Indonesian Journal of Electrical Engineering and Computer Science
Vol. 24, No. 2, November 2021, pp. 1173~1182
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v24.i2.pp1173-1182

Guidance system based on Dijkstra-ant colony algorithm with binary search tree for indoor parking system

K. Ibrahim Ata, A. Che Soh, A. J. Ishak, H. Jaafar
Department of Electrical and Electronic Engineering, Universiti Putra Malaysia (UPM), Selangor, Malaysia

Article Info

Article history:

Received Jun 8, 2021
Revised Aug 23, 2021
Accepted Sep 1, 2021

Keywords:

Ant colony optimization
Binary search tree
Djikstra's alogrithm
Single-source shortest path

ABSTRACT

A common algorithm to solve the single-source shortest path (SSSP) is the Dijkstra algorithm. However, the traditional Dijkstra's is not accurate and need more time to perform the path in order it should visit all the nodes in the graph. In this paper, the Dijkstra-ant colony algorithm (ACO) with binary search tree (BST) has been proposed. Dijkstra and ACO are integrated to produce the smart guidance algorithm for the indoor parking system. Dijkstra algorithm initials the paths to finding the shortest path while ACO optimizes the paths. BST has been used to store the paths that Dijkstra algorithm initialled. The proposed algorithm is aimed to control the shortest path as well as guide the driver towards the nearest vacant available space near the entrance. This solution depending on applying the optimization on an optimal path while the traditional ACO is optimizing the random path based on the greedy algorithm hence we get the most optimal path. Moreover, the reason behind using the BST is to make the generation of the path by Dijkstra's algorithm more accurate and less time performance. The results show a range of 8.3% to 26.8% improvement in the proposed path compared to the traditional Dijkstra's algorithm.

Corresponding Author:

K. Ibrahim Ata
Department of Electrical and Electronic Engineering
Universiti Putra Malaysia (UPM), 43400 Serdang, Selangor, Malaysia
Email: eng.k.alata1@gmail.com

1. INTRODUCTION

In recent years, there has been a resurgence of interest in the shortest path problem for usage in various transportation engineering applications. In a distributed route guidance system (RGS), an in-vehicle computer is commonly used to calculate the optimal route in a large traffic network [1]. Typically, the recommended routes must be found within a very short period of time [2]. In real-time, automated vehicle dispatching system (AVDS) new routes and schedules must be identified within a reasonable time after a customer requests a service. Because the travel times are the basic input to the real-time routing and scheduling processes and are dynamic in most urban traffic environments [3].

There is an implicit requirement to use a minimum path algorithm repeatedly during the optimization procedure [4]. Shortest path algorithms are currently used widely. They are the basis of some problems such as network flow problems, tree problems, and other related problems. There are many algorithms used to find the shortest path such as Dijkstra, A* algorithm, genetic algorithm, Floyd algorithm and Ant colony optimization algorithm [5]-[7]. All these algorithms demonstrated its ability to find the shortest path in the guidance system. However, some of them are not accurate enough to find the shortest and the optimist path. Hence, these algorithms need to be modified or optimized by other optimization algorithms [8], [9]. Many of successful studies which solved the shortest path by a professional algorithms such as [5],

[10], [11]. These studies have been focused on the Dijkstra’s, and A* algorithm to estimate search time and distance of algorithms to find the shortest path. They have presented the difference between both algorithms in the practical experiment. The results showed that A* behaves much more like Dijkstra’s, the only difference between both algorithms is that A* gives a better path by using a heuristic function while Dijkstra's just explore all possible paths. A* accomplished a better performance by using heuristics to guide its search and gives the optimal result much faster. Moreover, there are another type of the algorithm can find the shortest path such as genetic algorithm (GA) which is search, optimization, and machine learning techniques based on the mechanics of Natural Selection and natural genetics [12], [13]. GA is adaptive procedures of optimization and search that find solutions to problems by an evolutionary process inspired in the mechanisms of natural selection and genetic science. GA is effective at taking large, potentially huge search spaces and navigating them, looking for optimal combinations of things, solutions difficult to accomplish [14]. A protocol namely energy efficient ant colony optimized data transmission (EACODT) has been proposed [15]. A genetic algorithm has been found for the nearest vacant parking bay to the location. The proposed method is tested for different scenarios and accurate results are obtained.

Another work has hybridized the GA with Dijkstra for a mobile robot in a static environment with obstacles [16]. The aim of this study is to combine the global search capability of the GA and the local exact solution capability of the Dijkstra’s algorithm to obtain a faster solution for the shortest path problem. It is found that the proposed method is especially suitable for the large dimensional robot path planning problems in future applications. The algorithm is planned to be used in real-time with a laboratory mobile robot. The simulation experiment showed that the combination of the two algorithms achieved a possitive result in terms of time and iteration number. However, for small size problem, the Dijkstra’s algorithm provides the solution in a shorter time [17]. It is noted that when the problem size increases, the Dijkstra’s algorithm requires longer time to solve the problem. There is an unlimited application using Dijkstra’s algorithm to find the shortest path especially in a single source problem [18], [19]. In other work, the large computation has been decreased and the efficiency has been increased by a constrained optimization condition applied to the Dijkstra’s algorithm [20]. This system reduces the traditional Dijkstra computation successfully. However, it is a high-cost product and a default installation due to the usage of Wi-Fi, Bluetooth, and Zigbee together. On the other hand , the optimal path in communication and networks have been calculated by applying Ant colony algorithm (ACO) [21], [22]. These systems reduce the complexity of the ACO and the energy consumption form the network. Thus, the lifetime of the network has been prolonged successfully.

