently.

B. Challenges Posed by the UV Problem

We encounter many problems in solving the UV problem; some of them are:

1 Combinatorial Complexity: As the number of tasks to be performed and the number of vehicles performing these tasks grow, the number of ways in which

the set of tasks can be distributed amongst the UVs increases rapidly. A further difficulty arises in ensuring that such a distribution satisfies coordination constraints, i.e., classification of a potential target is performed earlier than attack and so on.

2 Coupling with Motion Planning: Since the primary goal of resource allocation is to utilize the life of UVs efficiently, the time taken by UVs to arrive at the targets will play a significant role. The constraints on maneuverability, such as the minimum turning radius for UVs at operational speeds, pose further challenges:

a) Motion Planning: Given a set of tasks and the order in which they are to be performed by a UV on presumably different targets, the problem of generating a time-optimal motion plan is not trivial. (This is the case even with the simplifying assumption that the UVs travel at a constant speed and that the curvature of their trajectories can be changed instantaneously.)

b) Combinatorial complexity: Suppose two UVs arrive at a target, G1, with different headings. The time it takes for them to arrive at another target, G2, starting from G1 will be different. This is the source of the combinatorial complexity arising from the coupling between distribution of tasks and motion planning.

For example, if we specify the set of tasks that must be performed by a UV but not the order in which the tasks are to be performed, the determination of the minimum time motion plan to complete all the tasks in the set by the UV is a difficult problem due to constraints on its maneuverability.

C. Organization of the Thesis

In the next chapter we discuss similar problem models, such as, the Traveling Salesman Problem (TSP) and its variations. We also mention some of the existing algorithms to solve the TSP. In chapter III, we categorize the given UV problem into four different classes and propose solutions to the first two classes of the UV problem. We also provide an example UV problem illustrating the proposed solution procedure.

D. Novelty of the Algorithm

We use linear programming and graph theory approach to find a solution to the simplified UV problem. The proposed algorithm takes advantage of the fact that, in the UV problem, triangular inequality may not always hold true. The algorithm also accounts for the fuel constraints on the UVs. The proposed algorithm is a generalization of the solution procedures available to solve the TSP.

CHAPTER II

SIMILAR PROBLEM MODELS

There exist different problems such as the Traveling Salesman Problem (TSP), which are similar the UV problem. However, the UV problem has certain unique characteristics. These problems are defined on a graph where we are given a set of V vertices and a link joining vertex v_i to vertex v_j. A link is identified as an ordered pair of vertices (v_i, v_j) and the weight associated with each link is equal to the distance $d(v_i, v_j)$ between v_i and v_j. A graph is called "symmetric" if the distance between (v_i, v_j) is same as the distance between (v_j, v_i). An open path is a sequence of links $\{(v_0, v_1), (v_1, v_2), ..., (v_{p-1}, v_p)\}$ and can be identified uniquely with the sequence of cities visited, namely $\{v_0, v_1, ..., v_p\}$. In a path, every vertex is visited only once. In case of a closed path or a circuit, $v_0 = v_p$ such that we return to the starting point completing a cycle.

A. A Related Problem - Traveling Salesman Problem (TSP)

As defined in [4], in the TSP, we are given a set of N cities and a distance $d(c_i, c_j)$ for each pair of cities c_i, c_j. The goal is to find a permutation π of the cities that minimizes $\sum_{i=1}^{N-1} d(c_{\pi(i)}, c_{\pi(j)}) + d(c_{\pi(N)}, c_{\pi(1)})$.

The TSP can also be defined in terms of graphs and Hamilton Circuits. Each city is considered as a vertex of a graph. The link joining vertices V_i, V_j has weight equal to distance between the cities c_i, c_j. A Hamilton circuit can be defined as: A path that starts at a vertex of a graph, passes through every vertex exactly once, and returns to the starting vertex. The TSP is to find an optimal Hamilton circuit where each city is covered exactly once. [2]

There are different kinds of algorithms proposed to find an approximate solution

to a TSP. Reference [10] has comprehensive listing of papers, source code, preprints and technical reports about the TSP and its variants. Reference [2] explains Hamilton Circuits and algorithms used to solve the TSP.

The unique characteristics of the UV problem are:

- The triangle inequality may not hold true for the weights of the links in the UV problem. For example, if A, B and C are three targets, then, as shown in figure 1, it is possible that $d(A, B) + d(B, C) < d(C, A)$. Generally for a UV, the cost of travel from A to C is different from that of C to A.

- In the TSP, there are no constraints such as the fuel constraint. Thus there is no upper limit on the distance to be traveled.

- Unlike the TSP, UVs are not required to return unless they have enough fuel and none of the tasks they are assigned is terminal (attack).

- The optimal solution of the TSP may start with any city. As the positions of a UVs are fixed, a UV can not start from any random vertex.

- The UV problem may be described on an incomplete graph unlike the TSP. In case of an incomplete graph, one cannot guarantee that all vertices will be visited in the graph.

B. Variations of the TSP

Orienteering Problem: This problem is defined only on symmetric graphs. We are given the starting vertex (which is referred to as a depot). Each vertex in the graph has a prize value $f(v)$ assigned to it and can be visited at MOST once. The goal is to find a open path $\pi = (v_0, v_1, ..., v_p)$ so as to maximize $\sum_{i=0}^{p} f(v_i)$ subject to

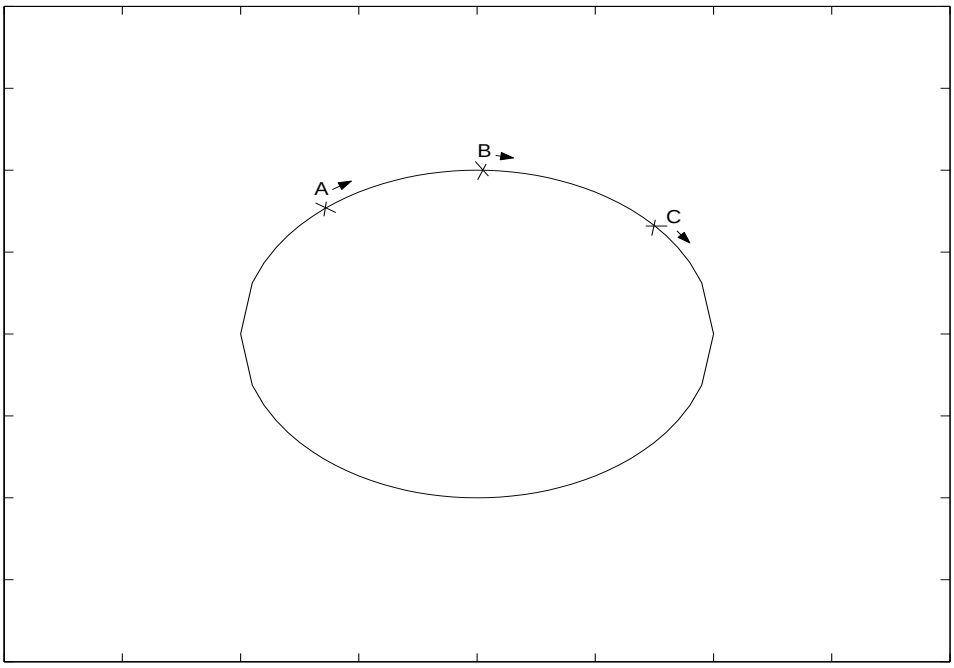

Fig. 1. Violation of triangular inequality

the constraint $\sum_{i=0}^{p-1} d(v_i, v_{i+1}) < D$ where D is the upper bound on distance traveled. Heuristic algorithms to solve the Orienteering Problem are given in [3] and [12].

