Path Optimization of a Wheeled Ground Mobile Robot

Akshay Hiregoudar[*] , Kartik Prakash[*] , Mukund Vishwanathan[*]

Department of Mechanical Engineering, Texas A&M University, College Station, TX, USA

This paper compares different optimization techniques for a path planning problem of a mobile robot in a grid environment with static obstacles. The mobile robot starts from an initial point and traverses through a grid subject to the given conditions, and returns to the start point. We use a gradient based optimization technique to evaluate its feasibility to solve the problem. Subsequently, a Genetic Algorithm is used to verify if it gives a better solution, and the conventional Genetic Algorithm is further extended to implement Pareto Optimality conditions to solve a Multi Objective problem.

Nomenclature

M	Mass vector
V	Volume vector
R	Requirement vector
GA	Genetic Algorithm
M_{mx}	Payload capacity
V_{mx}	Maximum allowable volume

I. Introduction

The path planning aspect of a mobile robot is a topic of active research, due to its wide applications in warehousing, space exploration, autonomous vehicles and manufacturing. A mobile robot must be capable of generating collision free-paths from one point to another, and at the same time, the paths must be optimized subject to the given constraints. In this report, we propose a path planning algorithm for a mobile robot, which receives an order of items from the customer, and the mobile robot plans its path through the warehouse, which is modelled as a grid environment. A majority of the customers who engage in online shopping prefer to get their order delivered within a day, and thats where warehouse automation plays a crucial part. We compare gradient based and heuristic based techniques to arrive at an optimal solution. The mobile robot has a payload capacity and a volume constraint to restrict the number of items it is capable of carrying during each trip. The robot returns to the start point at the end of each trip to unload the items it has picked up during the trip.

II. Problem Formulation

A. Terminologies and Concepts

1. Design Matrix

$$\begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 & x_{10} & x_{11} & x_{12} \\ x_{13} & x_{14} & x_{15} & x_{16} & x_{17} & x_{18} \\ x_{19} & x_{20} & x_{21} & x_{22} & x_{23} & x_{24} \\ x_{25} & x_{26} & x_{27} & x_{28} & x_{29} & x_{30} \end{pmatrix}$$

[*]Graduate Student

The design matrix as shown above is a matrix representation of the design variables x_1 through x_{30} in our optimization problem, which represents quantity of each item. Each row indicates the trip made by the robot, and sum of each column is the requirement vector of an item from that department.

2. Departments

Department refer to the location in the grid environment where the mobile robot can pick up items.For the problem considered in this paper we have assumed a total of six departments with one drop-off/start location. Each column in the design matrix represents items acquired from a particular department. The columns are shown below for clarity numbered A through F (left to right).

$$\begin{pmatrix} x_1 \\ x_7 \\ x_{13} \\ x_{19} \\ x_{25} \end{pmatrix} \begin{pmatrix} x_2 \\ x_8 \\ x_{14} \\ x_{20} \\ x_{26} \end{pmatrix} \begin{pmatrix} x_3 \\ x_9 \\ x_{15} \\ x_{21} \\ x_{27} \end{pmatrix} \begin{pmatrix} x_4 \\ x_{10} \\ x_{16} \\ x_{22} \\ x_{28} \end{pmatrix} \begin{pmatrix} x_5 \\ x_{11} \\ x_{17} \\ x_{23} \\ x_{29} \end{pmatrix} \begin{pmatrix} x_6 \\ x_{12} \\ x_{18} \\ x_{24} \\ x_{30} \end{pmatrix}$$

The first column represents the items acquired over 5 trips from department A, the second column represents the items acquired over 5 trips from department B and so on.

3. Trip Vector

Due to the mass and volume capacity of our robot imposed by its physical design, it cannot carry all the required items in a single run. And hence we have the concept of trips. The aim is to minimize over the number of trips the robot makes to complete the given order. In our Design Matrix, the i$^{\text{th}}$ row represents the items acquired from different departments during the i$^{\text{th}}$ trip. For example $(x_1\ x_2\ x_3\ x_4\ x_5\ x_6)$ represents the first trip, $(x_7\ x_8\ x_9\ x_{10}\ x_{11}\ x_{12})$ represents the second trip and so on.

4. Mass Vector and Payload capacity

Each department contains a specific item. Mass of each unit for respective departments are defined by a Mass Vector $[M_1\ M_2\ M_3\ M_4\ M_5\ M_6]$. There is also a maximum limit on the payload our robot can carry.

