Calculation of the shortest-time path for traversal of an obstacle course by a robot

Rishi T. Khar*, Ernest L. Hall**
Center for Robotics Research, University of Cincinnati, Cincinnati, OH 45221-0072

ABSTRACT

The problem of sequencing the movement of a robot so that it can carry out a given task in the minimum required time is of considerable importance, because of the efficiency of such a solution. The problem considered is an application of this idea, as applied to the context of the Navigation Challenge in the International Guided Vehicle Competition (IGVC). The objective is to find a sequence of points and a path in space that the robot has to traverse in order to complete the objective of the competition. A mathematical programming based model and example solution for the Bearcat Robot is given. The challenge in this event is for a robot to autonomously travel, using Differential GPS, from a starting point to a number of target destinations, while recognizing and avoiding the obstacles present, given only a map showing the coordinates of those targets, in the least possible time. The solution can be implemented easily using the Excel Solver, or AMPL. These solutions are practically applicable and easy to run in the competition since they give the sequence of points to be followed. In addition, the program is used together with a heuristic for situations where there are velocity constraints on the robot.

Keywords: Navigation, optimization, obstacles, waypoint, network

1. INTRODUCTION

In today's competitive economic and industrial scenario, where there is a greater emphasis on savings and efficiency of operation, management science and operations research has had a profound influence in the way these goals are achieved. Applying the principles and findings of applied statistics, mathematical modeling, management science and operations management has led to more efficient functioning of the industries. This is true for the service sector, but more so for the manufacturing sector. Operations research in the wide sense of the term has been applied in varied situations. The techniques have been applied for modeling, solving, simulating and analyzing many aspects of modern production techniques.

Several practical problems are related to determining the ideal sequence for a series of tasks, comparing two models of production or making the optimal choice of a means of production. These include locating company branches, drafting production schedules, duty-rotas, establishing the logistics of material delivery, placing the supply chain optimally, material utilization, routing delivery trucks and many more applications.

The solution to the above types of problems invariably has to do with making the best possible choice from among a great many possibilities. The number of possibilities increases with the number of variables and in most real situations this number is prohibitively large. In principle this choice could be made by examining all the possible options one by one and then deciding on the best among these based on our criteria; but the numbers involved in practice are so large that this simple approach would take far too long, even using very fast computers. Some types of problems may in fact not be solvable at all! So, more intelligent search strategies are needed. With the greater number of situations that these kinds of mathematical techniques are being applied to and the increasing complexity of these situations, this effort has taken an urgent focus. This issue has been the focus of a considerable body of research in recent years. The primary research has been in the development of faster and more efficient algorithms to solve these problems. The other focus has been on more efficient models and exploiting the structure of these models for solving the problems faster. In addition heuristic techniques have found appeal because for several of these problems have no exact solutions.

*kharr@email.uc.edu. **Ernie.Hall@uc.edu; www.robotics.uc.edu

These methods have served to help in overcoming to a large extent the problems associated with these real world situations. A well-known example is the Traveling Salesman Problem.

1.1 Problem background and research

1.1.1 The Traveling Salesman Problem

The Traveling salesman problem (**TSP)** is this: given a finite number of "cities" along with the cost of travel between each pair of them, find the cheapest way of visiting all the cities and returning to your starting point[8]. A variation would be to find the shortest route between a number of cities to be visited. This shortest route could be in terms of the time taken for the travel or the actual distance traveled. In practice there are many instances of this problem, from truck routing and drilling the holes in computer circuit boards to robot control and operation. The similarity in nature and structure between the classical or Hitchcockian Transportation problem and the TSP; as also the differences between these two problems have a great bearing on the solution techniques for the TSP.

This combinatorial optimization problem has been proven to have such computational complexity high that solving this problem in normal time would be very difficult. The size of the problem explodes with even a moderate numer of cities. Various approaches have been employed to heuristically or approximately solve this problem. Software developed in the past years now permits solution of problems with more than 3000 'cities' and algorithms exist that solve this to 11000 'cities'. This research also has produced new mathematical techniques to cope with related problems.

1.1.2 Vehicle Routing Problem

The Vehicle routing problem (**VRP**) consists of finding optimal delivery routes from one or more depots to a set of geographically scattered points, which are demand centers. Without other complications, this is the same as a shortest path problem. In its most complex form, the VRP is a generalization of the TSP, as it can include additional time and capacity constraints, precedence constraints and more.

The VRP is a well-known integer programming problem which falls into the category of NP-complete problems, meaning that the computational effort required to solve this problem increases exponentially with the problem size. For such problems it is often desirable to obtain approximate solutions, provided they can be found fast enough and are sufficiently accurate for the purpose. Usually this task is accomplished by using various heuristic methods, which rely on some insight into the problem nature.