Prize Collecting Traveling Salesman Problem: Each vertex in the graph has a prize value $f(v)$ assigned to it. Each vertex can be visited at MOST once. The goal is to find a path π and to minimize $\sum_{i=0}^{p} d(v_i, v_{i+1})$ subject to the constraint $\sum_{i=0}^{p} f(v_i) > B$ where B is the lower bound on the total prize to be collected.

Vehicle Route or Vehicle Dispatch Problem (VRP): This problem involves multiple vehicles. Each vertex has some demand and service time associated with it. Every vertex can be visited exactly once by only one vehicle. All vehicles start at the given vertex (depot) and come back to the same vertex. The goal is to find the minimum cost vehicle path such that total demand of every vehicle route does not exceed the vehicle capacity and the total route time inclusive of service time, does

not exceed a given upper bound. Reference [6] suggests a heuristic algorithm for the Vehicle Dispatch Problem.

If the UV problem is symmetric (the cost of travel from target A to target B is equal to the cost of travel from target B to target A), then it can be posed as an Orienteering Problem. We know the initial position of the UV, it can be considered as the starting vertex(depot). The maximum distance that can be traveled with the given fuel can be considered as the upper bound. We can assign a priority index to each vertex that would work as the prize of that vertex. If all vertices have same priority then the problem becomes the TSP where we minimize traveled distance. In general, the UV problem is not symmetric.

C. Algorithms to Solve the TSP

Reference [1] explains cutting-plane method suggested by Dantzig, Fulkerson and Johnson to solve the TSP. Approaches to solve an asymmetric TSP are given in [8] and [4]. Reference [2] suggests following "approximate" solutions. Links of the graph are also referred to as "edges".

1. Cheapest Link Algorithm (CLA)

- Choose the edge with the smallest weight (the "cheapest" edge), randomly breaking ties
- Keep choosing the "cheapest" edge unless it (a) closes a smaller circuit OR (b) results in 3 selected edges coming out of a single vertex
- Continue until the Hamilton Circuit is complete

This algorithm is exactly same as the one used in [11]. In this scheme, we sort edges of the graph in the increasing order of their weights. Then, we choose different edges

from this sorting based on the second condition explained above. If we have same weight for multiple links, then we randomly order those links. One of the drawbacks of this algorithm is that we get different results and different approximate total costs for a different ordering of links with the same weight.

2. Nearest Neighbor Algorithm (NNA)

- Start at a vertex
- Travel to the vertex that has not been visited along the link that has the smallest weight. (If there is a tie, break it randomly.)
- Continue until all vertices are covered
- Return to the starting vertex

As the starting vertex is randomly chosen, we may not obtain an optimal solution.

3. Repetitive Nearest Neighbor Algorithm (RNNA)

- Perform NNA repeatedly with different starting vertices and consider all possibilities
- Choose the best solution (smallest weight)
- If necessary, rewrite this solution with a particular starting vertex

4. The Brute Force Algorithm

- List all possible Hamilton circuits
- Find the weight of each circuit
- Choose the one with the smallest weight

This method guarantees an optimal solution to the TSP but it is computationally burdensome and can only be used when dealing with graphs with fewer vertices and edges.

We develop an algorithm to solve the UV problem, adapting the existing methods to solve TSP.

CHAPTER III

DEVELOPMENT OF A RESOURCE ALLOCATION ALGORITHM BASED ON MOTION PLANNING

In the UV problem we assign targets to each UV and find an efficient route to visit these targets, maximizing the total number of tasks performed by the collection of UVs. We categorize the given UV problem into four different classes and solve a simplified form of the UV problem in this thesis.

A. Simplified Problem Statement

The posed UV problem can be considered as an optimization problem where we have to find the minimum cost path traveled by each UV, covering the maximum number of targets. The optimization is subject to constraints such as fuel capacity of each UV, angle at which each target must be approached for proper classification (angle of approach) and the minimum turning radius of a UV. The problem can be classified into the following four different types; based on the knowledge of the sequence of targets to be visited and the angle of approach to each target.

- Angle of approach to each target is known and sequence of targets is also known. (Type I)
- Angle of approach to each target is known and sequence of targets is not known. (Type II)
- Angle of approach to each target is not known and sequence of targets is known. (Type III)
- Angle of approach to each target is not known and sequence of targets is not

known. (Type IV)

A problem of the first type is the easiest to solve. In this type of problem, we already know the angle at which each target must be approached and also the sequence of targets the UV should follow. Here, the assumption made is that if a target can be approached at an angle θ, then it can also be approached at the angle $\theta + \pi/2$. Thus we have four possible paths between every two targets in the sequence. The problem is to choose one of these four paths between two targets optimizing the total distance traveled. This problem is solved using linear programming as explained later.

In case of the problem of the second type, we do not know the sequence of the targets but we know the angle of approach. Thus, we know the distance to be traveled to go from one target to other. We have developed an algorithm, as described later in this thesis, to solve this type of a problem.

In the problem of the third type, we do not know the angle of approach but we know the sequence of targets. Here the optimization parameter is the angle of attack at each target. A solution to this problem is proposed in [9]. To find a solution to this type of a problem, we can also use the linear programming technique, used to solve a problem of type I, with few modifications.

In case of the problem of the fourth type, we do not know angle of approach as well as the sequence of the targets to be visited. No solution has been developed for this type of problem.

A UV can face the problem of type I or type II during its classification mode. In the classification mode, UVs classify potential targets into high or low value targets based on their capability to inflict the damage. To classify a target, UV approaches the target from two different angles. Generally these angles differ by $\pi/2$ so it is sufficient to define one angle of approach.

Problems of type III or type IV can occur while performing the assessment of the damage done to the targets. UVs can visit a target at any angle to assess the damage.

B. Solution to the Problem of Type I

Let A-B-C be the sequence of targets. Figure 2 shows the angles at which a UAV can approach these points.