5. Volume Vector and Maximum allowable volume

Similarly volume of each unit for respective departments are defined by a Volume Vector $[V_1\ V_2\ V_3\ V_4\ V_5\ V_6]$. There is also a maximum limit on the volume our robot can carry.

6. Prioritisation of Trips

This means that items acquired in first trip are prioritised over the subsequent trip. Its almost analogous to the concept of depreciation. Each subsequent trip is depreciated slightly than the previous trips.

7. SPGA

SPGA stands for Shortest Path Genesis Algorithm. It is an evolutionary optimization algorithm specifically customized for our problem, which returns the shortest path between two points with or without obstacles.

III. Assumptions and Specifications

The following assumptions are taken into consideration while solving this problem.

- The mobile robot is assumed to be a point mass object.
- The robot is holonomic and can move freely in any direction. This means that the robot does not have any constraints on velocity.
- The total number of trips the robot can undertake is restricted to 5.

IV. Warehouse Layout

In this project we have tried to address a very general layout of a warehouse. The warehouse consists of six departments labeled A through F. There is a start point which is also the drop-off location. This means that after each trip, the robot has to return to this location to drop-off the items in its carriage. A schematic is shown in Fig. 1.

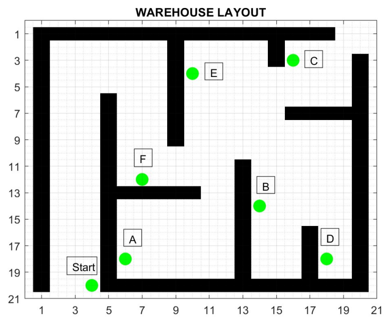


Figure 1. Unstructured Warehouse

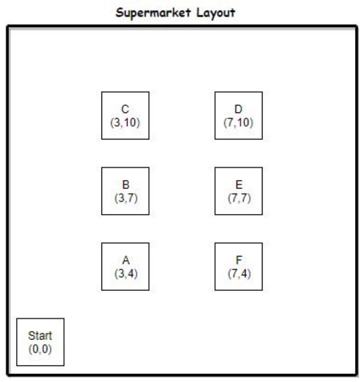


Figure 2. Structured Warehouse

Through this project we have tried to solve a case for unstructured department locations with obstacles. Although its very easy to notice that a general warehouse is very structured and if we take a case without obstacles, the problem can be solved much easily by using a geometric approach to make beneficial simplifications that reduce the run-times of the problem.

If we take the case shown in Fig. 2, which has structured department locations and no obstacles, one can easily notice that there is a lot of symmetry in the problem. And there is an optimal way to find the order in which the paths need to be traversed. Usually finding an optimal order means solving a Traveling Salesman Problem. But due to an inherent symmetry in the problem, one can employ a geometric approach. The geometric approach is centered around finding a closed polygon path. The layout can be divided into two parts namely left (A, B, C) and right (D, E, F). Given the values of the design variables, the problem reduces to finding the extreme y co-ordinates (minimum and maximum) from both the left and right parts. So we have now reduced the problem from a total of six departments to a maximum of just four departments.

V. Objectives and Constraints

In our work, we will be aiming to optimize the following two objectives for the order placed by customer(s) and assist workers in warehouse operations.

- Our first objective is to maximize the number of items per trip. We represent this function as follows:

$$I = 10^8 \sum_{i=1}^{6} x_i + 10^4 \sum_{i=7}^{12} x_i + 10^2 \sum_{i=13}^{18} x_i + 10 \sum_{i=19}^{24} x_i + 5 \sum_{i=25}^{30} x_i \quad (1)$$

  where x_i denotes the i$^{\text{th}}$ design variable as defined in the ***Design Matrix***.