In this paper, Dijkstra and ACO algorithms are integrated to produce the shortest path algorithm guidance system for the indoor parking system. This proposed solution will give the shortest path in fastest time by using binary search tree (BST) and accurate way by using ACO compared with the previous solutions. Dijkstra’s algorithm initials the paths to finding the shortest path while ACO optimizes the paths. Binary search tree has been used to store the data or information and retrieve it upon request. Next, the comparison proposed algorithm with the existing Dijkstra also discussed. The procedure of each algorithm will be explained in details. The application of them will be surmised. Then, the proposed solution depending on the algorithm hybridizing will be explained and applied on the parking system as a case study.

2. RESEARCH METHOD

This research applied the Dijkstra’s algorithm with ACO by using BST to find the shortest path in a parking system as a case study. Figure 1, shows the case study system which consists of the graphical user interface which represent the main part of the system that will calculate the shortest path and show the path on graphical user interface (GUI) and other hardware peripherals which devided into two circuites. The main circuit that includes GUI and RFID components. The proposed algorithm to calculate the shortest path and the optimal parking bay is realized by hybridizing Dijkstra and ant colony algorithm. In our meathod, dijkstras algorithim is used to calculate the shortest path from the parking bay to the mall entrance, and ACO is used to choose the path wich is the nearest path to the entrance and the nearest path to the parking gate at the same time. Hence, the optimal path will reduce both the driving and walking distance. Figure 2, shows the main steps of the proposed solution flowchart.

The driver needs to select the entrance from the GUI and specify the car size. Once the required ticket is selected by the driver, the BST will be created and start searching in the nodes. The BST is simple in the implementation and enables changes on the tree without interrupting the performance of the tree in the opposite, the array is fixed and does not provide dynamic data [23], [24]. The first step is to choose a node in the middle which is node 50 because we have 99 nodes. All the distances between each node and the intentional entrance will be presented. Then the root of the tree is assigned as x and x=y, y will be assigned as the parent of the next node.

The added node which is i=0 is checked to determine whether its distance to the intentional entrance is of a bigger value than the distance from node x to the intentinal entrance or of a smaller value than it. If the node i is bigger than x, it will search in the right subtree, if it is smaller, it will search on the left subtree. Then it checks if the current x = null or not. If x does not equal to null, that means there is a possibility for the node, and it completes the search. If this x is equal to null that means this is the last node and there are no possibilities, so y is the parent. Then the BST will check if the node i is smaller or bigger than the parent which is node y. If it is smaller than the parent, the node will be inserted to the left subtree. However, if it is bigger than the parent, the node will be inserted to the right subtree. Then it will check if i equals to 99 or not. If i gets to 99 that means the search is finished. If not, the BST will repeat the process until all nodes are inserted in the tree. Finally, BST will assign the information of the car size, the intentional entrance, the vacant parking bay, the edges between the nodes and the gate which the driver is using it to enter to the parking area. Figure 3, shows the BST procedure flowchart in the program.