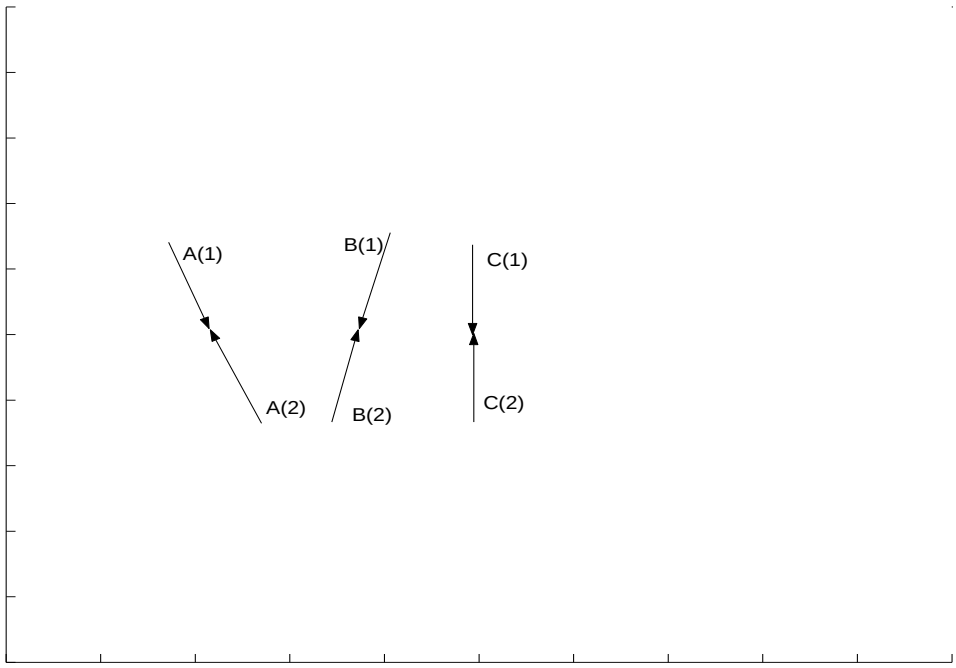

Fig. 2. Example of the problem of type I

C_{11} denotes the cost to travel from A1 to B1 and C_{12} denotes the cost to travel from A1 to B2 and so on.

Table I gives the cost matrix to travel from A to B. To generalize the problem, con-

Table I. Cost matrix

A/B	1	2
1	C_{11}	C_{12}
2	C_{21}	C_{22}

sider figure 3.

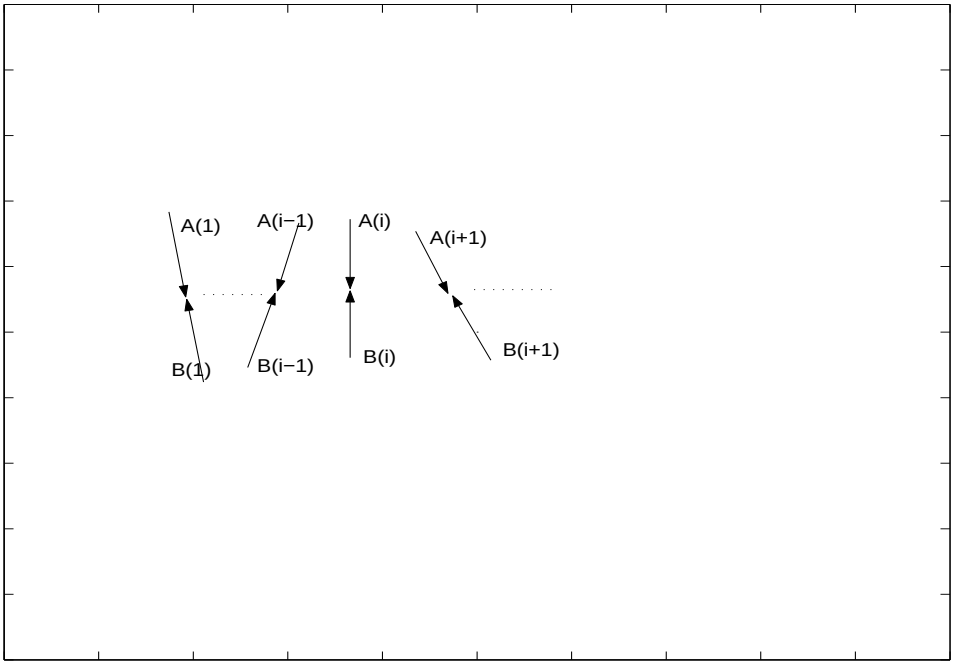

Fig. 3. Generalized problem of type I

The cost matrix to travel from "i" to "$i+1$" can be given by table II.

Let $X_{1i}, X_{2i}, X_{3i}, X_{4i}$ be binary variables. The variable $X_{1i} = 1$ when we choose the path from $A(i)$ to $A(i+1)$ with a corresponding cost C_{i1}; the variable $X_{2i} = 1$

Table II. Generalized cost matrix

$i/i+1$	$A(i+1)$	$B(i+1)$
$A(i)$	C_{i1}	C_{i2}
$B(i)$	C_{i3}	C_{i4}

when we choose the path from $A(i)$ to $B(i+1)$ with an associated cost C_{i2}; the variable $X_{3i} = 1$ when we travel from $B(i)$ to $A(i+1)$ with a corresponding cost C_{i3}; and the variable $X_{4i} = 1$ when we travel from $B(i)$ to $B(i+1)$, the cost of travel is C_{i4}. As we can choose only one path to travel from "i" to "i+1" only one of $X_{1i}, X_{2i}, X_{3i}, X_{4i}$ can be equal to 1 at a time and the other three variables equal zero. This leads us to the following equation:

$X_{1i} + X_{2i} + X_{3i} + X_{4i} = 1$

For example, if we choose the path from $A(i)$ to $A(i+1)$, i.e. if $X_{1i} = 1$, then we can not travel from $A(i)$ to $B(i+1)$ and hence $X_{2i} = 0$; similarly, $X_{3i} = 0$ and $X_{4i} = 0$.

It also implies that, during the travel from $(i-1)$ to (i), we chose $A(i)$ as the destination and not $B(i)$. Thus we did not travel from $A(i-1)$ to $B(i)$ i.e. $X_{2(i-1)} = 0$; also we did not choose the path from $B(i-1)$ to $B(i)$ i.e. $X_{4(i-1)} = 0$.

Only possible ways to come to $A(i)$ are to travel from $A(i-1)$ to $A(i)$ i.e. $X_{1(i-1)} = 1$ OR to choose the path from $B(i-1)$ to $A(i)$ i.e. $X_{3(i-1)} = 1$.