- Our second objective is to minimize the total path length of a mobile robot. Here its important to understand that there are two important concepts at play. The first concept to is to define path length between two points given the co-ordinates of the path. The second and perhaps more difficult concept is to obtain a function to define a path length for a list of departments in the trip vector. It is difficult to define because even though from the design variable values we know that the departments to visit, we still do not have any information about the order they need to be traversed. As explained in the Design of Experiments later, we have chosen the *best-first* search heuristic to define this function. Due to conditional nature of the function its hard to put it into an expression here. For the path length between two departments however, we can easily obtain a closed form formulation as described further. If we denote the path between two points by means of intermediate co-ordinates as $\{(x_{c1}, y_{c1}), (x_{c2}, y_{c2}), (x_{c3}, y_{c3}), ..., (x_{cn}, y_{cn})\}$ then we represent this function as follows:

$$\sum_{i=1}^{n-1} \sqrt{(x_{c(i+1)} - x_{ci})^2 + (y_{c(i+1)} - y_{c(i)})^2} \tag{2}$$

- The constraints are defined below

$$x_i + x_{(i+6)} + x_{(i+12)} + x_{(i+18)} + x_{(i+24)} = R_i, \text{ for i= 1,2,3,4,5,6} \tag{3}$$
$$Mass\ Vector * Trip\ vector \leq M_{mx}, \text{ for all trips} \tag{4}$$
$$Volume\ Vector * Trip\ vector \leq V_{mx}, \text{ for all trips} \tag{5}$$
$$\{x_{ci}, y_{ci}\}\ \epsilon\ \{\text{set of feasible paths}\} \tag{6}$$

VI. Modules

Fig 3 shows the various modules used in our work, and the functionality of each module is explained in some detail.

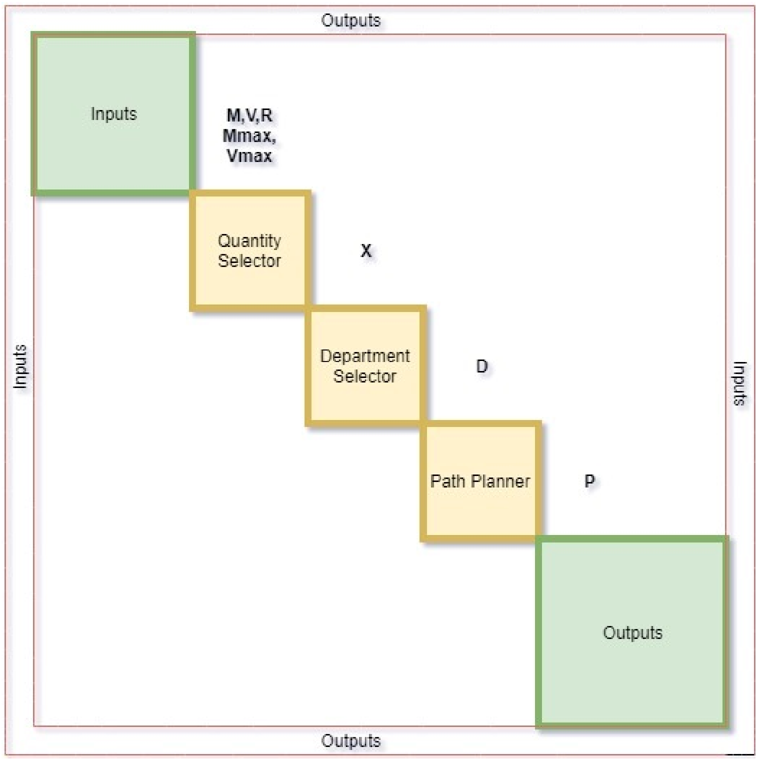


Figure 3. N^2 diagram

A. Quantity Selector

Our main algorithm of choice was standard Genetic Algorithm present in MATLAB. The genetic algorithm generates an initial random values for our design matrix, which forms the basis for evaluating our first objective function which depends on actual values. As it will be explained in Design space exploration later we have two design spaces i.e. Values of the actual variables(Quantity) and sequence of those values (Departments).

B. Department Selector

The department selector module converts all the non-zero values present in the design matrix to a value 1 (which means that the particular department has been selected) and rest are assigned 0 (which means that department was not chosen). These values are then sent to the path planner for finding shortest path between two consecutive departments as described by the trip vector(define the trip vector).

C. Path Planner(SPGA)

The path planner receives the department information from the Department selector module. The task of the path planner here is to obtain the shortest path between two departments. To decrease run times, the path planner was run on every pair of departments in order to form a vector of distance values, which was then directly used in the code. To achieve this we designed our custom evolutionary algorithm (SPGA - Shortest path Genesis Algorithm) which has the following characteristics.

- **Initial Population Generator:** A set of random feasible paths was generated between two points as demonstrated below. These were generated by randomly choosing subsequent points in a narrow window of directions roughly oriented towards the goal point. Four such paths are shown in Fig. 4.

