AGV routing using Dijkstra's Algorithm – A review

[1,]Er. Atique Shaikh , [2,]Prof. Atul D. Dhale
[1,] *(Department of Automobile Engg , Mumbai University, MHSSCOE , Mumbai - 400008.*
[2,] *(Associate Professor, Department of Mechanical Engg., SSJCOE, Dombivali, Mumbai University.*
[1,] atsshaksss@gmail.com, [2,] atuldhale32000@yahoo.com

Abstract

This paper present a review on AGVs used in warehouse and their routing decision using Dijkstra's Algorithm. Studies illustrate that Shortest Path problems are among the most studied network flow optimization problems, with interesting applications in a range of fields. In the current scenario of extremely fast technology development, automated logistic systems, used by industries, warehouses, seaports, and container terminals are adopting AGVs (Automated Guide Vehicles) as a flexible and scalable alternative to optimize transport tasks. It is important to emphasize that its productivity is highly dependent on the adopted routing scheme. Consequently, it is essential to use efficient routes schemes. Past research have shown that Dijkstra Algorithm is considered as the best algorithm in shortest path planning algorithm.

Keywords: Automatic Guided vehicle (AGV), Dijkstra's algorithm, vehicle routing.

1. Introduction

Automated guided vehicle systems (AGVS) are commonly used for transporting material within a manufacturing, warehousing, or distribution system. These systems provide for asynchronous movement of material through the system and are used in a wide variety of applications. They offer many advantages relative to other types of material handling systems, including reliable, automatic operation, flexibility to changes in the material handling requirements, improved positioning accuracy, reduced handling damage, easily expandable layout and system capacity, and automated interfaces with other systems.

In order to decrease costs of logistics and distribution of goods, it is quite common to find in developed countries mechatronic systems performing several tasks in harbor, warehouses, storages and products distribution center. Therefore, research in this topic is considered strategic to ensure a greater insertion of the individual countries in the international trade scenario. In this application, the vehicle routing decision is one of the main issues to be solved. It is important to emphasize that its productivity is highly dependent on the adopted routing scheme. Consequently, it is essential to use efficient routes schemes.

As one of the enabling technologies, scheduling of automated guided vehicles (AGVs) has attracted considerable attention and many AGV scheduling algorithms have been proposed. Recently, more container terminals have started utilising automated transporters like AGVs. Therefore the research on the scheduling of AGVs has become more important.

1. Dijkstra's Algorithm

Dijkstra's algorithm, conceived by Dutch computer scientist Edsger Dijkstra in 1956 and published in 1959, is a graph search algorithm that solves the single-source shortest path problem for a graph with nonnegative edge path costs, producing a shortest path tree. This algorithm is often used in routing and as a subroutine in other graph algorithms.

For a given source vertex (node) in the graph, the algorithm finds the path with lowest cost (i.e. the shortest path) between that vertex and every other vertex. It can also be used for finding costs of shortest paths from a single vertex to a single destination vertex by stopping the algorithm once the shortest path to the destination vertex has been determined. For example, if the vertices of the graph represent cities and edge path costs represent driving distances between pairs of cities connected

by a direct road, Dijkstra's algorithm can be used to find the shortest route between one city and all other cities. As a result, the shortest path first is widely used in network routing protocols, most notably IS-IS and OSPF (Open Shortest Path First).

Steps of Dijkstra's Algorithm

Step 1 Label the start vertex as 0.
Step 2 Box this number (permanent label).
Step 3 Label each vertex that is connected to the start vertex with its distance (temporary label).
Step 4 Box the smallest number.
Step 5 From this vertex, consider the distance to each connected vertex.
Step 6 If a distance is less than a distance already at this vertex, cross out this distance and write in the new distance. If there was no distance at the vertex, write down the new distance.
Step 7 Repeat from step 4 until the destination vertex is boxed.

2. Literature review

2.1 Fast Shortest Path Algorithms for Large Road Networks, Faramroze Engineer[1] Says :

These studies illustrate that Shortest Path problems are among the most studied network flow optimization problems, with interesting applications in a range of fields. Conventional techniques for solving shortest paths within large networks cannot be used as they are either too slow or require huge amounts of storage. In his project he has tried to reduce the runtime of conventional techniques by exploiting the physical structure of the road network and using network pre-processing techniques. Due to the nature of routing applications, we need flexible and efficient shortest path procedures, both from a processing time point of view and also in terms of the memory requirements. Unfortunately prior research does not provide a clear direction for choosing an algorithm when one faces the problem of computing shortest paths on real road networks. Past research in testing different shortest path algorithms suggests that Dijkstra's implementation with double buckets is the best algorithm for networks with nonnegative arc lengths [1,2].