This leads to following equations:

$X_{1i} + X_{2i} + X_{2(i-1)} + X_{4(i-1)} = 1$

$X_{3i} + X_{4i} + X_{1(i-1)} + X_{3(i-1)} = 1$

Thus the generalized problem of type I can be posed as the following Linear Program (LP):

$\min \sum_{i=1}^{n} \sum_{j=1}^{4} C_{ij} * X_{ji}$

Subject to :

$X_{ji} \geq 0 \qquad$ where $i = 1, 2, ..., n$ and $j = 1, 2, 3, 4$

$X_{1i} + X_{2i} + X_{2(i-1)} + X_{4(i-1)} = 1 \qquad$ where $i = 2, 3, ..., n$

$X_{3i} + X_{4i} + X_{1(i-1)} + X_{3(i-1)} = 1 \qquad$ where $i = 2, 3, ..., n$

$X_{1i} + X_{2i} + X_{3i} + X_{4i} = 1 \qquad$ where $i = 1, 2, ..., n$

This LP can be solved efficiently using a simplex technique.

1. Sample Problem

Consider a sequence of the targets given as A-B-C-D-E. Specified angles are as shown in the figure 4. Cost associated with different paths is shown in table III. Using these

Table III. Cost calculation for the sample problem

A/B	1	2	B/C	1	2
1	351	216	1	234	205
2	216	351	2	205	234
C/D	1	2	D/E	1	2
1	222	219	1	272	253
2	219	222	2	253	272

values, we can formulate the LP problem as:

Minimize

$351X_{11} + 216X_{21} + 216X_{31} + 351X_{41} + 234X_{12} + 205X_{22} + 205X_{32} + 234X_{42} + 222X_{13} +$

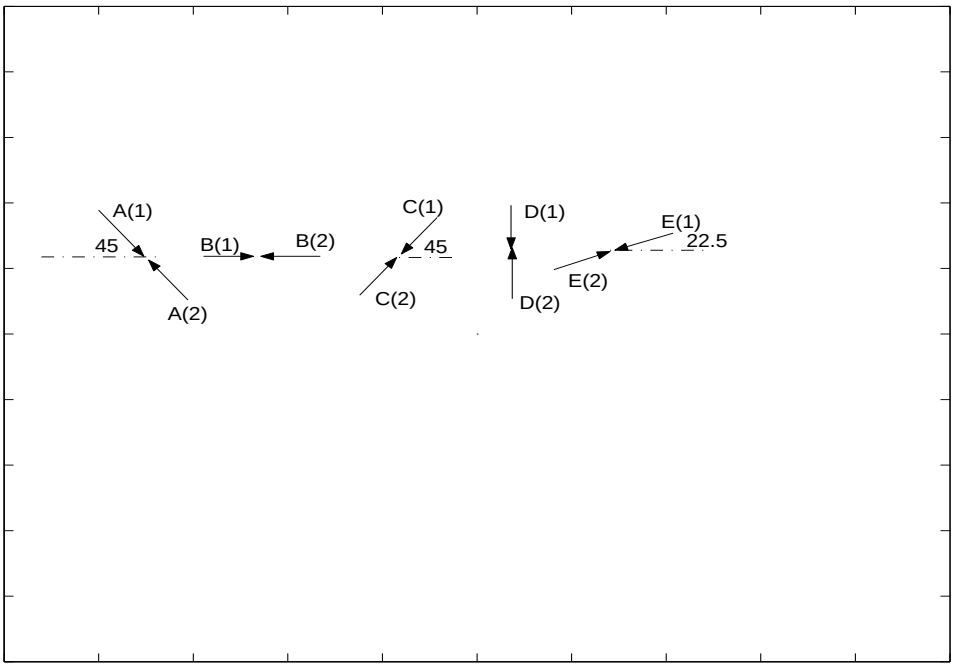

Fig. 4. Application of the solution to the problem of type I

$219X_{23} + 219X_{33} + 222X_{43} + 272X_{14} + 253X_{24} + 253X_{34} + 272X_{44}$

Subject to:

$X_{11} + X_{21} + X_{31} + X_{41} = 1$

$X_{12} + X_{22} + X_{32} + X_{42} = 1$

$X_{13} + X_{23} + X_{33} + X_{43} = 1$

$X_{14} + X_{24} + X_{34} + X_{44} = 1$

$X_{12} + X_{22} + X_{21} + X_{41} = 1$

$X_{13} + X_{23} + X_{22} + X_{42} = 1$

$X_{14} + X_{24} + X_{23} + X_{43} = 1$

$X_{32} + X_{42} + X_{11} + X_{31} = 1$

$X_{33} + X_{43} + X_{12} + X_{32} = 1$

$X_{34} + X_{44} + X_{13} + X_{33} = 1$

$X_{ji} \geq 0 \qquad$ *where* $i \in \{1, 2, 3, 4\}$ *and* $j \in \{1, 2, 3, 4\}$

We can solve this LP using the Simplex method. After solving, we get the optimal value of the objective function as 893. The solution is:

$X_{31} = 1$, $X_{22} = 1$

$X_{33} = 1$, $X_{24} = 1$

$X_{ij} = 0$ if $\{(i, j) \neq (3, 1), (3, 3), (2, 2), (2, 4)\}$.

Thus the optimal path of travel for the given problem is:

A2 - B1 - C2 - D1 - E2

C. Solution to the Problem of Type II

In order to explain the suggested algorithm, we define the UV problem on a graph. Each target and a UV in the problem is considered as a node in a graph. The angle of approach to each target is known. We can find the shortest route between two targets by treating a UV as a dubin's car and thus calculate the cost of travel between two targets. Distance or cost of travel from one node to other is given by the weight of the link joining those two nodes. The problem is to visit all target nodes exactly once, starting from one or more UV nodes and to minimize the total cost of travel. We explain the algorithm considering only one UV. The algorithm can easily be extended to the case of multiple UVs.

1. Suggested Algorithm

- Use the shortest weight path algorithm such as Dijkstra's algorithm and find the "shortest/cheapest" route from each node to every other possible node. Consider all "cheapest" paths with same cost. For example, in the figure 5, we consider C-E-F as well as C-A-B-F.

- Count the number of nodes covered by each path.
- Select the paths covering the same number of nodes and arrange them in different bins in the order of increasing cost.
- Arrange these sets in decreasing order of the number of nodes covered. The first set in the sequence would have paths covering the maximum number of nodes and the last set would have the paths covering the minimum number of nodes (two).
- Choose first path in the first set (it covers the maximum number of nodes and has the least cost in that set) and then choose other paths (including those in the first set) based on following set of rules (refer to figure 5):
  - Discard paths involving the links already chosen. For example, discard the path C-A-D if we have already chosen C-A-B-F.
  - Discard paths containing nodes that have already been shared by two chosen links, one of them being an incoming and other being an outgoing link. For example, discard D-B-E if we have chosen C-A-B-F. Here node B has already been shared by A-B (incoming) and B-F (outgoing).
  - Discard paths with same starting and ending node as that of one of the chosen paths. For example, discard the path C-E-F if we have already chosen C-A-B-F.
  - Discard paths that start at the last node of a sequence and end at the first node of the same sequence. For example, if we have chosen different paths and all of them together form a sequence C-A-B-F-D, then discard D-C.
- After choosing paths from the first set, consider the next set and choose possible paths using the same set of rules.