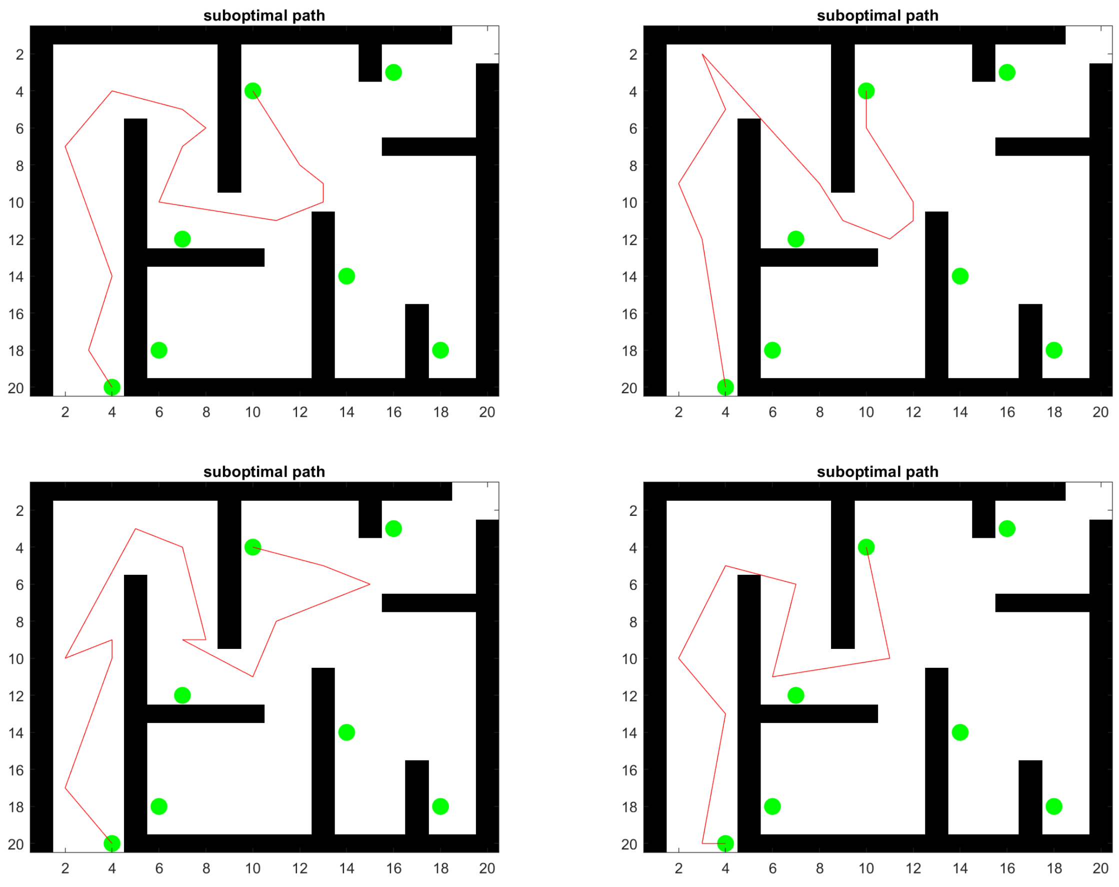
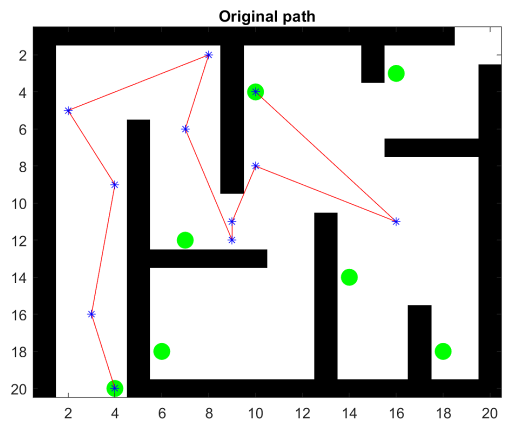


Figure 4. Sub Optimal Paths

- **Densification :** The second operation was to insert intermediary points in paths at an regular interval. Fig.6 shows a path where intermediate points were inserted at an interval of 0.5. This process helps achieve a better Crossover which was realized during execution of the program. Next step is selection.