In the classical vehicle routing problem, there are 'n' customers to be supplied from a few supply depots. Customer 'i' $(i = 1, \ldots, n)$ has a requirement of 'q_i', and each vehicle has a capacity Q (in some variants of the problem capacities vary according to the vehicle). There is a cost 'c_{ij}' and a time 't_{ij}' for traveling between each pair of customers 'i' and 'j' $(i = 1, \ldots, n, j = 1, \ldots, n, i \neq j)$. It is required to find tours for each of the vehicles, where each tour starts and ends at the depot, so that each customer is visited once, the vehicle capacity constraints are satisfied, and the total cost is minimized[8]. In a common variant of this basic model, there is a time window within which the delivery must take place.

1.1.3 A survey of current solution techniques

Researchers apply several techniques including constraint programming, linear and integer programming, optimizing algorithms for selected problems, and approximation algorithms and heuristics for large-scale problems. The solution methods are classified based on the solution technique.

Exact Approaches: This approach proposes to compute every possible solution and then choose the best among these or until some index of solution accuracy is reached.
Example: Branch and bound (up to 100 nodes), Branch and Cut algorithms
Heuristics: Heuristic methods perform a relatively limited exploration of the search (solution) space and produce good quality solutions within modest computing times.
Example: Matching Based Multi-route Improvement Heuristics
MetaHeuristics: In metaheuristics, the emphasis is on performing a deep exploration of the most promising regions of the solution space. The quality of solutions produced by these methods is much higher than that obtained by classical heuristics. They are used in combination with other breadth based search techniques[5].

Example: Constraint Programming, Genetic Algorithms, Simulated Annealing, Tabu Search
These methods are well suited to practical applications, for instance geometric methods stemming from linear programming apply to VLSI design, pattern recognition and robotic arm path planning[6].

2. DESCRIPTION

2.1 An introduction to the problem

The Bearcat III and now the Bearcat Cub have been the pride of the University of Cincinnati (UC) Robotics team. The U C Robotics team takes part in the renowned Annual Intelligent Ground Vehicles Competition. One of the competitive challenges at this competition is the Navigation Challenge. The challenge in this event is for a robot to autonomously travel from a starting point to a number of target destinations (waypoints or landmarks) and return to home base, given only a map showing the coordinates of those targets. Coordinates of the targets will be given in latitude and longitude as well as in meters on an x-y grid. The contestants are expected to use Differential GPS. Obstructing the way between these waypoints will be obstacles that the robot has to recognize and avoid. The competition is judged based on the time taken to traverse all the waypoints[7]. This aspect of the problem is one that lends itself easily to application of optimization techniques.

The problem at hand is of finding the optimal path that the robot should take for this navigation competition. The conditions being that certain specified waypoints have to be traversed, while at the same time, the obstacles have to be avoided. From the structure and the constraints on this problem it became evident that this problem is similar to the famous Vehicle Routing problem. There are some key differences between the two problems and their scope and these will become evident as the model is explained.

The Bearcat III currently operates on an algorithmic program that takes as its input an ordered sequence of the waypoints to be traversed. When an obstacle is recognized on this pre-programmed path, the obstacle avoidance algorithm takes over control of the robot till it successfully negotiates it past the obstacle. Then once again the robot follows the pre-programmed path[1,2]. So, the challenge was to find a method of having a program, which would, as its solution give a sequence of waypoints that would be the optimal least-distance path.

2.2 Modeling Approach

The similarity in the nature of the VRP and the Robot Navigation (**RN**) problem suggested a similar approach in solution. It was recognized that the differences between the two problems made them sufficiently different to enable a solution to the RN problem. The important differences are in the size of the problem and in the presence of obstacles; equivalent to ‘cities’ that do not have to be visited in the VRP. Similarly in the RN problem the obstacles do not have to be ‘visited’. Typically the number of waypoints for the competition would be around 7 to 10 and because of this smaller size of the problem it means that the RN problem can actually be solved exactly without resorting to algorithms or heuristic rules. These are the two most significant differences between the VRP and the RN problem. Thus, the traditional techniques of linear and integer programming can be applied to solving the RN problem.

To solve this problem the obstacles, waypoints, the starting point (origin) and the destination point are considered as nodes in a network. This approach would allow for using the techniques of Network flow problems or for modeling the problem as an Integer Programming model. The nodes are all connected with each other because the robot could move from any node to any other, with an important exception; the competition rules specify that the robot has an origin that it must return to after the waypoints have been traversed. So, the Network graph (Digraph) is maximally connected. Because this is so, the Integer Programming option is easier to implement[9,10].

The nodes are all connected by means of arcs. In the classical problem (VRP), the nodes are supply or demand points and the arcs represent the costs involved in the transit of vehicles along these arcs. In the case of the RN problem, we need to minimize the time taken by the robot to traverse the entire course (from the competition point of view), while traversing the waypoints and avoiding the obstacles; or equivalently, to minimize the total distance the robot needs to travel along some path that traverses all the waypoints and avoids the obstacles. Thus, the costs on the arcs, in the VRP, can be replaced in the RN problem by the (physical) distance between the nodes.

