Engineering Innovations
ISSN: 2813-1002, Vol. 6, pp 31-36
doi:10.4028/p-t122xr
© 2023 The Author(s). Published by Trans Tech Publications Ltd, Switzerland.

Submitted: 2023-03-01
Accepted: 2023-03-12
Online: 2023-06-21

Low Latency Extended Dijkstra Algorithm with Multiple Linear Regression for Optimal Path Planning of Multiple AGVs Network

Leong Wen Chek[1,a,*]

MIMOS Berhad, Technology Park Malaysia, Bukit Jalil 57000, Kuala Lumpur, Malaysia[1]

[a]wenchek.leong@mimos.my

Keywords: Dijkstra algorithm, Linear Regression, AGV network, Collision, Deadlock

Abstract. Dijkstra algorithms are typically used to find the shortest path from a source node to a destination node. It is widely used in various applications due to its reliability and less complexity. This paper presents the extended Dijkstra Algorithm with lower latency and consumes less computing memory intended for implementation in many AGVs networks for effective decentralized task distribution path planning. This paper proposed linear regression normalization across the node network in Dijkstra architecture to reduce computing time and memory consumption. The issue addressed through this optimization focused on reducing the possibilities of collision between AGVs and deadlock. The extended Dijkstra algorithm significantly reduces computing time compared to the traditional Dijkstra algorithm. In addition, the proposed solutions suggest better AGV routing for collision avoidance and deadlock prevention possibilities.

Introduction

Dijkstra algorithm was first conceived by Edsger W. Dijkstra [1, 2], a computer scientist, back in 1956. It was designed to compute the shortest path between nodes in a graph in 1959. It was the first design to compute the shortest path in internet protocol, mapping localization, routing protocol, social network analysis and path planning [3, 4]. This model favors research as it is highly efficient in computing the shortest path of a complex network. Researchers, scientists, and engineers have widely used the algorithm in various applications. The algorithm may exist in many forms nowadays due to its wide application and many variants of the extended version of the Dijkstra algorithm modified for different application types.

Much research has been done on optimizing and extending the Dijkstra algorithm. Dijkstra algorithm with multi-objective variant has been designed from terrain data to compute the optimal traversal in 3D surface [5, 6]. The computed path was obtained and implemented in the Global Rover Horizontal Optimization Planner (GRTOP) automation system to simultaneously set up optimized routes for multiple constraints. In another work, the Dijkstra algorithm has been modified with a vehicle emission measurement model and vehicle fuel consumption for a better driving experience [7, 8].

Dijkstra's algorithm can be used to find the shortest route between one city and all other cities. Widely used application of shortest path algorithms in network routing protocols, most notably IS-IS (Intermediate System to Intermediate System) and OSPF (Open Shortest Path First). It is also a subroutine in other algorithms such as Johnson's. For example, if the nodes of the graph represent cities and edge path costs represent driving distances between pairs of cities connected by a direct road (for simplicity, ignore red lights, stop signs, toll roads and other obstructions).

Fundamental of Dijkstra Algorithm

The Dijkstra algorithm is a fundamental algorithm for the AGV path planning. AGVs are autonomous vehicles that are used in transportation of various industries, such as warehousing, manufacturing, healthcare and logistics. AGVs require to navigate through complicated environments with obstacles and multiple paths to reach the destinations accurately.

The Dijkstra algorithm is a shortest path algorithm that can find the shortest path between two nodes in a graph. In AGV path planning, the nodes represent the locations in the environment, and the edges represent the connections between them. The weights assigned to the edges represent the distance or travel time between the locations.

The Dijkstra algorithm operated by starting at the source node and calculating the shortest path to all other nodes in the graph. It keeps track of the distances to each node and the shortest path that led to the target node. The algorithm then selects the node with the shortest distance and adds it to the traffic list of visited nodes. It repeats this process until it reaches the final target node.

AGVs can use the Dijkstra algorithm to calculate the shortest path between the source location and the target location while avoiding obstacles and taking into account the AGV’s speed and acceleration constraints. The algorithm can also be adapted to handle dynamic environments, where the weights on the edges are variable based on the real-time situations.