- Continue till we choose paths from the last set that has paths covering least number of nodes.
- If we have paths with same cost and same number of nodes covered in a set, permute their positions in that set and repeat the procedure to find a new final route and a new total cost. We can ignore the paths that are guaranteed to be excluded in the final route. We need not permute such paths. For example, if we have chosen C-A-B-F and paths C-A, A-B, B-F, A-D, D-B have same cost, we can ignore all these five paths and we need not permute their positions as we can never choose these paths for the final route or we have already chosen them.
- Choose the route with the least total cost.

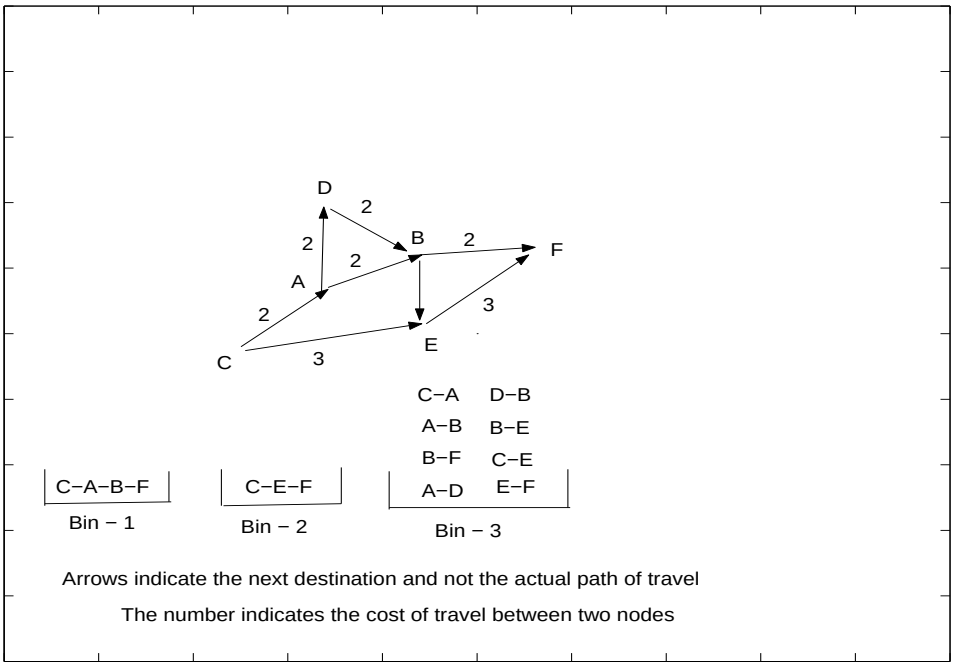

Fig. 5. Example graph

If we have total “n” nodes or targets, then in case of an asymmetric problem we have to deal with “n(n-1)” number of paths and in case of a symmetric problem the number reduces to “n(n-1)/2”.

a. Example Problem

The following problem compares the algorithm explained above with the “Branch & Bound” method and the “Assignment” method. We solve this problem using this algorithm and then compare the result with results obtained from other 2 methods. In this problem, the position of a UV is not fixed at any particular node. The cost of travel between two nodes is given by the table IV. In the table, the first column indicates the starting nodes and the first row indicates the ending nodes. This is an

Table IV. Cost matrix of the example problem

	A	B	C	D	E
A	0	2	5	7	1
B	6	0	3	8	2
C	8	7	0	4	7
D	12	4	6	0	5
E	1	3	2	8	0

asymmetric problem as $d(A, C) = 5$ and $d(C, A) = 8$.

Step 1: Find all possible cheapest paths from each node to every other possible node. Table V shows starting node in the first column, the ending node in the second column, the cheapest / shortest path between these nodes is given in the third column. The fourth column shows the number of nodes covered in the cheapest path while the

fifth column gives the total cost of the path.

Table V. Shortest path between two nodes

Start	End	Shortest route	Number of nodes covered	Total cost
A	B	$A-B$	2	2
A	C	$A-E-C$	3	3
A	D	$A-E-C-D$ or $A-D$	4 or 2	7
A	E	$A-E$	2	1
B	A	$B-E-A$	3	3
B	C	$B-C$	2	3
B	D	$B-C-D$	3	7
B	E	$B-E$	2	2
C	A	$C-E-A$ or $C-A$	3 or 2	8
C	B	$C-B$	2	7
C	D	$C-D$	2	4
C	E	$C-E$	2	7
D	A	$D-E-A$	3	6
D	B	$D-B$	2	4
D	C	$D-C$	2	6
D	E	$D-E$	2	5
E	A	$E-A$	2	1
E	B	$E-B$	2	3
E	C	$E-C$	2	2
E	D	$E-C-D$	3	6

Step 2: Arrange different sets as explained in the procedure. The order of different paths in these sets and the order of sets is shown in figure 6:

Step 3: Following the set of rules given in the procedure, we choose different paths

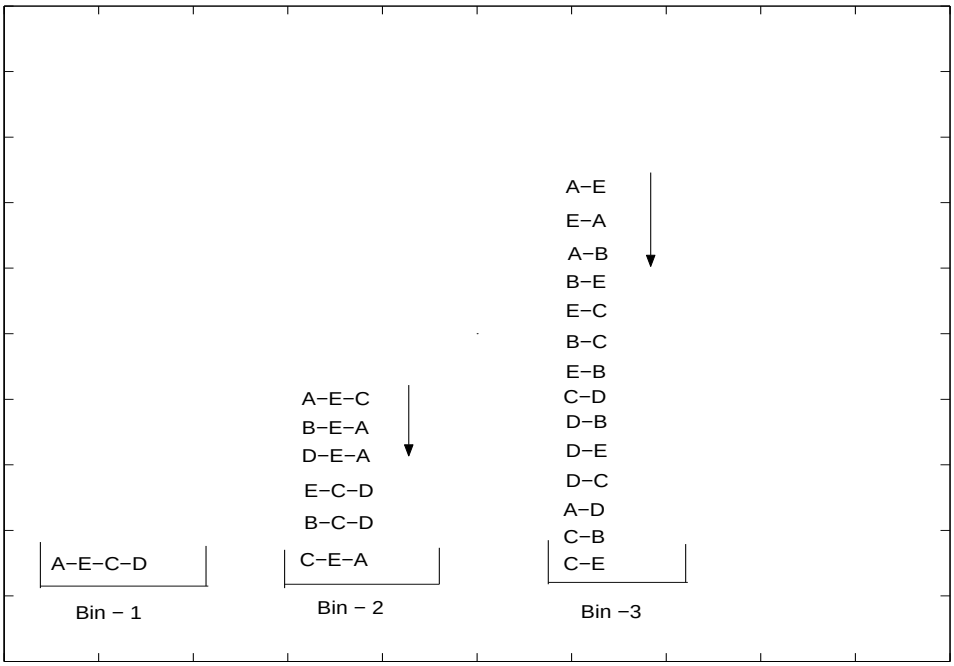

Fig. 6. Example - 1

from the ordered sets shown above.