Initially, to avoid the looping effect of returning to the starting point and the complication to the model, a simplifying assumption of having a destination node, separated in space from the origin node by a very small distance, was made. These two nodes are physically identical in terms of location, but for mathematical ease are assumed as distinct nodes. The complete model has n(n-1) decision variables and a variable number of constraints that grow greatly with the number of nodes.

The decision variables refer to the flow along the arcs. This flow is constrained to be a binary variable and this means that these variables are only going to give a decision whether the robot has to travel along a certain arc or not. In other words this means that the decision variable associated with each arc decides whether that arc connecting two nodes is part of the solution, i.e., part of the optimal path, or not. So, the decision variable related to a particular arc takes a value of one when this arc is part of the optimal path solution calculated by the program. If that arc is not part of the optimal path the decision variable associated with that arc takes the value of zero. Thus, reading the values of the decision variables will give those arcs that are part of the optimal path. Since the starting and destination nodes are known and specified by the user, the path that the robot has to traverse which is the shortest distance path will be obtained. Since travel along the two directions of an arc can lead to very different paths, each node is connected to the others by two distinct arcs, one for each direction. The classical VRP does not have single origin and destination nodes and that brings in additional constraints for the RN problem, reducing the solution space to be searched; although, at the same time, it increases the number of constraints.

3. SOLUTION AND RESULTS

3.1 Model formulation

The following data, which are the model parameters and the decision variables, are required in the model:

'i, j' = the index set of the nodes (model parameter)

Assuming that there are 'n' total nodes in the system, both 'i' and 'j' will take values 1 through 'n'. For simplicity, it is assumed that node 1 represents the Origin node and node 'n' will refer to the Destination node, which themselves are identical for all practical purposes.

'C_{ij}' = the distance between node i and node j (model parameter)

The following decision variables are defined:

X_{ij} = 1 if arc i-j is present in the optimal solution

= 0 if arc i-j is not present in the optimal solution

Let 'K' be the set of those nodes that represent the obstacles, so some i, j belong to this set K.

The formulation of the distance-minimization model is given below.

Objective Function: Minimize $\sum X_{ij} C_{ij}$ (Summed over all i and j, $i \neq j$)

$i, j \in \{1, 2, 3 \ldots\ldots n\}$

Subjected to constraints:

$$\sum X_{ij} = 1 \qquad \forall\ i;\ i \notin K \quad \text{(summed over j = 1 to n)} \tag{1}$$

$$\sum X_{ij} = 1 \qquad \forall\ j;\ j \notin K \quad \text{(summed over i = 1 to n)} \tag{2}$$

$$X_{1n} = 0 \tag{3}$$

$$X_{n1} = 1 \tag{4}$$

$$\sum X_{ij} = 0 \qquad \forall\ i;\ i \in K \quad \text{(summed over j = 1 to n)} \tag{5}$$

$$\sum X_{ij} = 0 \qquad \forall\ j;\ j \in K \quad \text{(summed over i = 1 to n)} \tag{6}$$

$$\sum X_{ij} \leq m-1 \quad \forall\ i, j;\ i \neq j \text{ (For all nodes taken m at a time)} \qquad m>2,\ n-m>=1 \tag{7}$$

X_{ij} are binary variables (can take values 0 or 1 only)

3.2 Model explanation

The objective function stated is to minimize the distance of the path traveled by the robot. Constraint set (1) and (2) make sure that each node will allow the robot to pass through it only once. (1) represents the flow from all other nodes into the node under consideration. So this constraint says that the combined 'flow' from all the other nodes into this node is one unit, in other words, only one node can send the robot into this node, making sure that the robot does not return to this node. (2) represents the flow from the node under consideration to all the other nodes. Thus, this constraint says that the combined 'flow' from this node into all the other nodes is one unit, in other words, saying that any node will let the robot pass from it only once. These two constraint sets ensure that each waypoint is only traversed, from and to, only once.

Similarly constraints (3) and (4) ensure that the robot does not go directly from the origin to the destination or from the destination node back into the rest of the nodes! Constraint (3) specifies that the origin cannot send the robot to the destination directly, but has to send it to one of the other nodes. At this stage Constraints (1) and (2) ensure that this node then passes the robot on and so on for the next node. In this way we are ensuring that each node has to be passed through, and only once. Constraint (4) makes sure that the destination node does not send the robot anywhere else but to the origin node. Constraints (5) and (6) ensure that the obstacle nodes are not traversed at all as they set the flow in and also flow out of such nodes to be zero. Constraint (7) is essential as it is what sets the constraint that the solution has to give a sequence of points. The fact that this sum has to be at least equal to 1, forces the fact that this node has to send the robot further on to another node, thus ensuring a connected solution. This is referred to as the Subtour elimination constraint and this set ensures the connectivity of the solution obtained. This set makes sure that the points are connected in a cohesive manner and the path is truly a path from the origin to the destination. This is a summary of the purpose of the various components of the model.