Overall, the Dijkstra algorithm is a fundamental algorithm for AGV path planning, as it can efficiently find the shortest path in complex environments with obstacles and multiple paths.

Implementation of Algorithm

Let the node at which the Dijkstra algorithm starts to be computed be called the initial node. Let the distance of node *Y* be the distance from the initial node to *Y*. Dijkstra's algorithm will initially start with infinite distances and will try to improve them step by step. Dijkstra model works effectively in finding the shortest path for various applications such as mapping, path planning, routing protocol and social network analysis. It is intended to configure the shortest-path issue for directed graphs with positive weights. Moreover, it can calculate graphs with the cycle as well. However, non-positive weights significantly impact efficiency by generating incorrect results. The setback of the conventional Dijkstra model is the latency issue which consumes high computing memory and time. This is mainly due to the computing parameter, including node, path, etc.

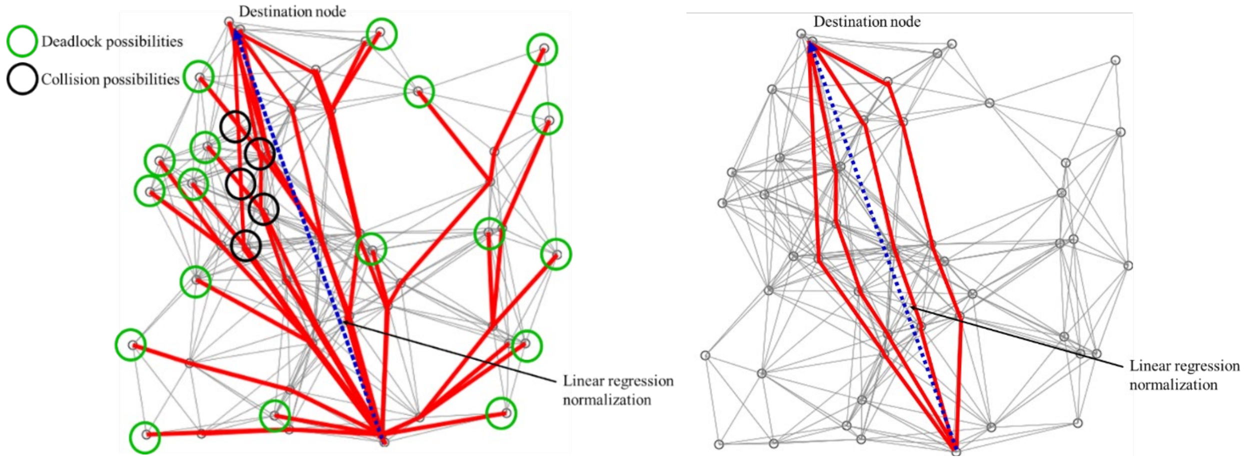

Figure 1. Dijkstra algorithm routing.
a) Dijkstra computing process, b) results of Dijkstra shortest path finding.

A traditional Dijkstra algorithm has been tested on a complex AGV network to determine its performance and setback. Fig. 1 (a) illustrates the computing process of the Dijkstra algorithm, while Fig. 1 (b) shows the results of the Dijkstra algorithm for four different nodes' weights. It can be concluded that the Dijkstra algorithm is best in finding the shortest path. However, it does not consider the traffic and unnecessary computing path. An unnecessary computing path produces higher memory consumption, which leads to longer computing time. The extended Dijkstra algorithm proposed in this paper eliminates the routing that provides possibilities of deadlock and collision.

Advantages of Extended Dijkstra algorithm in AGV path planning

Extended Dijkstra algorithm contributes to a shorter computation time where AGVs often operate in time-critical environments, and path planning needs to be done promptly. The extended Dijkstra algorithm can be faster than the traditional Dijkstra algorithm, especially for larger networks, as it searches from both nodes the origin and target nodes at the same time. This reduces significantly the search space and can result in a shorter computation time.

Additionally, the Extended Dijkstra algorithm able to reduce the memory usage. This benefits the AGVs typically the one with limited on-board memory, and extended Dijkstra algorithm anle to reduce memory usage as it only needs to use the storage for saving data on the nodes visited from each end.