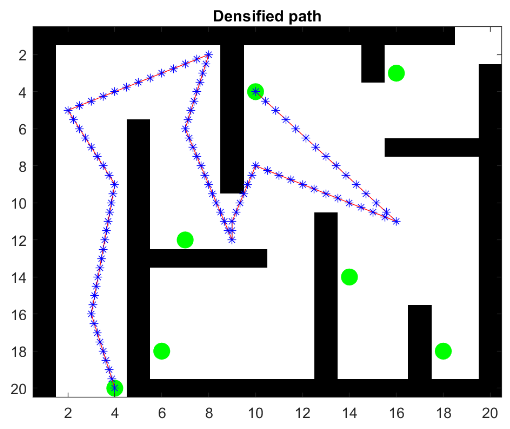
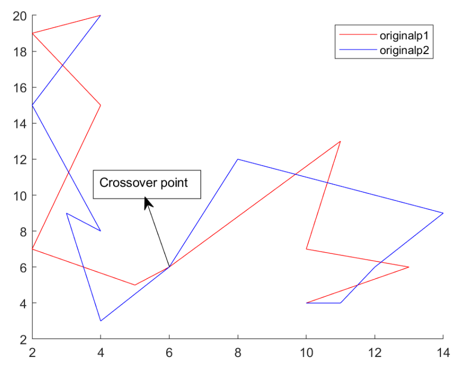


Figure 5. Original Feasible Path

Figure 6. Densified Feasible Path

- **Selection :** Based on the fitness function defined as the length of the path, we evaluated the best possible parents to pass their genes, and then from the pool with a finite probability two parents are selected to reproduce, which brings us to our next step which is Crossover.
- **Crossover :** The crossover step is a two step process, which involves the identification of crossover points, followed by the crossover which involves swapping across a random crossover point. No mutation is carried out for this implementation, since it is very easy to divert to infeasible paths that way. However a clever mutation can be selected where mutated points can be restricted to remain in the set of feasible paths. However for obtaining the shortest path, crossover worked out fine. A sample crossover is shown in Fig. 8.



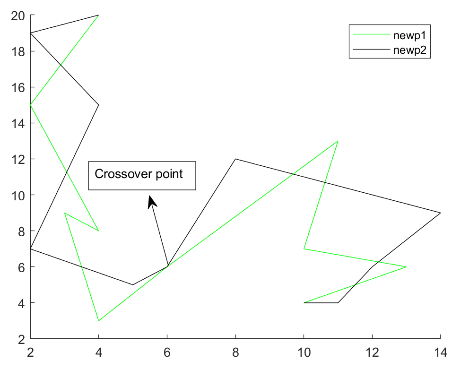


Figure 7. Before Crossover

Figure 8. After Crossover

- **Elitism :** We chose a elite count of 1 which means that only the best of previous generation was allowed to remain in the subsequent generation.
- **Convergence History :** The convergence history for *SPGA* is depicted in Fig. 9.
- **Results and Bench-marking :** The results obtained were compared with those obtained from a standard shortest path finder algorithm A star for bench-marking (Fig. 8). And it was very exciting for us to notice that our custom Genetic Algorithm and the A star gave identical results, with a minor caveat of extended run times. The main motive behind using a GA is that in future it can also incorporate fitness functions like smoothness and safety with obstacles. Till now, we have not been

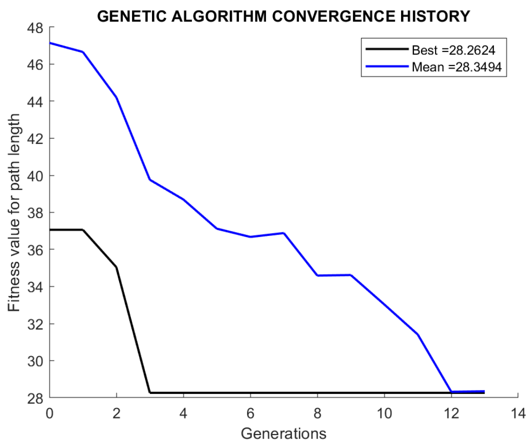


Figure 9. Convergence History

able to achieve the performance given by standard smoothing techniques and thus they have not been incorporated in our project. But there is a potential to do so in the near future.