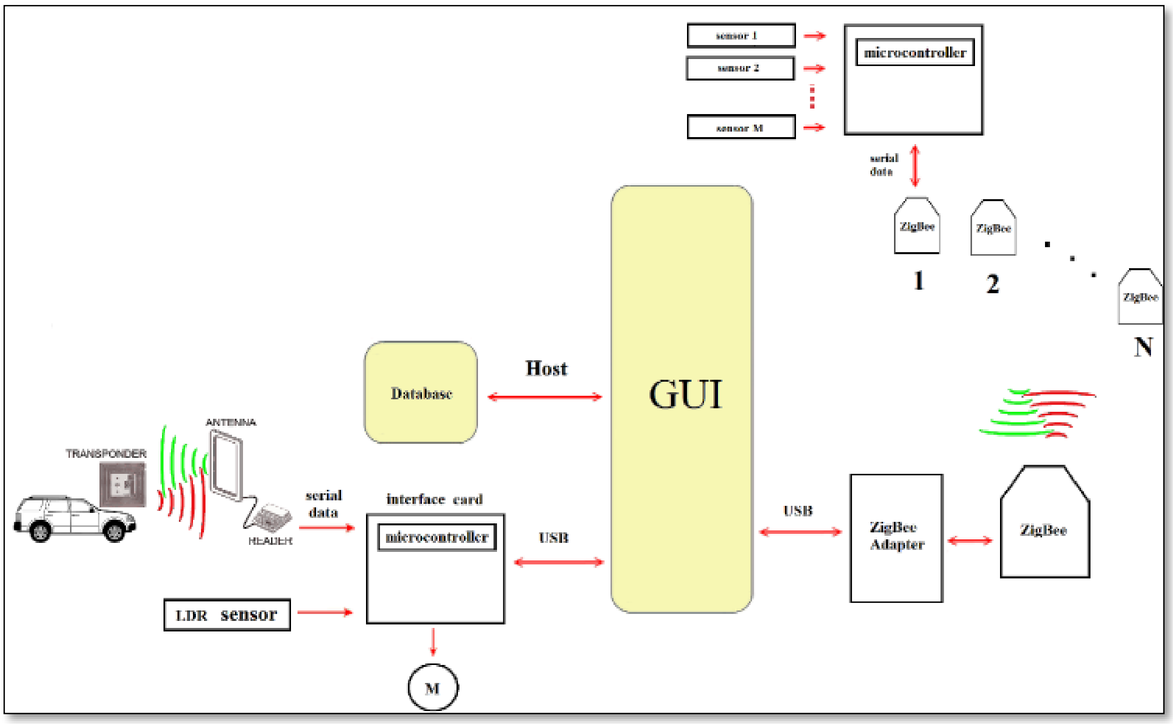

Figure 1. The indoor guidance parking system

The BST will give the related information to the Dijkstra’s algorithm. This information is the car size, the intentional entrance, the vacant parking bay, the edges between the nodes and the gate which the driver is using it to enter to the parking area. This will reduce the time for Dijkstra’s algorithm in finding the related and nearest nodes. Then the Dijkstra’s algorithm will calculate the shortest path between the gate and the intentional bay according to its procedure. Then it finds the shortest path from the specified source which is the gate to the destination which is the parking bay. Then it will maintain a matrix in which the previous equals unknown. Dijkstra will assign Zero to gate node(u) and infinity to all other nodes. Then it will define u which is the smallest distance to the current vertex which is the intentional parking bay. Then it will consider u as a visited node and remove it from the Q. Then its neighboring nodes will be calculated as distance u + the distance between u and its neighbor v. This procedure will be applied for all the neighbors of u, then they will be updated with the new minimum distance value (alt). All the visited and updated nodes with the smallest value will not be checked again. This step will pass all the nodes in the layout until arriving at the parking bay node. In every time the Dijkstra will compare the neighbor nodes, it will choose the shortest path. In this way, it will get into the parking bay in a minimal edge weight. Figure 4, shows the flowchart of the algorithm steps in the program. Hence, the ACO has been introduced to optimize this path based on its strategy. The flowchart in Figure 5 shows the implemented ant colony algorithm.

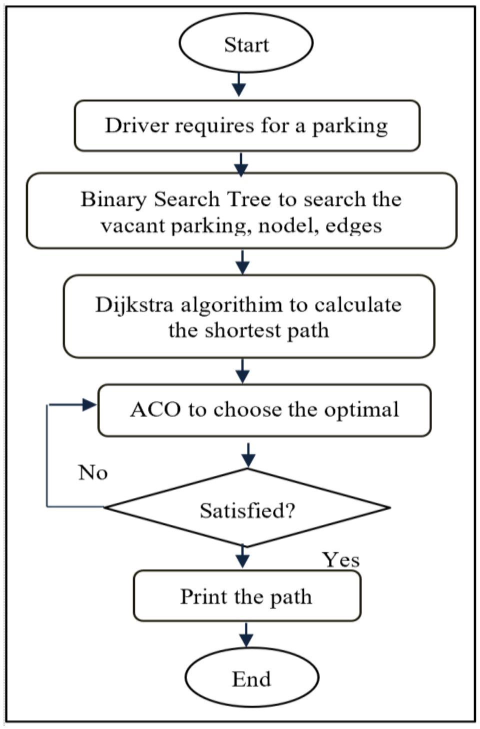
Start
Driver requires for a parking
Binary Search Tree to search the vacant parking, nodel, edges
Dijkstra algorithim to calculate the shortest path
ACO to choose the optimal
No
Satisfied?
Yes
Print the path
End

Figure 2. The steps of the proposed algorithm

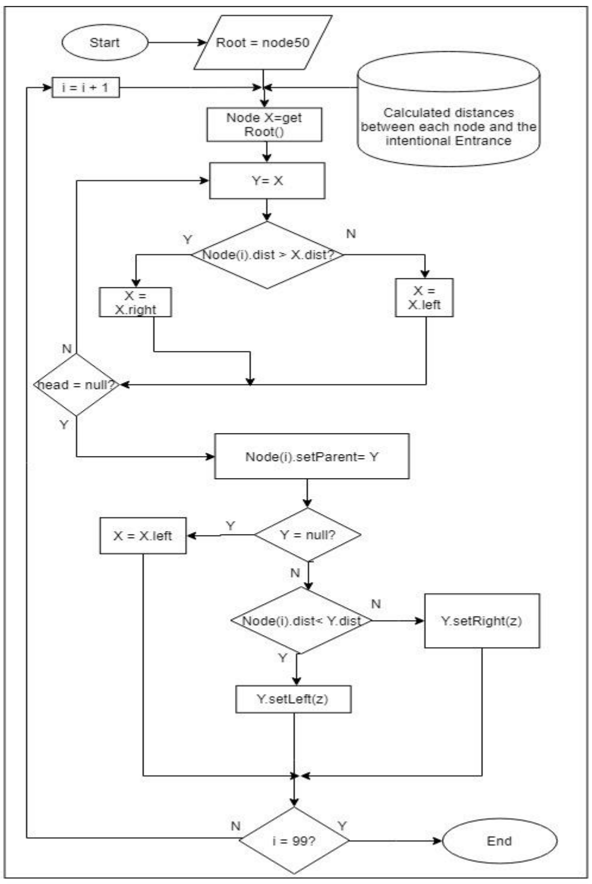

Figure 3. The BST flowchart

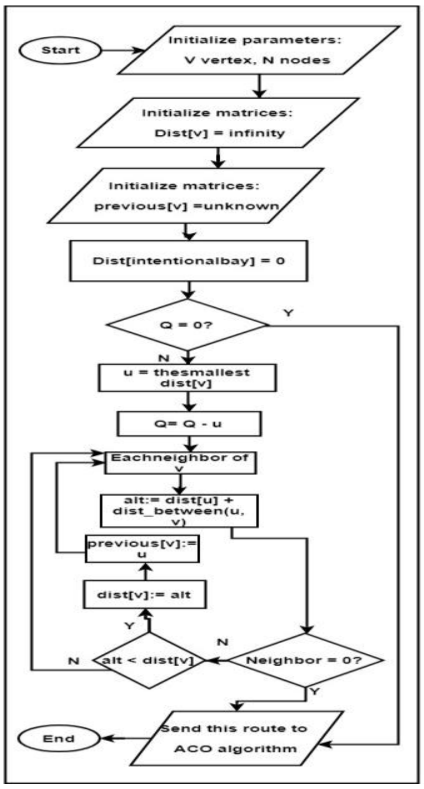
Start
Initialize parameters: V vertex, N nodes
Initialize matrices: Dist[v] = infinity
Initialize matrices: previous[v] =unknown
Dist[intentionalbay] = 0
Q = 0?
u = thesmallest dist[v]
Q= Q - u
Eachneighbor of v
alt:= dist[u] + dist_between(u, v)
previous[v]:= u
dist[v]:= alt
alt < dist[v]
Neighbor = 0?
Send this route to ACO algorithm
End