Real-time updates: AGVs can encounter unexpected obstacles or changes in the environment that require real-time path planning updates. The extended Dijkstra algorithm can allow for real-time updates as it only needs to update the path from the point of intersection, rather than recomputing the entire path.

Improved efficiency: AGVs are often used in environments with complex structures, such as warehouses or manufacturing facilities. The extended Dijkstra algorithm can efficiently handle complex graphs with multiple paths between nodes, making it suitable for AGV path planning.

Methodology

Comparison analysis has been performed between both methods based on a simple AGV network, as in Fig. 2.

Method 1: Traditional Dijkstra Algorithm

The traditional Dijkstra Algorithm is derived from the Euclidean distance between adjacent nodes.

Input:

The shortest path obtained from a bidirectional graph as in Figure X with weights *K = (N, W)*, all nodes set N, connected weighted edges set nodes W.

The source node A.

The target node D.

Output:

Length and shortest path from the source node A to the target node D.

It can be predicted that d = (V, A, w) produce positive weight network, $V = (v_1, v_2, \ldots, v_n)$. Therefore, the minimum $D (v_i, v_j) \in A$ will fulfill the equation as below:

$u_1=0$

$u_j= \text{minimum } (u_k+w_{kj})$ $(j=2, 3, \ldots, n)$

Thus, the shortest path from node v_1 to a different node in D is determined from longest distance to nearest as follows:

$u_{i1} \le u_{i2} \le \ldots \le u_{in}$

Assuming $i_1=1$, $u_{i1}=0$, by substituting into equation (1), the results are as follows:

$u_{ij}=\text{minimum}\{u_{ik}+w_{ikij}\}$

$k \neq j$

$= \text{minimum } \{\text{minimum}\{u_{ik}+w_{ikij}\} \text{ minimum}\{u_{ik}+w_{ikij}\}\}$

$(j=2, 3, \ldots, n)$

The Adjacency matrix is defined as follows:

$$\left\{\begin{matrix}0 & 1 & 1 & 1 & 0 & 0 & 0 & 0\\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0\\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0\\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1\\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1\\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1\\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0\end{matrix}\right\}\begin{matrix}A\\ B1\\ B2\\ B3\\ C1\\ C2\\ C3\\ D\end{matrix}$$

The distance matrix is defined as follows

$$\begin{array}{cccccccc} A & B1 & B2 & B3 & C1 & C2 & C3 & D\end{array}$$
$$\left\{\begin{matrix}0 & 1 & 1 & 1 & \infty & \infty & \infty & \infty\\ 1 & 0 & 1 & \infty & 1 & \infty & \infty & \infty\\ 1 & 1 & 0 & 1 & \infty & \infty & 1 & \infty\\ 1 & \infty & 1 & 0 & \infty & 1 & \infty & \infty\\ \infty & 1 & \infty & \infty & 0 & \infty & 1 & 1\\ \infty & \infty & \infty & 1 & \infty & 0 & 1 & 1\\ \infty & \infty & 1 & \infty & 1 & 1 & 0 & 1\\ \infty & \infty & \infty & \infty & 1 & 1 & 1 & 0\end{matrix}\right\}\begin{matrix}A\\ B1\\ B2\\ B3\\ C1\\ C2\\ C3\\ D\end{matrix}$$

Adjacent matrix, distance matrix and correlation matrix are used to determine the shortest path on the Dijkstra network matrix in computer program and algorithm. However, the case in Figure X is simple to AGVs path routing. It uses Dijkstra to compute the shortest path between source node A and target node D. Numerous N*N arrays are identified to compute and store graphical data, where N represents the network node's count. When the count of N is large, it requires higher CPU memory. If there are 4000 nodes, it requires 64 MB memory; if there are 8000 nodes, it requires 256MB memory. This will give high latency to the system if multiple AGVs are waiting to compute the shortest path before executing a task. In this case, the Dijkstra algorithm is extremely difficult to implement in massive network analysis data.

Method 2: Extended Dijkstra Algorithm

As mentioned earlier, the extended Dijkstra is suggested to be implemented to decentralize AGV task distribution path planning. The capability to find the shortest paths from the nearest AGV node to the task node stops the algorithm once the shortest path to the destination node has been determined.