One possible approach to solving shortest path problems would be to pre-calculate and store the shortest path from every node to every possible other node, which would allow us to answer a shortest path query in constant time.
In his project he implemented 4 search algorithms:
1. Dijkstra's Labelling algorithm (termed 'Dijkstra' in our results)
2. Dijkstra's Bi-directional algorithm (termed 'Symmetric' in our results)
3. A* algorithm (termed 'A*' in our results)
4. Dijkstra's Bi-directional algorithm with radius restriction (termed 'Radius' in our results)
From the experimental results plotted in Figure 1, he concluded that the Symmetric Dijkstra algorithm is almost twice as fast as the Dijkstra algorithm and the A* algorithm is almost three times faster than the Dijkstra Algorithm. However the most impressive reduction in time is through the Symmetric Dijkstra algorithm used in conjunction with radius restriction which is almost 50 times faster on average.

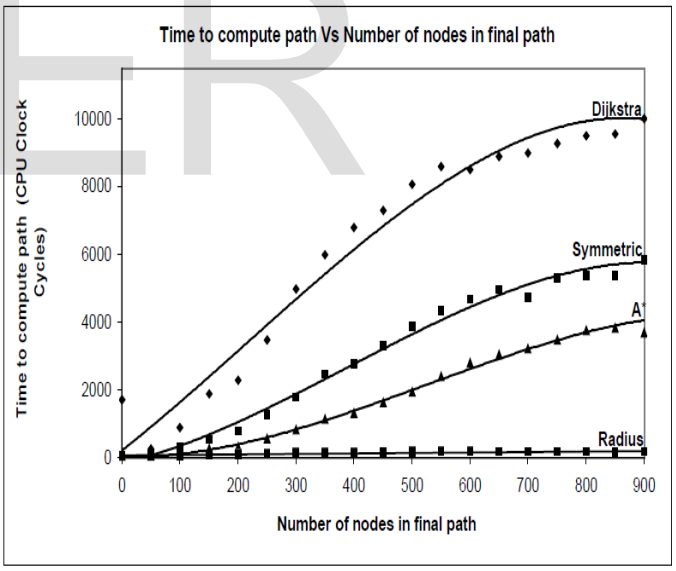

Fig no. 1: Experimental result

He concluded that by exploiting the physical structure of road networks, the A* algorithm is able to bias its search towards a goal and reduce the search space. By using the concept of radii
as a measure of importance of nodes, he was able to incorporate pre-processing within our shortest path algorithm to further restrict the search space. This dramatically reduces the search complexity in terms

of the run time performance while still maintaining an acceptable level of inaccuracy.

2.2. Comparative Analysis of Algorithms for Single Source Shortest Path Problem, Shweta Srivastava says that:

The single source shortest path problem can be defined as: given a weighted graph (that is, a set V of vertices, a set E of edges, and a real-valued weight function f: $E \rightarrow \mathbf{R}$), and one element s of V (i.e. a distinguished source vertex), we have to find a path P from s to a v of V so that is minimal among all paths connecting s to v .

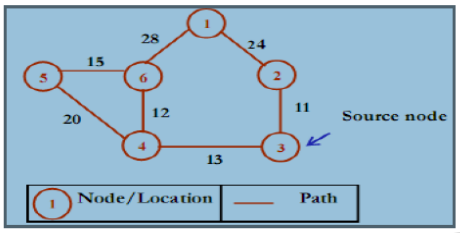

Fig no. 2: An Undirected Graph of 6 nodes and 7 edges

She proposed that the performance of an algorithm for single source shortest path problem depends on the 3 attributes:
(1) Preprocessing time: time required to construct a search structure suitable for search.
(2) Space: storage used for constructing and representing the search structure.
(3) Search Time: time required to find shortest path from a query source s, using the search structure.

Several algorithms have been proposed for this problem which is based on different design paradigms, improved data structure, parameterization and input restrictions.

Different algorithm were proposed for this purpose

1. Thorup's algorithm in 2000
2. Augmented Shortest path algorithm in 2000 for improvement over
3. Arc flag method in 2006
4. a heuristic genetic algorithm in 2007 to achieve high performance
5. Adjacent node algorithm in 2009
6. In 2009, faster algorithms have been proposed.

All the above algorithm were implemented and a comparision was done with the Dijkstra Algorithm.