- Select: (A-E-C-D) & (D-B)
- Complete route of the travel: A-E-C-D-B
- Total cost of the travel: 11

To return to the starting point after covering all the nodes we can use the cheapest path to go from B to A. That gives complete route of travel as "A-E-C-D-B-E-A" with the total cost of the travel being 14.

Using "Branch & Bound" method and "Assignment" approach, we get following result for the same problem. [7]

- Complete route of the travel: A-B-C-D-E-A
- Total cost of the travel: $15 (> 14)$

If we are not required to return to the starting point then, we have the route of travel as "A-B-C-D-E" with the total cost of travel being 14 (> 11).

It can be noted that, with the algorithm explained above, if we return to the starting point, node E is covered twice. It is reasonable to visit nodes more than once while returning to the starting node. In the UV problem, more importance is given to the shortest distance path due to constraint on fuel available.

2. Multiple Vehicle Problem

Even though previously explained algorithm is for planning the motion of one UV, it can still be used to accommodate motion planning for multiple vehicles. Route assignment is done simultaneously for all UVs, instead of considering one vehicle after other. In case of the assignment, where we consider one vehicle at a time, we assign the route from available nodes (nodes that are not already assigned to other UVs). Simultaneous route assignment can be more efficient and can give better solution to the problem.

3. Fuel Constraint Considerations

Case 1: Distance that can be traveled with a given amount of fuel is greater than the distance of the least cost path in the first bin covering the maximum number of nodes.

In this case, limited fuel defines an upper bound on the distance a UV can travel. We use the algorithm proposed here and find the final route for the UV. We change the final destination of the UV by omitting last few nodes from the route to satisfy

the upper bound on the total traveled distance.

Case 2: Distance that can be traveled with a given amount of fuel is less than the distance of the least cost path in the first bin covering the maximum number of nodes.

Let us assume that the maximum number of nodes that can be covered by a path obtained using Dijkstra's algorithm is "n". The least cost path with "n" nodes has the least distance among all possible paths covering "n" nodes. As the given fuel is not sufficient to travel even this least distance, we can not cover "n" nodes with given fuel. We choose the least cost path covering "n-1" nodes. If still the fuel is not sufficient then we move to "n-2" nodes path and so on. Thus finally we find the path with maximum number of nodes that can be covered with the given fuel constraints.

In the UV problem, we have a constraint on the starting node. We can start only from a UV node. In this case, we consider only those paths that start from a UV node. But here too, the two cases explained above apply.

4. Starting Node Constraint Considerations

In a case where we have an "incoming" link to a UV node, we use the algorithm with small modifications. Instead of choosing the overall least cost path covering maximum number of nodes (OLCMN), in the first step, we consider only the paths starting from a UV node. From these paths we choose the one covering maximum nodes having the least cost.

In the second step, we consider all the paths including OLCMN. While choosing the paths for the final route, care must be taken that the starting node of the OLCMN path remains as the starting node of UV. Note that, unlike in OP or VRP, a UV is not required to return to the starting node.

When we are given the positions of UVs, we can not ignore the distance a UV must travel from its position to the first node or first target. This distance can make

a significant difference in an optimal route to cover all the nodes. To overcome this, we consider the positions of UVs as additional nodes that has only "outgoing" links, thus any UV does not try to cover this fictitious node in its route and these nodes automatically become the starting nodes. In this case, we use the original algorithm without any modification.

5. Benefits of the Proposed Algorithm

- The algorithm can be used for the problems where the triangular inequality does not hold true.
- Fuel constraint can be enforced while finding a solution.
- The proposed algorithm can be applied to one UV as well as to a team of multiple UVs located at one or more depots.
- The algorithm can be applied for a symmetric as well as for an asymmetric type of problem.

6. Limitations of the Algorithm

- It is not guaranteed if we can return to the starting node visiting each node exactly once. If the solution gives the route that comes back to the starting point then each node is visited exactly once, else we must add cheapest path from the last node to the starting node and in this case one or more nodes are visited twice on the way back.
- The algorithm reduces to the Cheapest Link Algorithm (CLA) when triangular inequality holds.

- The number of permutations increases with an increase in the number of paths with same number of nodes and cost.

7. Example Problem

The example problem given on page 21 shows that we can get better solutions than the Branch and Bound method or the Assignment approach. The example given below shows that we can not get the best solution for all problems. Consider following symmetric TSP.

Each city is represented with a letter shown in parenthesis. The cities considered in this problem are Galveston (G), Caldwell (C), Houston (H), Bryan (B), Austin (A), Killeen (K), Waco (W), Grand Prairie (GP) and Lewisville (L). The table VI gives the cost of travel from one city to another. The first column of the table indicates the starting city while the first row of the table indicates the ending city.

Table VI. Multiple cities problem

	G	C	H	B	A	K	W	GP	L
G	–	141	51	134	189	213	209	268	292
C	141	–	90	23	63	73	75	149	175
H	51	90	–	85	139	162	160	224	248
B	134	23	85	–	85	86	76	142	167
A	189	63	139	85	–	49	85	162	189
K	213	73	162	86	49	–	43	114	141
W	209	75	160	76	85	43	–	78	106
GP	268	149	224	142	162	114	78	–	27
L	292	175	248	167	189	141	106	27	–

Step 1: Find all possible cheapest paths from each node to every other possible node. As the problem is symmetric, path from A - B is same as B - A. We prepare the table VII in the same way as we did in section A example problem.

Table VII. Shortest path between two cities

Start	End	Shortest route	Number of nodes covered	Total cost
G	C	$G-H-C$ or $G-C$	3 or 2	141
G	H	$G-H$	2	51
G	B	$G-B$	2	134
G	A	$G-A$	2	189
G	K	$G-H-K$ or $G-K$	3 or 2	213
G	W	$G-W$	2	209
G	GP	$G-GP$	2	268
G	L	$G-L$	2	292
C	H	$C-H$	2	90
C	B	$C-B$	2	23
C	A	$C-A$	2	63
C	K	$C-K$	2	73
C	W	$C-W$	2	75
C	GP	$C-GP$	2	149
C	L	$C-L$	2	175

Table VII Continued.