The traditional Dijkstra algorithm has been used for several decades to compute the shortest path in a graph between two nodes by computing the distance matrix and adjacent matrix. The application is not only limited to computing the shortest path in a graph. It applies to any case with two defined parameters: adjacent and distance. The adjacent consists of the source node and target node.

This paper's proposed extended Dijkstra algorithm is derived from the inversed multi-linear regression. The derivation enables multiple Dijkstra algorithms to compute simultaneously with fewer possibilities of collision and deadlock. Besides, the proposed extended Dijkstra is also applicable for various applications such as multicasting, internet protocol and routing.

$$D^* = (B \cdot C) + \varepsilon$$

$$D = \begin{cases} 1 \text{ if } D^* \leq \theta_1, \\ 2 \text{ if } \theta_1 < D^* \leq \theta_2, \\ 3 \text{ if } \theta_2 < D^* \leq \theta_3 \\ \vdots \\ K \text{ if } \theta_{K-1} < D^* \end{cases}$$

$$P(D = k|\mathrm{x}) = P(\theta_{k-1} < \theta_k|\mathrm{x}$$
$$= P(\theta_{k-1} < B \cdot C + \varepsilon \leq \theta_k)$$
$$= \Phi(\theta_k - B \cdot C) - \Phi(\theta_{k-1} - B \cdot C)$$

$$\log \mathcal{L}(B, \theta|C_i, y_i) = \sum_{k-1}^{K} [y_i = k] \log[\Phi(\theta_{k-1} - B \cdot C) - (\theta_{k-1} - B \cdot C_i)]$$

The proposed extended Dijkstra model is intended to be used in task distribution decentralization path planning for AGV with an extensive network. The proposed approach uses lower memory consumption, thus computing the shortest path faster than the conventional Dijkstra model. In addition, the proposed extended Dijkstra model can reduce collision and deadlock possibilities in multiple AGV network task distributions.

Result

Fig. 2 illustrates the computed results using traditional and extended Dijkstra algorithms. The comparison was performed for the AGV path planning network to compute from source node A to target node D. In the case of implementation, and multiple AGVs share the same network executing multiple tasks simultaneously. The path is computed using the traditional Dijkstra algorithm utilizing the shortest path. Therefore, the path is computed preferably to avoid passing by many nodes and, at the same time, maintain the shortest distance from source node A to target node D. This solution is designed for multiple AGVs task distribution path planning.