USEFULNESS OF ALGORITHMS IN VARIOUS APPLICATIONS

1. Thorup's algorithm is much slower than the algorithms with the heaps as for the whole execution time is compared. It is very slow for SSSP due to the time of the construction of the data structures. Due to need of huge amount of memory and the complicated, large programs, Thorup's algorithm is not useful in practice today.
2. The faster algorithm works well for the graphs having smaller number of distinct edge lengths than the density of the graph.
3. The Heuristic Genetic algorithm proved to be suitable for the network of different size and topology. HGA took reasonable CPU time to reach the exact solution and didn't variate much with increased input size.
4. The Augmented Shortest path algorithm is suitable for the graphs with less value of f. According to author it is suitable for the road networks, electronic circuit designs etc.
5. The Adjacent Node algorithm in is efficient for the huge data and takes less space. So it is suitable for the traffic analysis type of applications.
6. The algorithm based on partitioning of graph (Arc flag method) although performed better than Dijkstra's algorithm for some cases but for large networks its performance is degraded than that of Dijkstra's.

In this paper she tried to be explained- first, what are the different algorithms for the SSSP problem. Second, how do they perform in comparison of the Dijkstra's algorithm. Third, which algorithm is suitable for a particular application or situation.

2.3. AUTOMATIC ROUTING OF FORKLIFT ROBOTS IN WAREHOUSE APPLICATIONS, Kelen C. Teixeira Vivaldini, Marcelo Becker, Glauco A. P. Caurin Says That:

Forklift robots are frequently applied in automated logistics systems to optimize the transportation

tasks and, consequently, to reduce costs. Nowadays, in a scenario of extremely fast technological development and constant search for costs minimization, the automation of logistic process is essential to improve the productivity and reduce costs. In this application, the vehicle routing decision is one of the main issues to be solved. It is important to emphasize that its productivity is highly dependent on the adopted routing scheme. This paper proposes an algorithm that produces optimal routes for AGVs (Automated Guided Vehicles) working inside warehouse as forklift robots. In the routing algorithm each AGV executes the task starting in an initial position and orientation and moving to a pre-established position and orientation, generating a minimum path. This path is a continuous sequence of positions and orientations of the AGVs. The algorithm is based on Dijkstra's shortest-path method.

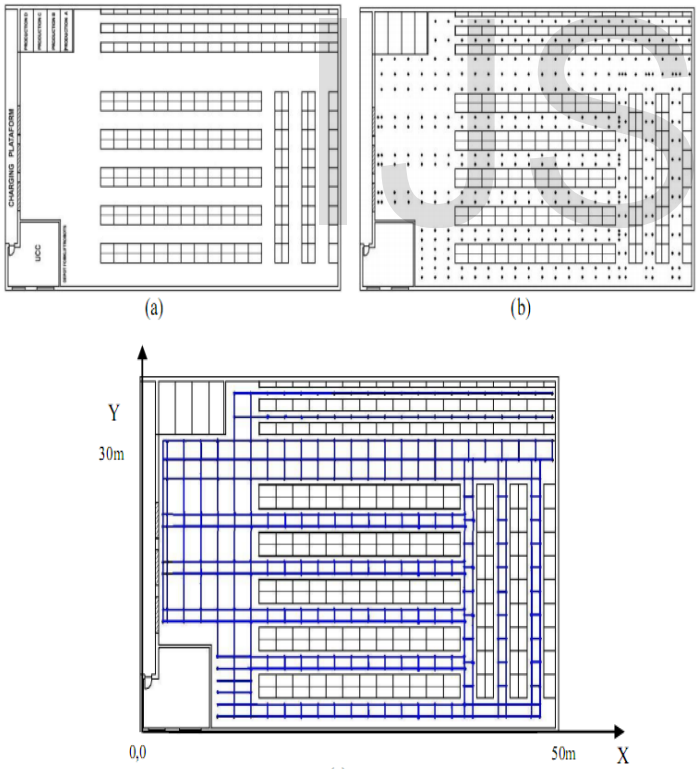

Fig no. 3:. The mapping model considered in the simulations: (a) the warehouse 2D model; (b) the topologic map with the nodes, and (c) final graph used in the simulations.