H	B	$H-B$	2	85
H	A	$H-A$	2	139
H	K	$H-K$	2	162
H	W	$H-W$	2	160
H	GP	$H-GP$	2	224
H	L	$H-L$	2	248
B	A	$B-A$	2	85
B	K	$B-K$	2	86
B	W	$B-W$	2	76
B	GP	$B-GP$	2	142
B	L	$B-L$	2	167
A	K	$A-K$	2	49
A	W	$A-W$	2	85
A	GP	$A-GP$	2	162
A	L	$A-GP-L$ or $A-L$	3 or 2	189
K	W	$K-W$	2	43
K	GP	$K-GP$	2	114
K	L	$K-GP-L$ or $K-L$	3 or 2	141
W	GP	$W-GP$	2	78
W	L	$W-GP-L$	2	105
GP	L	$GP-L$	2	27

Step 2: Arrange different sets as explained in the procedure. The order of different paths in sets and the order of sets is as shown below:

(W-GP-L), (L-GP-W), (G-H-C), (C-H-G), (K-GP-L), (L-GP-K), (A-GP-L), (L-GP-A), (G-H-K), (K-H-G)

It is obvious that, if we choose two paths from first four, then all others are invalid paths as they have common incoming or outgoing node. This leads us to 4 different selections given below:

i) Select (W-GP-L) & (G-H-C) ii) Select (L-GP-W) & (C-H-G)

iii) Select (W-GP-L) & (C-H-G) iv) Select (L-GP-W) & (G-H-C)

Since it is a symmetric problem, it suffices to consider just first two selections (i) & (iii).

(i) Select (W-GP-L) & (G-H-C):

After the selection of these two paths, the set containing paths with 2 nodes has only following paths. They are arranged in the order of increasing cost as shown in figure 7 .

(C-B), (K-W), (A-K), (K-A), (C-A), (C-K), (C-W), (B-W), (B-A), (A-B), (A-W), (B-K), (K-B), ((B-G), (L-K), (L-B), (A-G), (L-A), (K-G), (L-G)

(ii) Select (W-GP-L) & (C-H-G)

In the same way we have following set with 2-nodes paths.

(B-C), (K-W), (A-K), (K-A), (A-C), (K-C), (B-W), (B-A), (A-B), (A-W), (B-K), (K-B), (G-B), (L-K), (L-B), (L-C), (G-A), (L-A), (G-W), (G-K)

Step 3: Following the set of rules given in the procedure, we choose different paths from ordered sequence given above.

(i) Select (W-GP-L) & (G-H-C): For this case, following the procedure described,

- We select: (W-GP-L), (G-H-C), (C-B), (K-W), (A-K), (B-A), (L-G)
- Complete route of travel is G-H-C-B-A-K-W-GP-L-G
- Total cost of the travel: 738

(ii) Select (W-GP-L) & (C-H-G): For this case, we get following results,

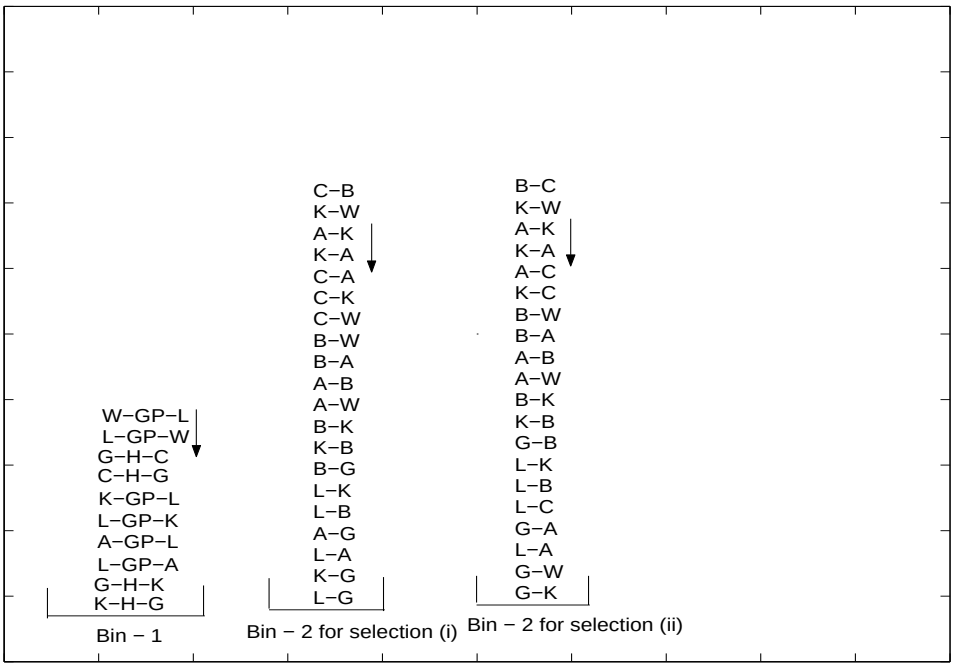

Fig. 7. Multiple cities problem

- We select: (W-GP-L), (C-H-G), (B-C), (K-W), (A-K), (L-B), (G-A)
- Complete route of travel is A-K-W-GP-L-B-C-H-G-A
- Total cost of the travel: 717

Thus we select second solution with total travel cost = 717.

When we solve this problem using RNNA, CLA or Branch & Bound method, we get following solutions. [5]

(i) RNNA

- Complete route of the travel: G-H-B-C-A-K-W-GP-L-G
- Total cost of the travel: 711

(ii) CLA

- Complete route of the travel: G-H-B-C-A-K-W-GP-L-G
- Total cost of the travel: 711

(iii) B & B

- Complete route of the travel: H-C-A-K-W-GP-L-B-G-H
- Total cost of the travel: 702

Thus, in this example, we can not get the best solution with the algorithm but it is close to the optimal solution. If we assume that the solution given by B&B is the optimal solution, then we can compare this solution with the solution that gives total cost of 738. Interestingly, both the solutions have following sequences, A-K-W-GP-L and G-H-C. The only difference is in the placement of the city B (Bryan).

Also if we compare the first solution with that of RNNA or CLA, it can be noticed that the only difference between the two is that B and C are interchanged. RNNA has H-B-C-A, while the solution with proposed procedure has H-C-B-A.

8. Multiple UVs Example Problem

Consider a problem with 3 UVs and 8 targets. Let us assume that initial positions and headings of each UV is given and there are only "outgoing" links from these positions. Thus we can not return to the starting node. Let A,B,C,D,E,F,G,H be targets and let V_1,V_2,V_3 be UVs. The co-ordinates and headings of targets and vehicles are given in the table VIII. The task here is to visit all targets at a given heading minimizing the total cost of travel.

We can code the algorithm to solve this multiple UV problem. We have developed a MATLAB program that requires positions and headings of UVs and targets as input. First we treat a UV as a Dubin's car to find the cost matrix. Once we find the cost matrix, we can apply the proposed algorithm. The program assigns targets to each vehicle in a particular sequence. It also provides the control for each vehicle to follow the suggested route.

The vehicle route for each vehicle is given as:

V_1 - A - C - H - E - F

V_2 - Does not visit any target

V_3 - D - G - B

The total cost of this assignment is 65.87. The figure 8 shows the proposed motion plan for each UV.