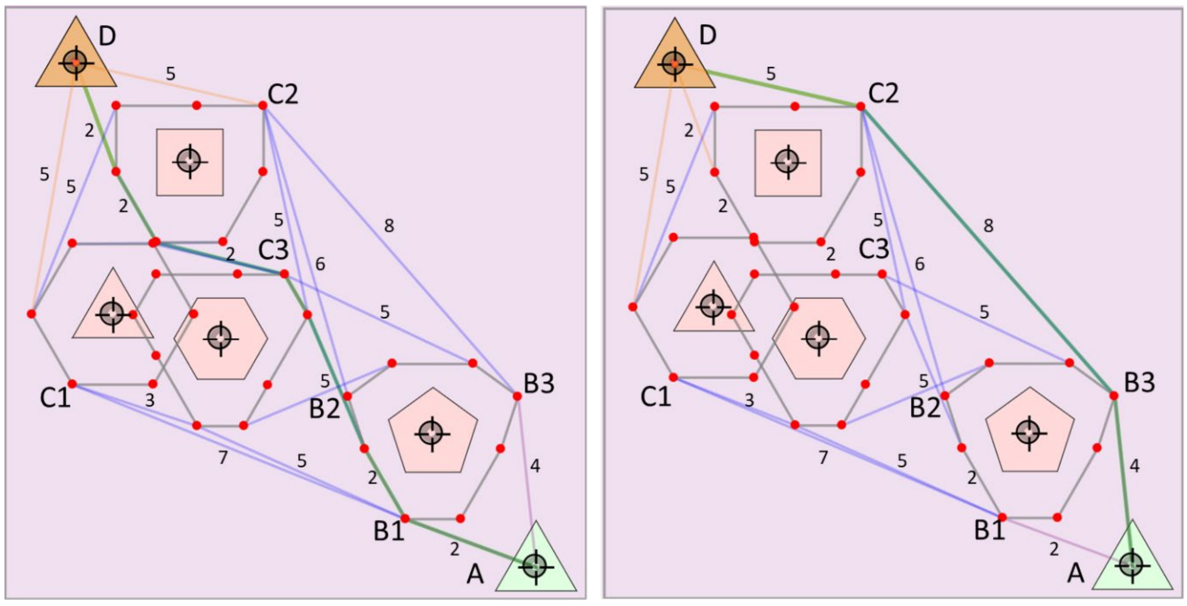

Figure 2. Path-finding results.
a) Traditional Dijkstra algorithm, b) Extended Dijkstra algorithm.

The computed path from origin A to destination D shown in Fig. 2 (a) path selected by Dijkstra algorithm using the shortest path but passing by common nodes and heavy traffic nodes which cause high possibilities of collision and deadlock if implemented in multiple AGVs network. Meanwhile for extended Dijkstra algorithm select the appropriate path while maintaining the shortest path without passing by heavy traffic nodes.

There are few setbacks identified in Dijkstra algorithm implementation in multiple AGVs task distribution path planning, especially in long compute time if involved multiple nodes and distance. There are few possibilities of collision and deadlock since the computed path using the Dijkstra algorithm passes through commonly used nodes which refer to high traffic congestion nodes and paths.

Simulation in Wolfram Mathematica shows traditional Dijkstra algorithm comsume 256MB, 60 seconds, while extended Dijkstra algirithm consume 32MB, 7 seconds of memroy and computing time respectively for the same network.

Conclusion

An extended version of Dijkstra algorithm has been proposed in this paper for multiple AGV path planning solutions. The traditional Dijkstra algorithm is used to compute the shortest path between the source node and target node. However, the extended Dijkstra able to compute faster and consume less memory compared to the traditional Dijkstra algorithm. Furthermore, the extended Dijkstra algorithm reduce the possibilities of collision and facilitate with deadlock prevention in multiple AGVs network. It is also suitable to be used in large network of AGVs for task distribution path planning. The extended algorithm has been validated through an experimental verification using turtlebots by considering the inputs from LiDAR sensor as an additional weight to the network.

References

[1] E. W. Dijkstra, E. W. Dijkstra, E. W. Dijkstra, and E. W. Dijkstra, A discipline of programming. prentice-hall Englewood Cliffs, 1976.

[2] E. W. Dijkstra, "A note on two problems in connexion with graphs," Numerische mathematik, vol. 1, no. 1, pp. 269-271, 1959.

[3] M. Noto and H. Sato, "A method for the shortest path search by extended Dijkstra algorithm," in Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics.'cybernetics evolving to systems, humans, organizations, and their complex interactions'(cat. no. 0, 2000, vol. 3: IEEE, pp. 2316-2320.

[4] Y.-z. Chen, S.-f. Shen, T. Chen, and R. Yang, "Path optimization study for vehicles evacuation based on Dijkstra algorithm," Procedia Engineering, vol. 71, pp. 159-165, 2014.

[5] W. Fink, V. R. Baker, A. J.-W. Brooks, M. Flammia, J. M. Dohm, and M. A. Tarbell, "Globally optimal rover traverse planning in 3D using Dijkstra's algorithm for multi-objective deployment scenarios," Planetary and Space Science, vol. 179, p. 104707, 2019.

[6] W. Fink, V. R. Baker, D. Schulze-Makuch, C. W. Hamilton, and M. A. Tarbell, "Autonomous exploration of planetary lava tubes using a multi-rover framework," in 2015 IEEE aerospace conference, 2015: IEEE, pp. 1-9.

[7] D. Guo et al., "A vehicle path planning method based on a dynamic traffic network that considers fuel consumption and emissions," Science of the Total Environment, vol. 663, pp. 935-943, 2019.

[8] Y. C. Zheng et al., "Study of multi-objective path planning method for vehicles," Environmental Science and Pollution Research, vol. 27, no. 3, pp. 3257-3270, 2020.