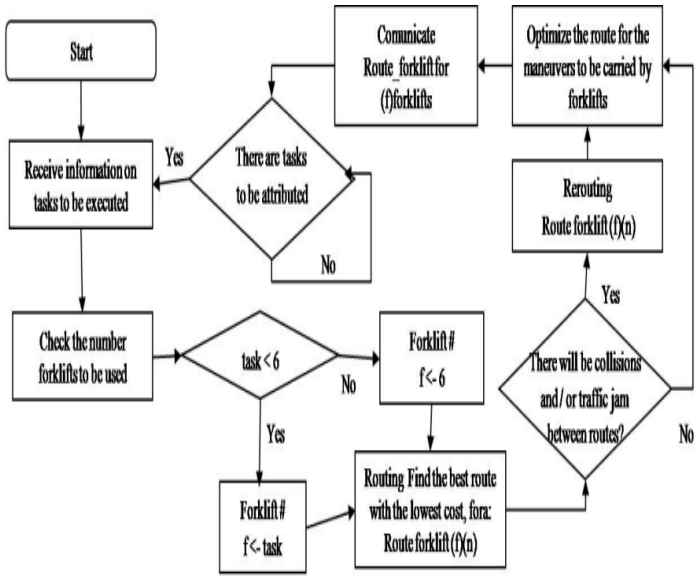

Fig no 4: proposed Algorithm

To verify the algorithm efficiency, we performed several simulations inside virtual warehouse. Therefore, in the proposed scenario, each task related route constitutes a set of 5 sub-routes automatically generated by the algorithm. We tested the three stages of routing, all of them leaving the depot and coming back to it after finishing the tasks. In order to measure the improvement of the computed routes in the stages, we compare the total time spent for tasks, and found the total duration of each task (Tables 1, 2, and 3).

Forklift Robots	Depot	Task #1		Task #2		Depot
		Origin	Destination	Origin	Destination	
1	0	88	208	440	612	760
2	0	148	344	464	604	816
3	0	164	320	432	540	736
4	0	120	316	424	532	656
5	0	80	252	392	540	656
6	0	88	200	328	436	544

Table no 1: Information on the time of route execution - Distance

Forklift robots	Depot	Task #1		Task #2	
		Origin	Destination	Origin	Destination
1	0	88	208	440	612
2	0	148	344	464	604
3	0	164	320	432	540
4	0	136	332	408	524
5	0	80	252	392	540
6	0	88	232	352	460

Table no 2: Information on the time of route execution – Conflict free

Forklift robots	Depot	Task #1		Task #2		Depot
		Origin	Destination	Origin	Destination	
1	0	88	208	440	612	760
2	0	148	280	400	540	762
3	0	164	320	432	540	736
4	0	136	332	408	524	648
5	0	80	252	392	540	656
6	0	88	232	352	460	560

Table no 3: Information on the time of route execution – Optimization maneuvers

The results show that in Fig. 5-a Forklift # 6 had two routing problems: a traffic jam with the Forklift # 1 (detail represented by a green arrow) and a deadlock with the Forklift # 4 (detail represented by a red arrow). In addition to this, in the conflict-free stage (Fig. 5-b), the time cost can be higher than the distance x cost routing (Fig 5-a), because it deals with the traffic jams and dead-locks. Finally, in the stage of maneuvers optimization (Fig. 5-c) the best results for calculated routes were obtained, demonstrating the method efficiency.

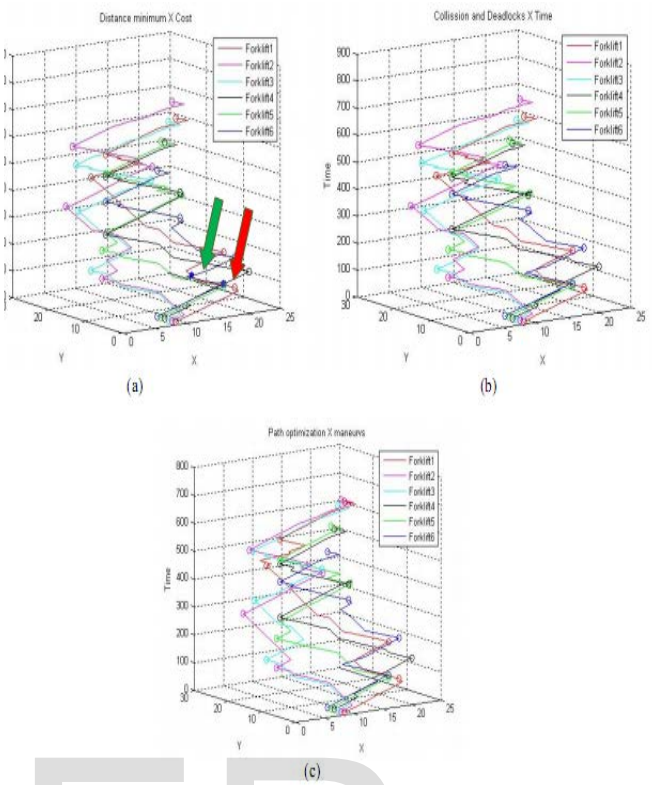

Fig no 5: In (a) the minimum route calculated for the desired tasks; in (b) routes without deadlocks and traffic jams after the rerouting; and in (c) the path was optimized, reducing the total maneuver quantity.