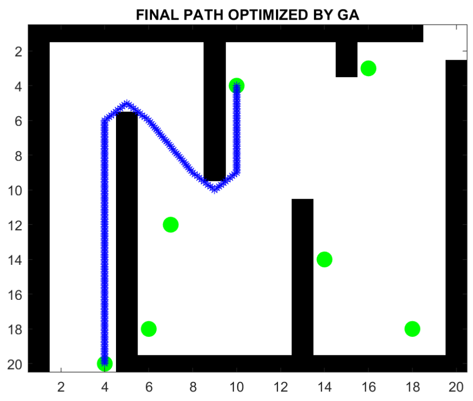


Figure 10. Path optimized by SPGA

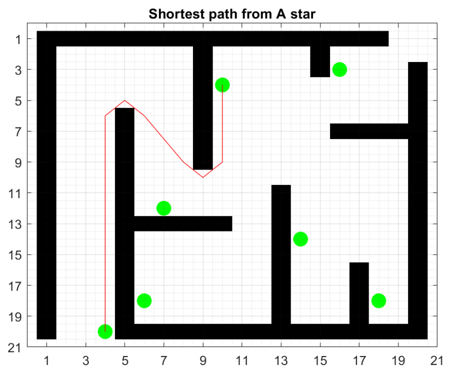


Figure 11. Path optimized by A star

The parameters used for the custom designed evolutionary algorithm *SPGA* are summarised in Table 1.

Table 1 : Algorithm Parameters

S.No	Property	Value	Remarks
1	Crossover fraction	1	100 % probability of crossover
2	Population Size	11	Number of paths in each generation
3	Max Generations	500	Maximum allowable generations
4	Function Tolerance	1e-6	Halting criteria for convergence
5	MaxStallGenerations	10	Maximum generations with the same best fitness in tandem
6	Densification	0.1	Insertion of intermediary points at an interval of 0.1 units
7	Elite Count	1	Number of best paths from previous generation to be allowed in the subsequent generations

VII. Design of Experiments

Our original problem consists of six pickup stations but for simplicity and ease of exploration, we have considered a case with three pickup stations viz.; A, B, and C. The requirement vector R is chosen as [5, 10, 15] and the maximum number of trips is restricted to three. Mass vector M is given by [3, 2, 1] and the payload capacity is limited to 20 units.

There are two design spaces for our problem viz,; Value and Sequence. A full factorial search was performed to visualize the design spaces. The design space of value and sequence is represented using Fig. 12 and Fig. 13, where the magenta and blue points represent the feasible and infeasible points respectively. The graph is obtained using the equality and inequality constraints of the problem. Fig. 12 is a representation of the inequality constraint (mass constraint) and Fig. 13 is a representation of the equality constraints. The actual design space of value is an intersection of Fig. 12 and Fig. 13, which can be represented only in a 5 dimensional space.

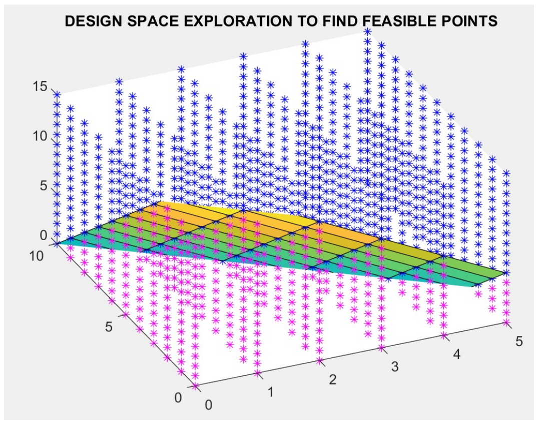


Figure 12. Value Design Space

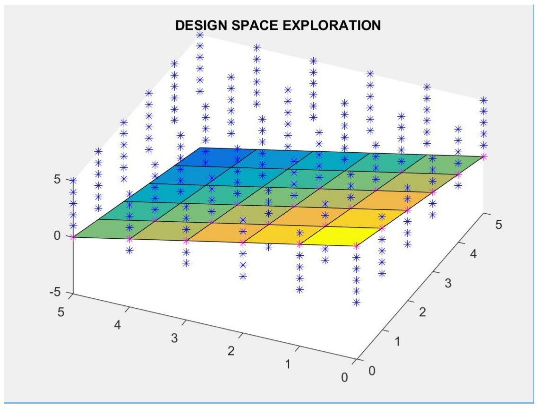


Figure 13. Sequence Design Space

The design space of sequence represents the order in which the items need to be picked up by the robot. This problem involves using the traditional approach of solving a travelling salesman problem but this problem is nested within another optimizer and thus the traditional approach becomes computationally expensive and time consuming. Hence, in order to solve the problem we chose the Best First Search method which for our problem will always give a near-optimal solution if not the optimal solution. This method aids the robot to detect the distance between the pickup stations and itself and help decide the closest one. There was a trade-off between the optimality and time complexity. Since, we want the robot to function real-time, we prioritised lesser run time over an optimal solution.