The output of this model gives a set of "X_{ij}'s" that take the value 1, indicating that the arc between these nodes i, j has to be traveled by the robot as part of the optimal path. If the "X_{ij}'s" take a value '0' it means that that arc between nodes i, j should not be traveled by the robot as it is not part of the optimal path solution From the positions of the nodes along the path, the optimal sequence will be evident, starting from the origin node and ending at the destination node.

It was observed that this model could be further refined, as the presence of obstacle nodes was not effecting the optimal path solution, as they had no flow into or out of them. In addition the constraint set (7), called the Subtour elimination constraints in the classic formulation of the VRP, could be further consolidated so that only the relevant ones can be identified and the model run accordingly, though this will require an initial run of the full model. This means that the needed constraints (cuts) will be added on an ad-hoc basis.

3.2.1 Model nuances

It was seen that the obstacle nodes are only adding to the computational complexity of the problem without really contributing to the solution and yet they could not have been dropped from the model without accounting for their removal. This led to a very important assumption that would account for the obstacle nodes while at the same time decrease the size of the problem. This is that, given a solution considering a situation where only the waypoint positions are known (which is how the actual situation during the Navigation challenge is), once the obstacle positions are also known or estimated, this solution will be altered. This is because these positions will affect the true distance of traversal between the nodes.

An obstacle node, when it has been added to the existing network, could lie in one of three distinct positions:
1) On a straight line connecting two way points, or connecting the origin/destination to a way-point
2) Somewhere in the region of space between the straight lines connecting the existing nodes/origin/destination.

As can be seen, if any of these obstacle nodes fall in a position where their location is as in case 1 or 2 above, they may make the existing solution non-optimal. To the distance between two nodes, now with an obstacle node between them, would be added the distance it would take to avoid the obstacle. Thus if they were part of the optimal solution without considering the obstacle positions, they may no longer be so. Similarly, other paths that had been left out of the optimal solution without obstacles may now become a part of the optimal path. This argument drives to the

conclusion that the presence of the obstacles can be modeled as an increase in the distances between two nodes, depending on what the position of the obstacle is.

For obstacles in case 1 the obstacle node is certainly going to make the arc between the two waypoints more time-consuming for the robot to traverse and this could alter the optimal greatly. The additional time taken to avoid the obstacle can be assumed to be equivalent to uniform velocity travel at the avoidance velocity of the robot. This part is in fact going to be traveled under the Obstacle Avoidance algorithm and so may involve velocities lower than ordinary straight line GPS travel. To account for this action of obstacles while still ensuring that they are never traversed, they are assigned the effect of increasing the cost of traveling along a certain path (the distance). Physically this means that the robot has to travel some extra distance for going around the obstacle.

This is extremely important because the distances between the nodes are typically much greater than those at which the robot can detect obstacles and so the robot travels the optimal path the program initially calculates, immaterial of the presence of obstacles. So, to make this closer to the true optimal path the assumption has to be made that the distance between the waypoints increases. This increase will have to be decided based on the position of the obstacle and the specifications of the robot, as to its stopping distance, obstacle avoidance procedure etc.

In case the obstacle is in condition 2, there will not be a great change in the distance calculations. In physical terms the robot will not have to deviate much from the path it is traversing. The effect of obstacles in state 1 is thus to alter the distances between the existing nodes of the system. This change or correction factor for the distances between the waypoints that the obstacles affect is very important as it decides how close the solution provided by this program is to the 'true' optimum. This true optimum can never actually be achieved because the obstacle positions are not known exactly before the competition, and neither the effect of the obstacle on the robot's behavior, in terms of the exact extra distance it will have to traverse to avoid the obstacle.

3.3 Refined model

Objective Function: Minimize $\sum X_{ij} C_{ij}$ (Summed over all i and j, $i \neq j$)
$i, j \in \{1, 2, 3 \ldots\ldots n\}$ (only waypoints)

Subjected to constraints:

$\sum X_{ij} = 1 \quad \forall\ i;$ (summed over j = 1 to n) (8)

$\sum X_{ij} = 1 \quad \forall\ j;$ (summed over i = 1 to n) (9)

$X_{1n} = 0$ (10)

$X_{n1} = 1$ (11)

$\sum X_{ij} \leq m-1 \quad \forall\ i, j;\ i \neq j$ (For all nodes taken m at a time) (12)
$m>2,\ n-m>=1$

X_{ij} are binary variables (can take values 0 or 1 only)

3.4 Solution

Several geometric arrangements of a number of nodes were considered, for setting an imaginary obstacle course for the robot to traverse; and the optimal solution was obtained each time. The model was initially solved without the obstacles, the course consisting of waypoints, an origin and a destination. The above formulation for these example situations was drawn up and modeled and solved to optimality in MS Excel Solver. The constraints were all satisfied and the optimal path given was compared to the manually computed solution. The optimal path was found to have an exact match each time. Out of the alternate optimal paths that can exist which can be seen manually, the program chooses one at random. The solution times were small for the smaller models, but with the larger modules the limits of MS Excel Solver were reached. An AMPL code was written for these cases.