Table VIII. Vehicles and targets positions

	X co-ordinate	Y co-ordinate	Heading in degrees
A	10	5	0
B	35	5	90
C	15	10	0
D	28	10	90
E	20	8	34
F	6	12	45
G	30	15	58
H	18	8	80
V_1	8	2	90
V_2	20	15	0
V_3	30	10	270

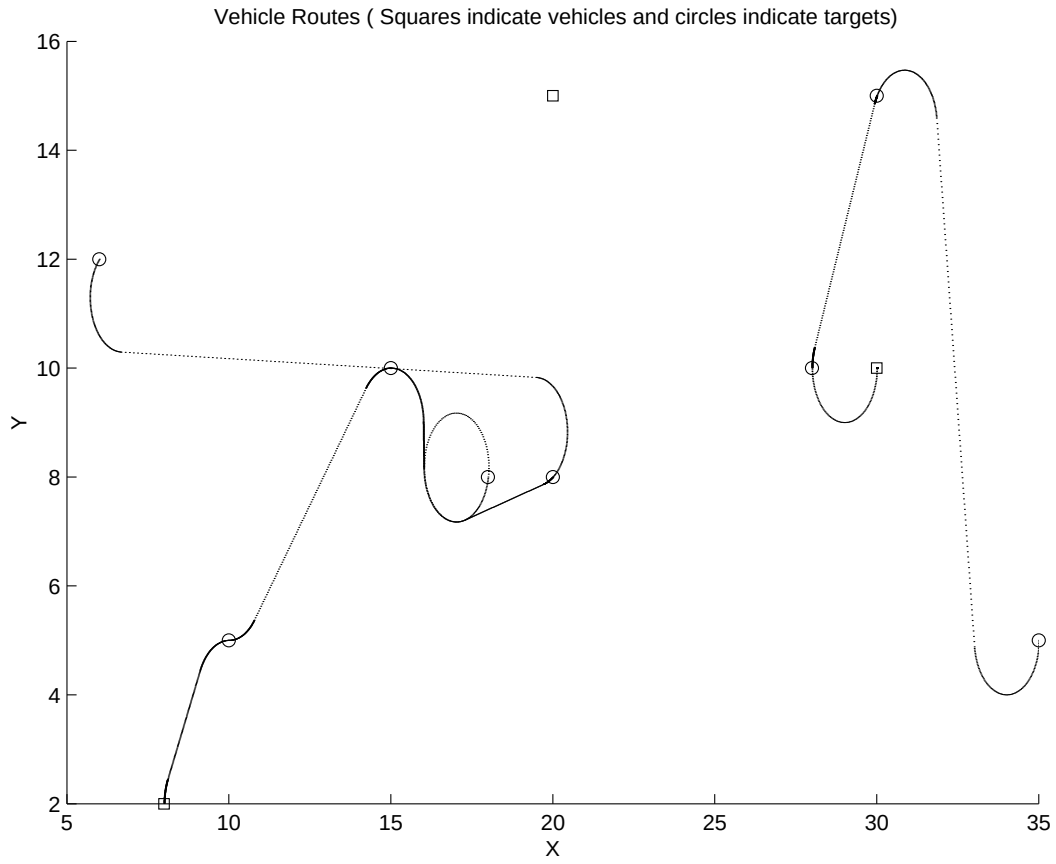

Fig. 8. Multiple vehicles route assignment

CHAPTER IV

CONCLUSION

The heuristic approach proposed to solve the UV problem does not guarantee an optimal solution but it can be effectively used to solve a complicated resource allocation problem. The suggested algorithm can be implemented on MATLAB and it accounts for motion planning constraints along with the fuel constraints for UVs. The algorithm is also effective to solve a multiple UV problem. This work will definitely be useful for future research on the UV problem.

A. Future Work

We have considered a simplified version of the UV problem in this thesis. In general, the tasks a UV has to perform are time constrained. A UV can not perform certain tasks on a target before completing others. For example, a UV can not attack a target before it is classified. Different targets might have different status, some targets might need an assessment of damage and some must be classified. A UV can classify a target followed by an attack or it can classify some targets, assess some targets for damage and then attack a high valued target. We have not considered time constraint in the thesis. We have assumed that all targets are detected as potential targets and they need to be classified (Problem of type I and type II). We have not considered tasks of attack and assessment of damage. Future work can be done to find a solution to the UV problem considering the time constraints.

REFERENCES

[1] D. Applegate, R. Bixby, V. Chvtal, W. Cook, "Dantzig, Fulkerson, and Johnson's Cutting-Plane Method", http://www.math.princeton.edu/tsp/dfj, June 2001

[2] L. Bowen, Introduction to Contemporary Mathematics, http://www.ctl.ua.edu/math103/hamilton/quick.htm, visited Aug. 2002

[3] I.M. Chao, B.L. Golden, E.A. Wasil, "A Fast and Effective Heuristic for the Orienteering Problem", *European Journal of Operational Research* Vol. 88, pp. 475-489, 1996

[4] J. Cirasella, D.S. Johnson, L.A. McGeoch, W. Zhang, "The Asymmetric Traveling Salesman Problem: Algorithms, Instance Generators and Tests", *Proceedings of ALENEX 01, Springer Lecture Notes in Computer Science 2153*, New York, pp. 32-59, 2001

[5] S. Forman, TSP Generator, http://www.sju.edu/∼sforman/research/usa_tsp.html, visited Aug. 2002

[6] B.E. Gillett, L.R. Miller, "A Heuristic Algorithm for the Vehicle Dispatch Problem", *Operations Research* Vol. 22, pp. 340-349, 1974

[7] P.K. Gupta, D.S. Hira, *Operations Research*, S.Chand & Co., New Delhi, 1999.

[8] D. Johnson, G. Gutin, L.A. McGeoch, A. Yeo, W. Zhang, A. Zverovitch, "Experimental Analysis of Hueristics for the ATSP" *The Traveling Salesman Problem and Its Variations*, Kluwer Academic Publishers, Dordrecht, 2002.

[9] V. Kapila,G. Yang, "Optimal Path Planning for Unmanned Air Vehicles with Kinematic and Tactical Constraints", *Proceedings of the 41st IEEE Conf. Decision and Control*, Las Vegas, vol. 2, pp. 1301-1306, Dec. 2002

[10] P. Moscato, TSPBIB Home Page, http://www.densis.fee.unicamp.br/∼moscato/TSPBIB_home.html, Oct. 2000

[11] D. Swaroop,"Teaming strategies for a resource allocation and coordination problem in the cooperative control of UAVs", AFRL report, Wright Patterson Airforce Base, Aug. 2001.

[12] T. Tsiligirides,"Heuristic Methods Applied to Orienteering", *Journal of Operational Research Society* Vol. 35, No. 9, pp. 797-809, 1984

APPENDIX A

We have provided the MATLAB program, in the form of a CD, that can be used to solve a simplified UV problem for three UVs and eight targets. The program requires positions and headings of UVs and targets as input. The program gives motion plan for each UV as an output.

VITA

Sudhir Pargaonkar

B.E. Karnataka Regional Engineering College, Surathkal, India

M.S. Texas A&M University, College Station, TX, USA

2710 Grants Lake Blvd, K-4

Sugarland, TX - 77479