2.4. A self-organizing neural network approach for the single AGV routing problem, Mustafa Soylu, Nur E. Ozdemirel, Sinan Kayaligil says:

The objective is to find the shortest tour for a single, free-ranging AGV that has to carry out multiple pick and deliver (P&D) requests. An artificial neural network algorithm based on Kohonen's self-organizing feature maps is developed to solve the problem, and several improvements on the basic features of self-organizing maps are proposed. Performance of the algorithm is tested under various parameter settings for different P&D request patterns and problem sizes, and compared with the optimal solution and the nearest neighbor rule. Promising results are obtained in terms of solution quality and computation time.

In this section, we try to explore the potential of our approach in solving the routing problem for a single AGV transporting job in various P&D request patterns. In the first stage of experimentation, the

effects of several parameters on the performance of the algorithm are investigated in an experimental design setting, and the treatment combinations that yield relatively better performance are determined for each request pattern.

We experimented with five different request patterns. These are as follows.

1. One-to-many (OTM): P&D requests have the same origin but different destinations.
2. Many-to-one (MTO): P&D requests have different origins but the same destination.
3. Cellular (CEL): P&D requests are created within non-overlapping cells and do not intersect.
4. Mixed-cellular (MIX): P&D requests are created within partially overlapping cells.
5. Fully random (RAN): P&D requests are created randomly with no restrictions.

Problem size	Layout pattern	Neural network algorithm				Nearest neighbor	
		Average final	Best final	Average incumbent	Best incumbent	Average	Best
20	OTM	8.15	2.86	2.06	0.44	25.34	16.84
	MTO	7.91	2.60	2.04	0.36	15.96	9.36
	CEL	16.38	7.14	6.49	2.08	26.30	14.86
	MIX	16.16	7.66	8.65	2.66	29.96	16.41
	RAN	21.34	10.43	13.13	6.37	34.97	23.97
40	OTM	8.05	3.91	3.70	1.49	22.26	16.76
	MTO	7.95	4.12	3.78	1.66	12.15	9.74
	CEL	20.63	8.85	9.82	3.91	48.97	18.03
	MIX	21.55	12.31	13.65	8.53	28.61	17.01
	RAN	28.60	19.03	22.83	14.90	36.94	23.97

Table no 4: responses of the neural network algorithm and nearest neighbor rule(in average % deviation from the optimal solution for fifty problems)

The best solution quality is obtained for OTM and MTO. For these patterns, BI and AF are on the average 0.4% (1.5%) and 8% (8%) larger than the optimum for 20 (40) job problems. The deterioration rate is highest for RAN because the hopping behavior of the algorithm is strongest in this request pattern. A general result of these comparisons is that our algorithm provides good solutions within reasonable computation time for certain request patterns.

3. Conclusion

Based on the various paper studied, we can conclude that in the fast developing environment, AGVs are coming as very useful for transportation purpose as they help to minimize cost and gives reliability and flexibility. The Main issue to deal with AGV is their Routing decision. Shortest path planning is one of the important factors in scheduling, moving, transporting, etc. Various factors considered for Shortest path planning includes avoiding obstacles, minimum path finding, multi objectives constraining, safe distance travelling, etc. Dijkstras's Algorithm can comes very useful as it is the Shortest path planning algorithm and can find variety of application. By finding the Shortest path we can help to minimize Transportation cost and time to a great extent in warehouses, airport and storage terminals.

4. References

[1] Faramroze Engineer, "Fast Shortest Path Algorithms for Large Road Networks", Proceedings of 36th annual ORSNZ conference, 2001

[2] Mrs. Shweta Srivastava, "Comparative Analysis of Algorithms for Single Source shortest Path Problem", International Journal of Computer Science and Security (IJCSS), Volume (6) : Issue (4) : 2012

[3] Kelen C. Teixeira Vivaldini, Marcelo Becker, Glauco A. P. Caurin, "AUTOMATIC ROUTING OF FORKLIFT ROBOTS IN WAREHOUSE APPLICATIONS " ABCM Symposium Series in Mechatronics - Vol. 4 - pp.335-344(2012)

[4] Mustafa Soylu, Nur E., Ozdemirel *, Sinan Kayaligil, "A self-organizing neural network approach for the single AGV routing problem", European Journal of Operational Research 121 (2000) 124±137

[5] Peters, Brett A., Jeffrey S. Smith, and S. Venkatesh. "A control classification of automated guided vehicle systems." *International Journal of Industrial Engineering* 3 (1996): 29-39.

[6] http://en.wikipedia.org/wiki/Dijkstra's_algorithm