VIII. Algorithm Selection

Various optimization techniques including gradient based, heuristic approach and integer programming were explored to find an optimal solution for the problem. All the three algorithms provided a similar result for the single objective problem while heuristic approach and integer programming provided a slightly better result. In case of our second objective which is non-linear in nature, both gradient based and heuristic approach were explored and the heuristic approach provided a much better result. The parameters used for tuning the algorithms are as follows

- Gradient Based optimization was implemented using MATLABs fmincon function. With the default parameters, the algorithm gave an error of unsatisfied constraints and hence a tuning was necessary in order to achieve an optimal solution. The TolFun was set to 0.0001, MaxIter was set to 100,000 and MaxFunEval was set to 100,000 for tuning.

- Heuristic Approach was implemented using the Genetic Algorithm (GA) function in MATLAB. This produced a slightly better result as compared to fmincon. This algorithm did not give us an optimal solution with the default parameter values and the constraints also the constraints were not satisfied. An optimal solution was achieved by tuning certain parameters. The maxstallgenerations was set to 200, functiontolerance was set to 1e-10 and maxgenerations was set to 500 for tuning.

- Integer Programming was implemented using the intlinprog function in MATLAB. This produced the same result as that of the Genetic Algorithm.

IX. Post Optimality Analysis

On calculating the Hessian, we get the matrix to be a 30x30 matrix, with all the diagonal elements of the matrix greater than zero, which implies that the Hessian is positive definite. The eigen values of the Hessian were close to 1, which implies that the the solution obtained has been scaled.

A. Sensitivity Analysis

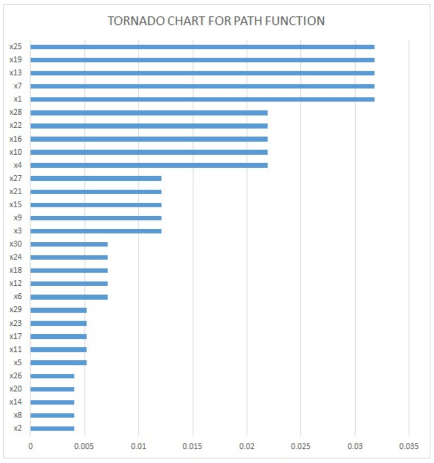


Figure 14. Tornado Chart for Path Function

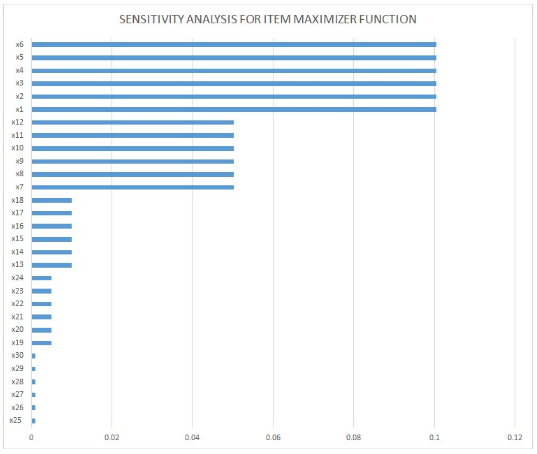


Figure 15. Tornado Chart for Item Maximizer Function

From the tornado charts shown in Fig. 14 and Fig. 15, elements of a column have equal sensitivity since they represent the same departments according to our problem formulation. The elements of each row which represent the mass functions also have same sensitivity since our function prioritizes each variable in a row equally.