Figure 4. The dijkstra's algorithm flowchart

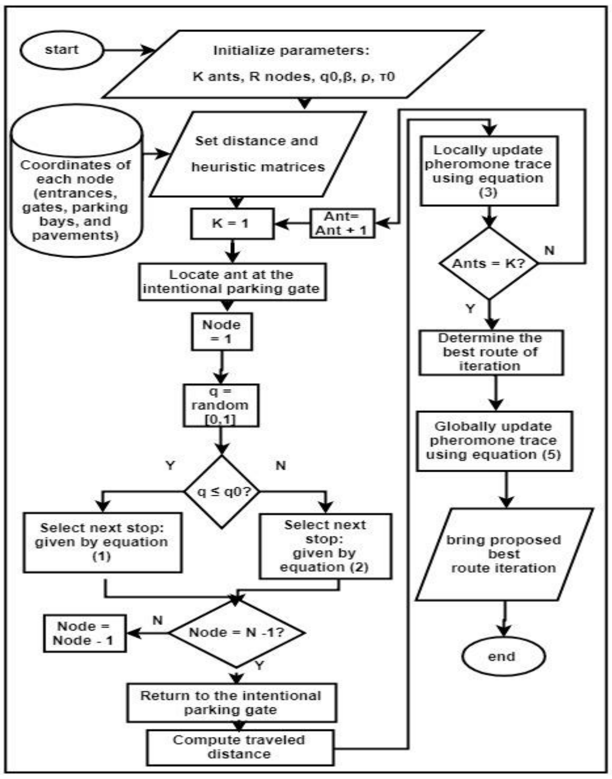
start
Initialize parameters: K ants, R nodes, q0,β, ρ, τ0
Coordinates of each node (entrances, gates, parking bays, and pavements)
Set distance and heuristic matrices
K = 1
Ant= Ant + 1
Locate ant at the intentional parking gate
Node = 1
q = random [0,1]
q ≤ q0?
Select next stop: given by equation (1)
Select next stop: given by equation (2)
Node = Node - 1
Node = N -1?
Return to the intentional parking gate
Compute traveled distance
Locally update pheromone trace using equation (3)
Ants = K?
Determine the best route of iteration
Globally update pheromone trace using equation (5)
bring proposed best route iteration
end

Figure 5. The ACO flowchart

In nature, the ants start roaming randomly looking for food to bring to their colony. In the system, food is represented as a value (0, 1). When the status of the parking is 0, it means there is an unoccupied parking bay which indicates food for ant. When the status of the parking is 1, it means parking bays are occupied which indicates there is no food. Moreover, those ants deposit the pheromone on the path they used. This pheromone evaporates by time if the path has not been used for a while. That means another path is used by other ants. It keeps updating the pheromone which helps the other ants to use this path as a food path. Those ants are represented as a number in the simulation. Moreover, the pheromone has been represented as a value which can be increased and decreased based on ant's procedure. Once the ant finds a vacant parking bay nearby the parking that has been found by Dijkstra's algorithm, it starts optimizing this path based on its procedure.

In the beginning, the ACO will initialize its parameters which are the nodes and ants' number (10), the initial path is generated by Dijkstra's algorithm and the pheromone which represented as acounter. Then ACO will set the distances of the nods and the matrices. Once all the parameters are ready, the ACO will start by the first ant which is K assigned as 1 and locate it on the initial node which is the gate used by the car. Then the ant will start from the first node and will define a random number from zero to one. If this random number is smaller than the initial path which is q0, the (1) will be applied.

$$S = \begin{cases} \arg\max u \in jk(r)\{[\tau(r,u)]^{\alpha}.[\eta(r,u)]^{\beta}\} & if\ (q \le q_0(exploitation) \\ s, & otherwise(biased\ \ eploration) \end{cases} \tag{1}$$

If the random number is bigger than the initial path which is q0, the (2) will be applied.

$$P_{k(r,s)} \begin{cases} \dfrac{[\tau(r,s)]^{\alpha}.[\eta(r,s)^{\beta}}{\sum_{u\epsilon jk(r)}[\tau(r,u)]^{\alpha}.[\eta(r,u)]^{\beta}}, if\ s \in j_k(r) \\ 0, \quad otherwise \end{cases} \tag{2}$$

Then the ant will deposit pheromone on the edges as in the (3).

$$\tau(r,s) \leftarrow (1-p).\tau(r,s) + p.\Delta\tau_0 \tag{3}$$

where 0 < ρ < 1 is a parameter. The term Δτij may be defined as the (4).

$$\Delta\tau_0({}^{1}/_{l_{mn}}.n) \tag{4}$$

This step will be repeated until all the ants are initialized. If all the ants are initialized, there will be 10 ants in this case. Then ACO will determine the best route of the iteration by depositing additional pheromone on each visited edge using the (5).

$$\tau(r,s) \leftarrow (1-p).\tau(r,s) + p.\Delta\tau(r,s) \tag{5}$$

The pheromone level update is performed at the end of an iteration using the (6).

$$\Delta\tau(r,s) = \begin{cases} \dfrac{1}{L_{gb}}, & if\ (r,s) \in globalbest\ tour \\ 0, & otherwise \end{cases} \tag{6}$$

where ρ is the pheromone decay parameter (0 < ρ < 1), and L best is either LGB (the length of the globally best tour since the start of algorithm execution) or Lib (the length of the best tour found during the current iteration of the algorithm) [25]. Then the optimal path will be determined and showed to the user.