3.5 Example results and analysis

3.5.1 Example 1

The first example situation that was considered is an arrangement with four nodes, three waypoints and the origin node. The geometric arrangement and the network are shown in Figure 1.

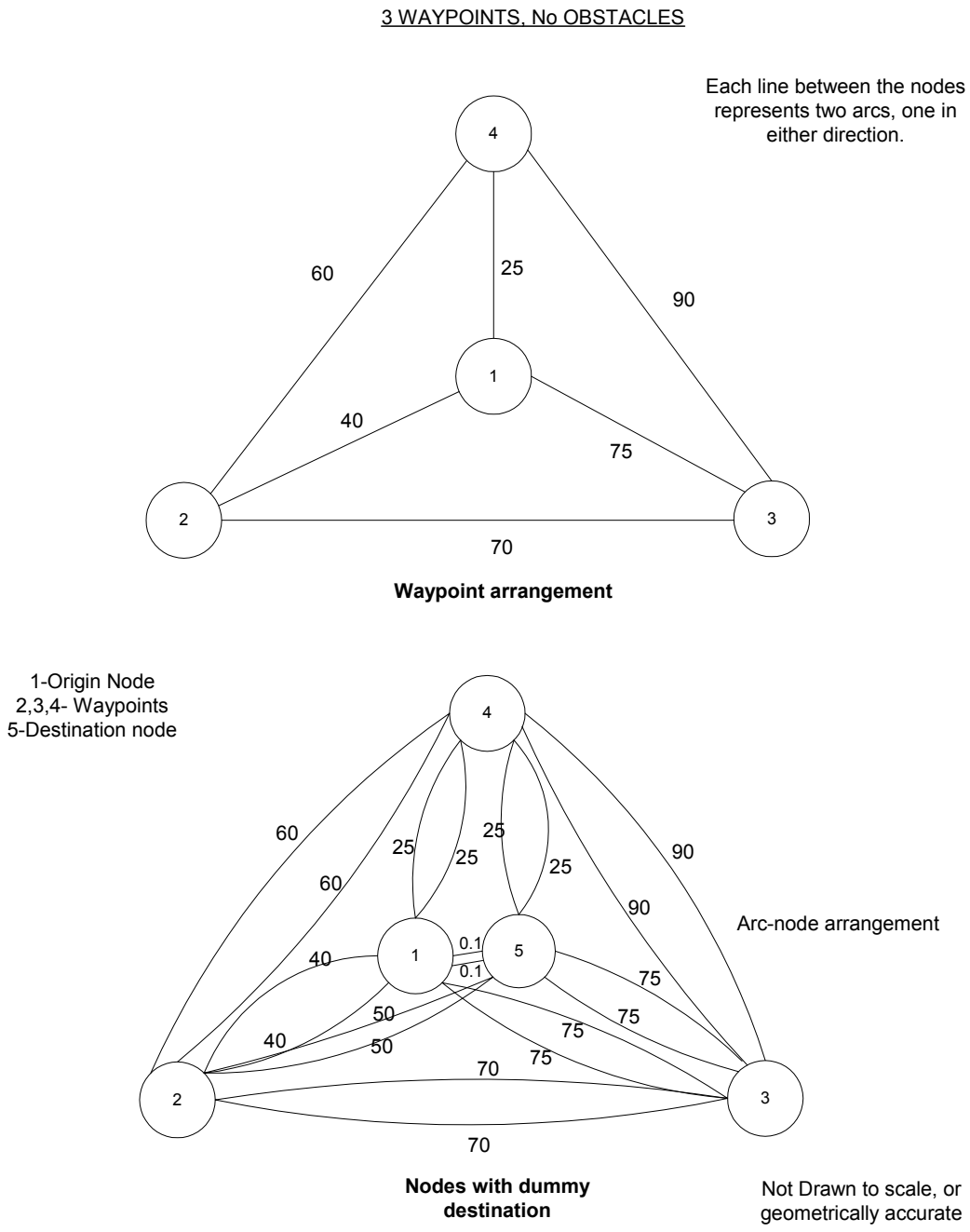

Figure 1. Example 1 Network

The lower part of the figure gives the arc-node network along with the dummy destination node created. The distances from the waypoints are assumed identical for both the origin and destination node, since they can be arbitrarily close mathematically and are identical physically. This problem has 20 variables and 22 constraints.

3.5.1.1 Results

The model for the above situation is obtained as above with nodes 1 to 5. This model was input into MS Excel Solver and the formulation was set up. The elements of the model, like the objective function, the constraints and the decision variables and parameters are all set up as a convenient spreadsheet formulation. Setting the required solution options the solution was obtained as 225.1 units.

The decision variables that take the value 1, indicate that that arc has to be traversed as part of the optimal path. From these, starting at node 1, this shows that the optimal path is 1 to 2 to 3 to 4 to 5 and back to 1. The corresponding distances when added will give the objective function value of 225.1. The limits on each constraint and the value that constraint takes in the optimal solution are also shown. This indicates whether a certain constraint is binding or not. Answer and Sensitivity reports also can be obtained for this solution, which give some indication of the quality and flexibility of a solution obtained.