B. Scaling

There are no continuous design variables for the problem defined and hence the constraints were scaled to analyze their effect on convergence and convergence rates. Ten different factors were used to study the effect of scaling of constraints using both GA and gradient based fmincon algorithms, and Fig. 16 shows a comparison of the scaling factors between GA and gradient based algorithms. In all the runs, GA performed

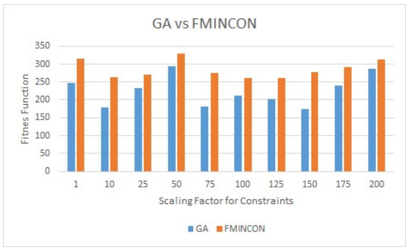


Figure 16. Comparison of scaling factors

better than fmincon. For the scaling factor of 150, GA gave the best value of fitness function whereas in case of fmincon, the best result was obtained for the scaling factor of 100. It does not follow a specific pattern.

X. Multi Objective Analysis

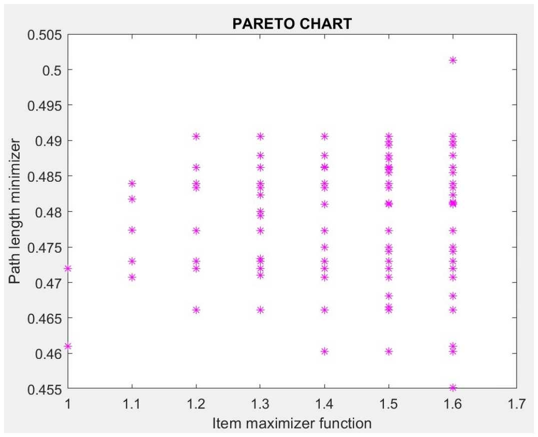


Figure 17. Pareto Front of the Multi Objective Analysis

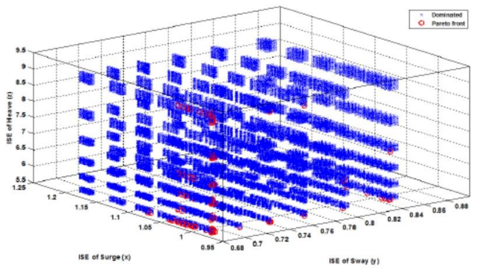


Figure 18. Sample pareto front of a Remotely Operated Vehicle

A Pareto front was plotted, with the objective function to maximize the total number of items during each trip on X-axis, and the objective function to minimize the path length on the Y-axis. The values on the axis are further scaled to show the spread of the plot and should not be mistaken for the actual fitness values. The plot shown in Fig. 17 shows a mapping from variable space to the objective space , and it appears discrete because the algorithm works in a way that the robot plans its path based on the departments it has to visit. Based on whether the design variables take a non zero value or not, the path length function takes certain discrete values.A literature survey was conducted to validate Fig. 17, and verified that with a Pareto front obtained from the multi objective optimization of a remotely operated vehicle[5], as shown in Fig. 18.

XI. Conclusions and Future Work

A feasible solution was obtained from the analysis techniques mentioned in this paper, but it cannot be said if the obtained solution is the global optimum. It can also be said that GA is better suited to solve our problem, since the design variables are entirely discrete. The problem can be easily expanded to optimize objectives such as path smoothness and safe clearance distance, to mimic real life scenarios better. In the future, the operating space of the robot can be expanded further, to increase the number of departments and mimic a real life warehouse scenario. On these lines, the algorithm can be modified to include dynamic obstacles, and more number of mobile robots so that these agents are capable of interacting with each other while moving in the environment. To test the robustness of the model developed, the algorithm can be implemented on a mobile robot platform to validate the results.

References

[1]Oscar Castillo, Leonardo Trujillo, Patricia Melin (2006), Multiple objective genetic algorithms for path-planning optimization in autonomous mobile robots, Springer-Verlag 2006

[2]Bashra K. Oleiwi, Hubert Roth, Bahaa I. Kazem (2014) Modified genetic algorithm based on A* algorithm of multi objective optimization for path planning, Journal of Automation and Control Engineering Vol.2, No.4, December 2014; 357-362

[3]Alexander Lavin (2014) A pareto front-based multiobjective path planning algorithm

[4]Masoud Fetanant, Sajjad Haghzad, Saeed Bagheri Shouraki (2015) Optimization of dynamic mobile robot path planning based on evolutionary methods, 2015 AI Robotics (IRANOPEN), Qazvin, 2015; 1-7

[5]M.F.Nor Shah, S.S.Abdullah, A.Faruq (2011), Multi-objective optimization of ROV control system using surrogate modeling, 2011 IEEE International Conference on Control System, Computing and Engineering; 138-143