The optimized path by ant is created and sent to the driver as a printed path to ease the way for the driver. The parking bay will be reserved for three minutes. After three minutes if the driver has not followed the reserved parking, the booking will be cancelled. This procedure repeats the same steps for all gates and entrances. Finally, the path is created, and the driver is ready to move to the desired parking. The simulation has been implemented to provide an optimal path which helps people to save time, fuel consumption and solve the traffic jam problem especially at shopping malls. Moreover, the cases and scenarios have been examined and explained in the next section to illustrate the benefits of the proposed system.

3. RESULTS AND DISCUSSION

The comparison between the proposed solution and the traditional Dijkstra's algorithm is implemented depending on the path length which is generated in the program. Different cases are generated randomly to confirm the proposed solution effectiveness. In some cases, the generated path is the same because it is already the optimal path. However, in some cases, when the path generated by traditional Dijkstra is not the optimal and there is another path is shorter, the optimization chooses the shorter one considering both driving and walking distance.

In this layout, the weight edges are calculated by the distance formula between two points with coordinates (x_1, y_1) and (x_2, y_2) which is $d=\sqrt{((x_2-x_1)^2 +(y_2-y_1)^2)}$. For example, the node number 0 has two edges; the first edge is between 0 and 1, and the second edge is between 0 and 39. The Table 1 shows the nodes ID and their coordinates and Table 2 shows the edges weights. All the vertices and the weights of the edges between them are calculated in the same way.

Table 1. The nodes' locations in the layout

Node ID	X	Y
0	80	70
1	80	100
39	100	70

Table 2. The edges weight

Edge	Edges wieght
(0,1)	30
(0,39)	20

The case 1 was applied on gate 1 with intentional entrance 1 and standard car. The traditional Dijkstra's algorithm chose the parking bay b47 as shown in Figure 6. On the other hand, the proposed solution chose the parking bay number b45 as shown in Figure 7. The walking distance was the same in both cases. However, the driving distance in the proposed solution was shorter by 6 meters. Thus, the path length is optimized by 26.08%. For case 2, the algorithm was applied on gate 1 with intentional entrance 3 and standard car. The traditional Dijkstra's algorithm chose the parking bay b25 as seen in Figure 8 and the proposed solution chose the parking bay number b56 as seen in Figure 9. The length of walking distance also the same in both cases. While the driving of the traditional Dijkstra was 63 meters, it was 52 meters in the proposed solution. Thus, the path length by the proposed solution is optimized by 17.46%. The case 3 was applied on gate 2 with intentional entrance 3 and a small car. The traditional Dijkstra's algorithm chose the parking bay S35 as illustrated in Figure 10 and the proposed chose the parking bay number S36 as illustrated in Figure 11. The walking distance was the same in both cases. While the driving distance in traditional Dijkstra was 36 meters, it was 33 meters in the proposed solution. Thus, the path length by the proposed solution is optimized by 8.3%.

Other scenario which is the case 4, the algorithm was applied on gate 2 with intentional entrance 2 and standard car. The traditional Dijkstra's algorithm chose the parking bay b51 as demonstrated in Figure 12. On the other hand, the proposed technique chose the parking bay number b52 as demonstrated in Figure 13. The walking distance was the same in both cases. While the driving distance in traditional Dijkstra was 37 meters, it was 34 meters in the proposed solution. Thus, the path length by the proposed solution is optimized by 8.10%. Last scenario, the case shows when the path generated by traditional Dijkstra is the best and no other better choices, so the optimization will confirm that this is the optimal path and choose it. For example, in Figure 14, the traditional Dijkstra chose the bay b47 with a path length of 46 meter and a distance to the entrance of 22 meter. Moreover, the proposed solution chose the same parking bay, because if it chose the b49 or b45 which are the nearest vacant bays, the path length will be shorter but the distance to the entrance will be longer. Thus, the proposed solution chose the same path which is already the optimal bay among all the nearest and available bays as shown in Figure 15.

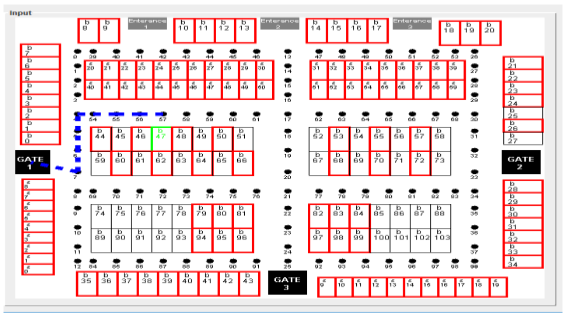
Figure 6. Dijkstra's algorithm for case 1

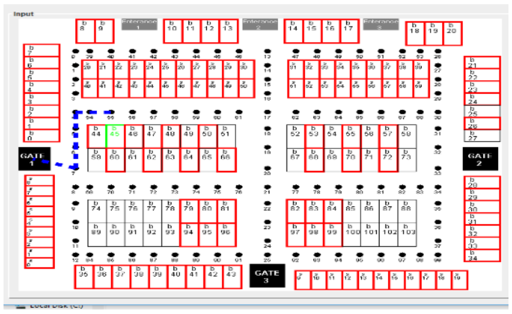
Figure 7. The proposed algorithm for case 1