3.5.1.2 Analysis

The optimal path can be obtained manually by calculating each path and then choosing the best among those. Since this is a small problem this kind of testing is possible. Once it is established that the model is performing as expected, the results for larger problems can be taken at face value. The possible paths are given below. These paths are based on our rules that each node will be traversed only once and that the travel begins at the origin node (1) and ends at the destination node (5).

The paths and the corresponding lengths are shown

1 – 2 – 3 – 4 – 5 – 1	225 units
1 – 2 – 4 – 3 – 5 – 1	265 units
1 – 4 – 2 – 3 – 5 – 1	230 units
1 – 4 – 3 – 2 – 5 – 1	225 units
1 – 3 – 2 – 4 – 5 – 1	230 units
1 – 3 – 4 – 2 – 5 – 1	265 units

It can be seen that the shortest of these paths is either of 1 – 4 – 3 – 2 – 5 – 1 or 1 – 2 – 3 – 4 – 5 – 1. The second of these two paths is the optimal path that has been given as the solution by the Solver program also. The value of the objective, in other words the minimum total distance to be traveled is 225 units. The program gives a value of 225.1, differing by 0.1 from the optimal. This is natural because we have assumed a distance from the destination node to the origin node of 0.1 units, whereas in reality these two points are the same. Thus the program has given the optimal path and the sequence of points to be given to the robot.

3.5.2 Example 2 (with obstacles)

The third example situation that was considered is an arrangement with six nodes; three waypoints, two obstacle nodes and the origin node. The geometric arrangement and the waypoint network are shown below in Figure 2. Here the obstacles are nodes T1 and T2. The effect of causing the node distances to increase, because of ‘curving’ is clearly seen. Distance W2-W3, which was 90 units in Figure 1 is now increased to 100 units, because of the obstacle between them.

For a robot that has to travel the distance between these two nodes, the straight line distance of 90 units between them was what it would have planned on traveling, as an input from the path planning algorithm used above. This same distance is actually now equal to 100 units, simplistically shown in the figure as the curved path between these two nodes, that bypasses the obstacle. This computation actually reflects the fact that the Path planning algorithm in tandem with the GPS has to relinquish control of the robot path to the Obstacle avoidance algorithm. Then the robot has to take a curved path around the obstacle and come back to its original path, when the GPS navigation again takes over. This extra distance and time needed is reflected in that 10 units increase. It is obvious how this factor of increase can affect the optimal path calculated. Figure 2 reflects the fact that when the obstacle nodes T1 and T2 are dropped from the model the distances between the existing waypoints will have to be suitable modified. Similarly is the case with distance O-W1.

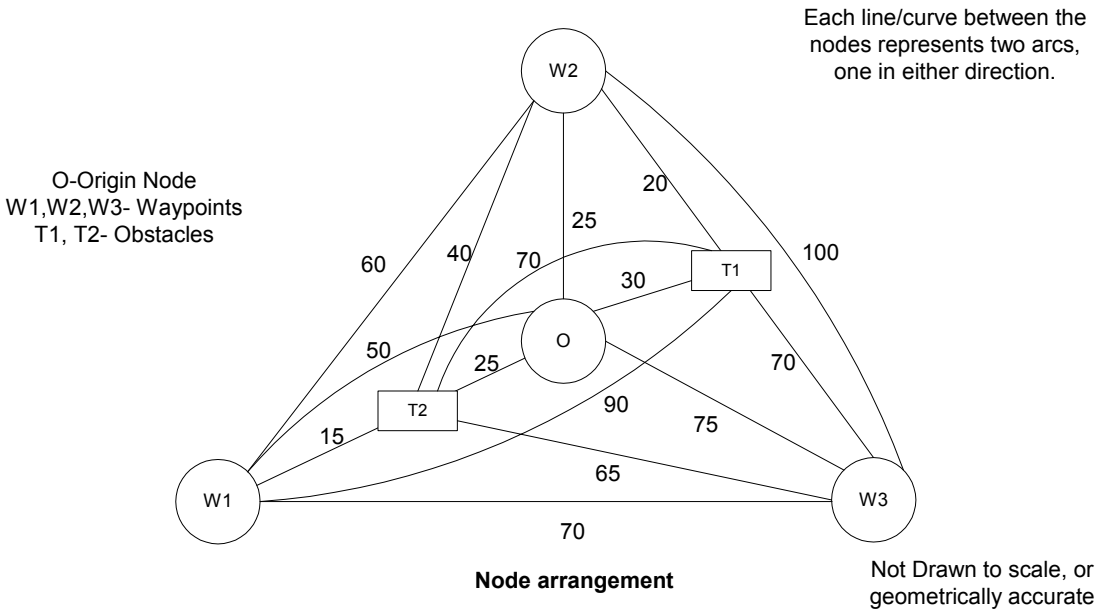

Figure 2. Example 2 Network

3.5.2.1 Results

The is solved using the refined model, thereby dropping obstacle nodes, and accounting for that by increasing the distances between nodes that have obstacles along their path. Thus simplistically, the above model is identical to the one in Example 1, except that the distances have been modified to account for the obstacles. The optimal path obtained in the solution was 230.1 units. The path goes from node O (origin) to W3 to W1 to W2 to destination node to O.

3.5.2.2 Analysis

The presence of obstacles and their effect was modeled in this example. We see how the presence of obstacles has changed the optimal path. The sensitivity to the factor of increase in the distances also becomes evident. There was a 10% and 20% respective increase in the paths considered and this caused a different optimal path.

3.5.3 Example 4 (The given obstacle course)

The obstacle course given in the IGVC website for the description of the navigation challenge was considered as representative of a real world situation for testing the algorithm. This course was the competition course in an earlier competition. It consists of eight waypoints and six obstacles, and in addition the starting and destination point. The coordinates are given in meters measured from the given origin. If the coordinates are given as latitude and longitude, the distances can easily be computed. Using an arbitrary factor the distances between waypoints were changed depending on the presence of the obstacles. This problem was modeled in Excel Solver and in AMPL, a mathematical programming modeling language. A grid figure with the positions of the waypoints, origin and obstacles is given in Figure 3.

3.5.3.1 Results

The model was set-up in the Excel Solver spreadsheet optimizer. The decision variables and the constraints were set-up as in the earlier examples. The model was solved both for the situation where the obstacles would not be considered at all and also the situation where the obstacles would increase the effective distance between the nodes. This step requires the knowledge of the physical specifications, the algorithm speed and obstacle avoidance algorithm of the robot. Based on this model an optimal path of Node 1 (origin, not the labeled Origin) to node 3 to node 8 to node 7 to node 11 to node 12 to node 14 to node 15 to node 2 to node 16 (destination node) and then back to node 1, was obtained as the optimal path for this course. This has 120 decision variables and so solving with the obstacles

included in the model is very slow and inefficient. Solving the problem without the obstacle nodes is an easier problem. The optimal path given above is the solution to this problem. The objective function value, i.e., the shortest path through this obstacle course is calculated as 226.25 meters. In this solution the technique of adding constraints as needed for removing any subtours obtained was used. Thus, if a solution gives a tour that contains subtours, where the robot essentially circulates in a subset of nodes, a tight constraint is introduced to specify that the robot has to pass from one of those nodes out of the subset. Such constraints were added as needed.