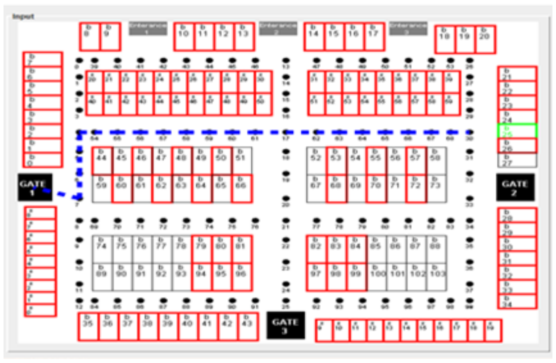
Figure 8. Dijkstra’s algorithm for case 2

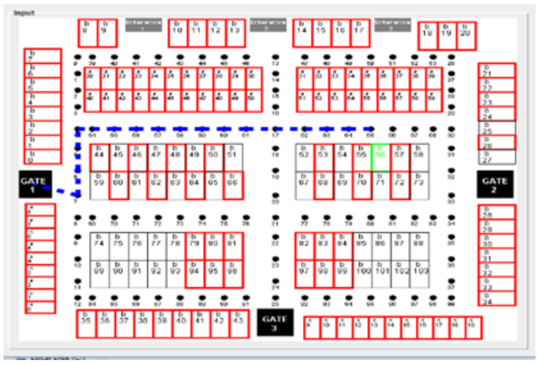
Figure 9. The proposed algorithm for case 2

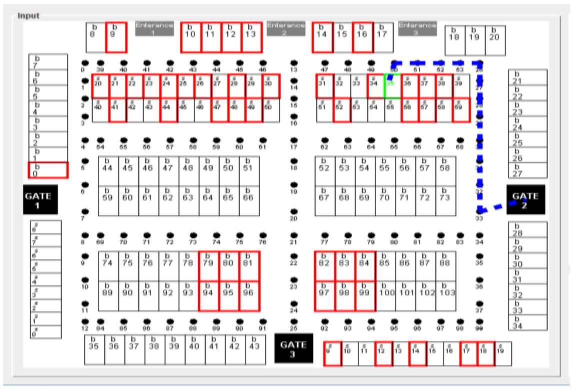
Figure 10. Dijkstra’s algorithm for case 3

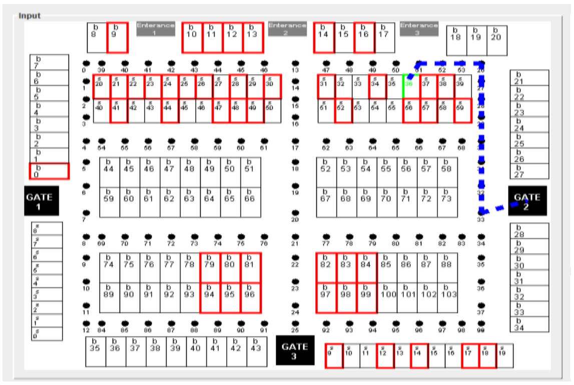
Figure 11. The proposed algorithm for case 3

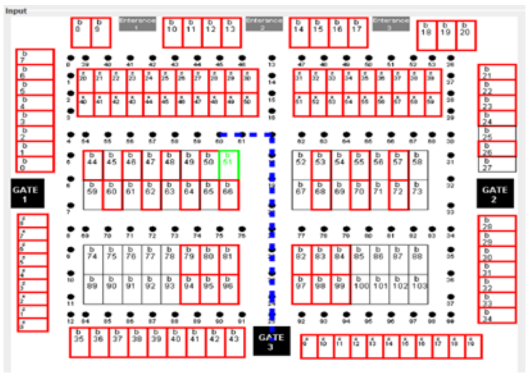
Figure 12. Dijkstra’s algorithm for case 4

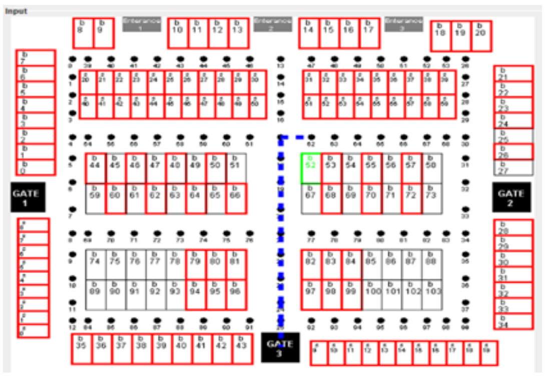
Figure 13. The proposed algorithm for case 4

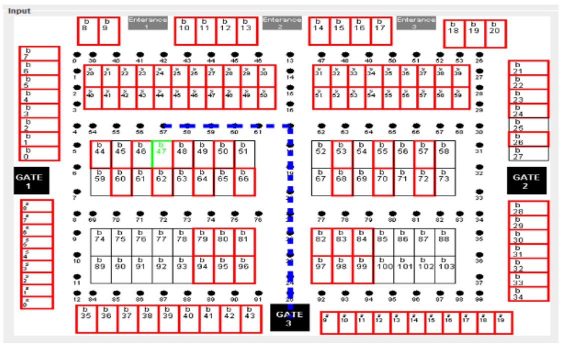
Figure 14. Dijkstra’s algorithm for case 5

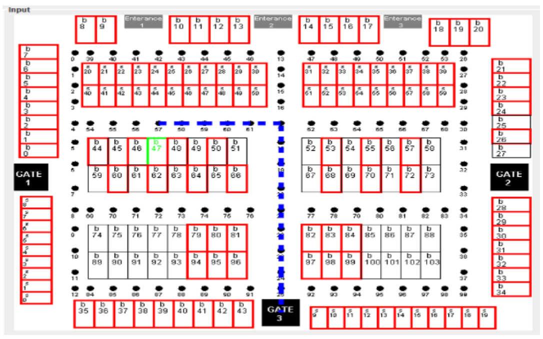
Figure 15. The proposed algorithm for case 5

In order to show the overall differences of all the cases, the average values for the Dijkstra’s and the proposed paths have been computed as seen in Table 3. Form the Table 1, the overall cases for the traditional Dijkstra’s algorithm and the proposed solution. In this table, Dijkstra’s algorithm is used to calculate the shortest path between the source (gate) and the destination (parking bay). In some cases, this path will be the optimal path, but in other cases, there is another vacant parking bay has path shorter than the bay which Dijkstra’s algorithm has chosen. Thus, the ACO tried to find another parking with shorter path. If the pheromone has been highlighted in that path, it will be chosen as the optimal path which is initialized by Dijkstra’s algorithm which will be cancelled.

Table 3. The performance of proposed solution (Dijkstra-ant colony algorithm) compare to Dijkstra's algorithm

Scenario	Dijkstra's Algorithm		Dijkstra-Ant Colony Algorithm	
	Path Length(m)	Walking Distance(m)	Path Length(m)	Walking Distance(m)
Case 1	23	22	22	17
Case 2	63	22	22	52
Case 3	36	8	8	33
Case 4	37	22	22	34
Case 5	46	22	22	46
AVERAGE	**41**	**19.2**	**19.2**	**36.4**

4. CONCLUSION

The binary search tree prepared the inquired data to the Dijkstra’s algorithm. Dijkstra’s algorithm has initialized the path to the ant colony optimization. The ant colony optimization optimized the path and has reselected the parking according to the distance from the gate to the parking bay and the distance from the parking bay to the intentional entrance. The system has been applied to the proposed layout which provided more parking bays by the car size classification.

The system graphical user interface has been tested and simulated successfully by Java. The GUI has been implemented to allow the user interaction entering the parameters and to achieve its intentional objective properly. The scenarios of the program have been discussed in details. Different cases have been compared between the traditional Dijkstra and the proposed solution. The average value of the generated paths is calculated and compared. The proposed algorithm has achieved positive outcomes in comparison to the traditional Dijkstra’s algorithm with regards to the shortest path. The results show a range of 8.3% to 26.8% improvement in the proposed path compared to the traditional Dijkstra’s algorithm. The findings also indicate that the proposed algorithm for shortest path algorithm for guidance system based on Dijkstra-Ant Colony Algorithm in the indoor parking system will help in reducing the time wasted in searching for a parking bay and will increase the efficiency of the parking system in shopping malls. For the future work, the Smart Indoor Guidance Parking System should have application on the smartphone which provides online parking reservation, so the driver could make a reservation for the parking.

ACKNOWLEDGEMENTS

This work is financed by Universiti Putra Malaysia Grant Scheme (Putra Grant). The authors would also like to acknowledge the Department of Electrical and Electronic Engineering, Universiti Putra Malaysia for supporting this project.

REFERENCES

[1] M. Zouari, N. Baklouti, J. Sanchez-Medina, H. M. Kammoun, M. Ben Ayed, and A. M. Alimi, “PSO-based adaptive hierarchical interval type-2 fuzzy knowledge representation system (PSO-AHIT2FKRS) for travel route guidance,” *IEEE Transaction Intelligent Transportation System*, vol. 99, pp. 1–15, 2020, doi: 10.1109/TITS.2020.3016054.

[2] L. Fu, D. Sun, and L. R. Rilett, “Heuristic shortest path algorithms for transportation applications: State of the art,” *Computers and Operations Research*, vol. 33, no. 11, pp. 3324–3343, 2006, doi:10.1016/j.cor.2005.03.027.

[3] M. R. Jabbarpour, H. Malakooti, R. M. Noor, N. B. Anuar, and N. Khamis, “Ant colony optimisation for vehicle traffic systems: applications and challenges,” *Int. J. Bio-Ins. Comp.*, vol. 6, no. 1, pp. 32–56, 2014.

[4] K. I. Ata, A. C. Soh, A. J. Ishak, and H. Jaafar, “A smart guidance indoor parking system based on dijkstra’s algorithm and ant colony algorithm,” *Int. J. Intr, Eng.*, vol. 12, no. 2, pp. 1–9, 2020, doi: 10.30880/ijie.2020.12.02.001.

[5] A. S. Alija, “Analysis of Djikstra’s and A* algorithm to find the shortest path,” Master Thesis, Universiti Tun Hussein Onn Malaysia, 2015.

[6] A. M. Hasan and S. M. Rafaat, “Optimized formation control of multi-agent system using PSO algorithm,” *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 20, no. 3, pp. 1591–1600, 2020, doi: 10.11591/ijeecs.v20.i3.pp1591-1600.