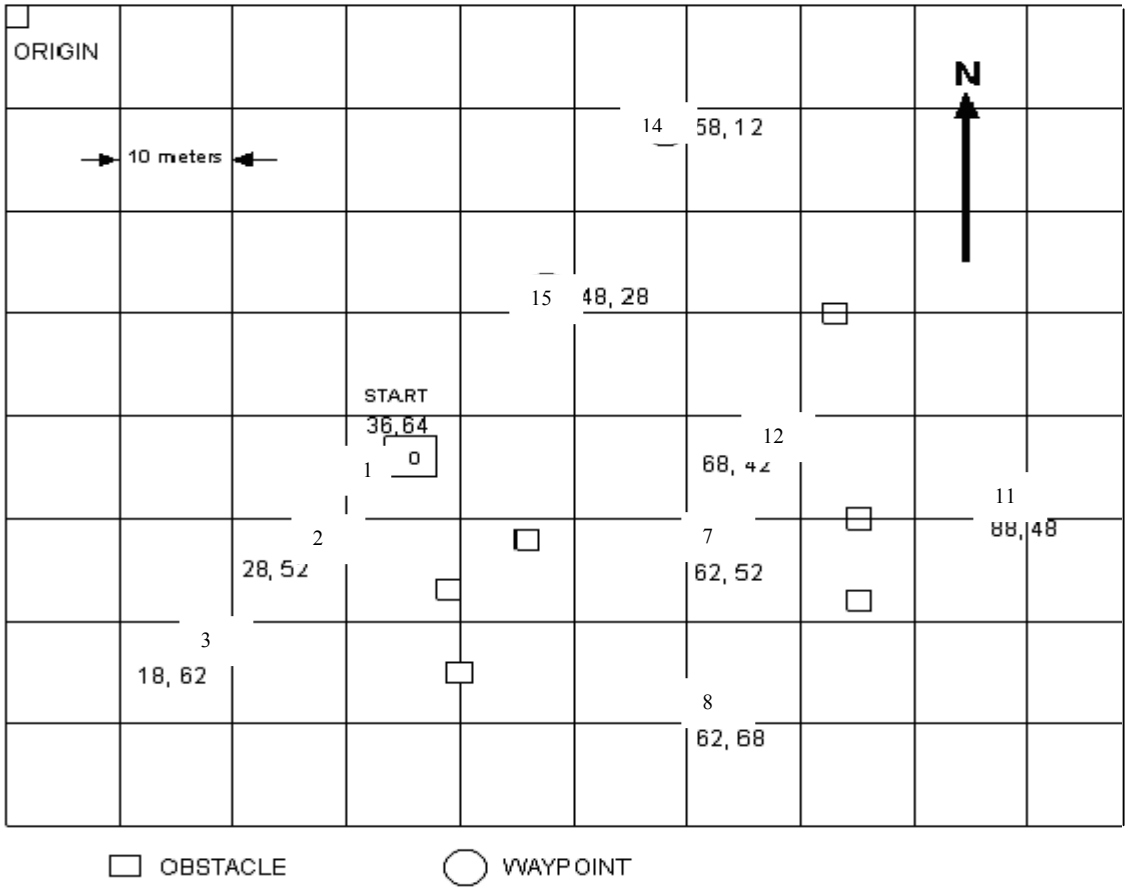

Figure 3. IGVC Network

3.5.3.2 Analysis

The objective function value obtained above is subject to the 'scaling' factor that will be applied to the distances to account for the positions of obstacles. Since that kind of data was not available, the scaling factor was applied as one, in effect as if there were no obstacles in the course. This was done only to prove the correctness and functioning of the program. With better information about the specifications of the robot, in reality, these factors can be calculated very easily and the same model used to get a better estimate of the optimal path. The optimal path achieved in this way gives a sequence of points and these can be fed directly to the navigation algorithm of the robot, for a pre-programmed route through the obstacle course.

In several situations, these competitions have time limitations and so the calculated path has to be traversed in the available time. Needless to say, this is dependent on the speed of the robot. In many situations it will not be possible for the robot, with its given velocity limitations to cover this path in the given time. In that situation the program has to make a choice between what points to keep and which ones to drop. Once that decision is made, the above program can be used for this subset of points and the optimal path calculated. For making this decision some kind of heuristic can be used. One popular one is to drop that point that is on the presently calculated path and has the maximum

distance from its neighbor points on the path. If a certain waypoint is away from a cluster of waypoints, it can be dropped. If dropping this point does not bring the optimal path within the capabilities of the robot, another point from the re-calculated optimal path will have to be dropped. And so on till the subset of nodes can be traversed by the robot in the available time with its speed limitation.

In the above situation the program needs to be modified. This is not really a major modification, as it will only involve changing the constraints and decision variables present in the model and removing some of them. This can be done using some kind of heuristic rules or iteratively. The processing power required for such a program will be immense. But the basis of the decision making process will still be provided by this program above. This can be accomplished using VBA Macros with Excel Solver.

4. FUTURE WORK AND CONCLUSIONS

With more powerful optimizers like AMPL, Matlab etc., the size of the problem that can be solved can be greatly increased. In addition the above-delineated process of iterative application of a heuristic rule of thumb can be coded in these powerful optimizers. Alternatively, and as is being pursued by the author, this process of making a choice of nodes can also be included in the mathematical program. The formulation itself makes a choice of what nodes to consider. The velocity limitations can be modeled as a simple constraint.

Having a powerful processor and better coupling with the existing obstacle avoidance and navigation algorithms will make solutions much more efficient. In addition, this will open the way for real-time calculation of the solutions while the robot is in the course.

4.1 Conclusions

The problem of interest is solved in a convenient fashion, though the scope for future research is huge. Because of the typical applications and context of this problem, the size does not become unwieldy and simple spreadsheet optimizers can be used to get many solutions. These solutions are practically applicable as they give a sequence of points that has to be followed. The program is easy to run and understand so it can be used to obtain results in the field amidst competition. Another way has opened for optimization to be applied in our lives, especially in this application that can be a great learning experience with a lot of fun.

REFERENCES

1. UC Robot Team, *Design report for Bearcat III*, University of Cincinnati Center for Robotics Research, Cincinnati, 2002
2. E.L. Hall, B.C. Hall, *Robotics: A User-Friendly Introduction*, Holt, Rinehart and Winston, New York, 1985
3. K. Kolli, E.L. Hall, "Steering Control System for a Mobile Robot", *Proceedings of SPIE - The International Society for Optical Engineering*. v 3208, pp 162-167, Society of Photo-Optical Instrumentation Engineers, Bellingham, 1997
4. M.L. Fisher, "Optimal solution of vehicle routing problems using minimum K-trees", *Operations Research*, **v 42**.,**No. 4**, pg 626-642, July 1994.
5. G. Laporte, *Classical and modern heuristics for the vehicle routing problem*, Centre for research on Transportation, Montreal, 1998
6. http://www.igvc.org/deploy/
7. http://www.math.princeton.edu/tsp/
8. http://www.nist.gov/dads/HTML/vehicleRouting.html
9. M.S. Bazaaraa, J.J. Jarvis, H.D. Sherali, *Linear Programming and network flows*, Wiley Publishers, New York, 1990
10. H.A. Eiselt, C.L. Sandblom, *Integer Programming and Network models*, Springer-Verlag, Berlin, 2000