[7] A. Fitriansyah, N. W. Parwati1, D. R. Wardhani and N. Kustian, " Dijkstra's algorithm to find shortest path of tourist destination in Bali", *Journal of Physics: Conference Series* 1338 (2019) 012044, pp 1-7,2019, doi:10.1088/1742-6596/1338/1/012044.
[8] G. U. Mali and D. K. Gautam, "Shortest path evaluation in wireless network using fuzzy logic," *Wireless Personal Communication*, vol. 100, no. 4, pp. 1393–1404, 2018, doi: 10.1007/s11277-018-5645-1.
[9] O. A. Gbadamosi and D. R. Aremu, "Design of a modified Dijkstra's algorithm for finding alternate routes for shortest-path problems with huge costs", *International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS*", Ayoba, April 2020, doi**:** 10.1109/ICMCECS47690.2020.240873.
[10] P. Shiv, K. Goel, S. Ansari, and M. T. Kuwalekar, "Using A * algorithm to find shortest path in Indoor positioning system," *International Research Journal Engineering and Technology*, vol. 4, no. 6, pp. 2226–2228, 2017.
[11] D. Rachmawati1 and L. Gustin, " Analysis of Dijkstra's algorithm and A* algorithm in shortest path problem", *Journal of Physics: Conference Series 1566 (2020) 012061*, 2020, pp.1-7, doi:10.1088/1742-6596/1566/1/012061.
[12] L. Lin, C. Wu, and L. Ma, "A genetic algorithm for the fuzzy shortest path problem in a fuzzy network," *Complex and Intelligent System*, vol. 7, no. 1, pp. 225–234, 2021, doi: 10.1007/s40747-020-00195-8.
[13] Z. Yang, H. Xia, F.Su, J. Zhoa and F. Feng, " Application of genetic algorithm in modeling of shortest path problem", *Chinese Automation Congress*, 2021, pp. 3347-3450, doi: 10.1109/CAC51589.2020.9327269.
[14] G. Panchal and D. Panchal, "Solving NP hard Problems using Genetic Algorithm," *Int. J. Comput. Sci. Inf. Technol.*, vol. 6, no. 2, pp. 1824–1827, 2015.
[15] I. Aydin, M. Karakose, and E. Karakose, "A navigation and reservation based smart parking platform using genetic optimization for smart cities," *2017 5th International Istanbul Smart Grid and Cities Congress and Fair*, Istanbul, 2017, pp. 120–124, , doi: 10.1109/SGCF.2017.7947615.
[16] C. Soyleyici and S. B. Keser, "A hybrid genetic algorithm for mobile robot shortest path problem," International *Journal of Intelligent System and Applications in Engineering*, vol. 1, no. 4, pp. 16–19, 2016, doi: 10.18201/ijisae.2016SpecialIssue-146987.
[17] M. Akram, A. Habib, and J. C. R. Alcantud, "An optimization study based on Dijkstra algorithm for a network with trapezoidal picture fuzzy numbers," *Neural Computing and Application*, vol. 33, no. 4, pp. 1329–1342, 2021, doi: 10.1007/s00521-020-05034-y.
[18] G. Deepa, P. Kumar, A. Manimaran, K. Rajakumar, and V. Krishnamoorthy, "Dijkstra algorithm application: shortest distance between buildings," *International Journal of Engineering and Technolology*, vol. 7, no. 4.10, p. 974, 2018, doi: 10.14419/ijet.v7i4.10.26638.
[19] K. C. Ciesielski, A. X. Falcão, and P. A. V. Miranda, "Path-value functions for which Dijkstra's algorithm returns optimal mapping," *J. Math. Imag. Vis.*, vol. 60, no. 7, pp. 1025–1036, 2018, doi: 10.1007/s10851-018-0793-1.
[20] W. Cai, D. Zhang, and Y. Pan, "Implementation of smart parking guidance system based on parking lots sensors networks," *2015 IEEE 16th International Conference Communication Technology,* Hangzhou, 2016, pp. 419–424, doi: 10.1109/ICCT.2015.7399872.
[21] Y. Sun, W. Dong, and Y. Chen, "An improved routing algorithm based on ant colony optimization in wireless sensor networks," *IEEE Communication Letter,* vol. 21, no. 6, pp. 1317–1320, 2017, doi: 10.1109/LCOMM.2017.2672959.
[22] K. Sundaran, V. Ganapathy, and P. Sudhakara, "Energy efficient multi-event based data transmission using ant colony optimization in wireless sensor networks," 2*017 International Conference on Intelligent Computing, Instrumentation and Control Technologies,* Kerala, 2017, pp. 998–1004, doi: 10.1109/ICICICT1.2017.8342703.
[23] D. Fano Yela, F. Thalmann, V. Nicosia, D. Stowell, and M. Sandler, "Online visibility graphs: Encoding visibility in a binary search tree," *Physical Review Research*, vol. 2, no. 2, pp. 1–9, 2020, doi: 10.1103/PhysRevResearch.2.023069.
[24] P. Suresh, R. Sukumar, and S. Ayyasamy, "Efficient pattern matching algorithm for security and Binary Search Tree (BST) based memory system in Wireless Intrusion Detection System (WIDS)," *Computer Communication,* vol. 151, no. November 2019, pp. 111–118, 2020, doi: 10.1016/j.comcom.2019.11.035.
[25] A. A. Kazharov and V. M. Kureichik, "Ant colony optimization algorithms for solving transportation problems," *J. Comp. Sys. Scie. Int.,* vol. 49, no. 1, pp. 30–43, 2010**,** doi: 10.1134/S1064230710010053.

BIOGRAPHIES OF AUTHORS

Karimeh Ibrahim is a second-year PhD student in Computer and Communication department in University Putra Malaysia (UPM). Her main research interest centers on Artificial Intelligence and its applications in smart cities, Traffic flow prediction, Shortest Path Algorithm for Guidance System, Deep learning (Spatial-Temporal models) and Embedded system. She is also concerned about Artificial Intelligence in Healthcare. She finished her master degree in Smart guidance system for indoor parking system from UPM, Malaysia, 2019. She received her bachelor degree in Computer Engineering from Fahad Bin Sultan University (FBSU), Saudi Arabia, 2013.

Azura Che Soh received her B.Eng. degree and M.Sc. in electrical and electronics engineering from Universiti Putra Malaysia (UPM), Malaysia in 1998 and 2002, respectively. She received the PhD degree in electrical engineering (major in control engineering) from Universiti Teknologi Malaysia. Her research interests include control system, intelligent control system, system modelling, and system modelling and energy management system. Currently, she is an Associate Professor at Department of Electrical and Electronic Engineering, Faculty of Engineering, UPM, and Researcher at Control System & Signal Processing (CSSP) Research Centre, UPM.

Asnor Juraiza Ishak obtained her bachelor degree in Electrical-Mechatronic Engineering from University Technology Malaysia (UTM) and MSc in Control Automation System Engineering from University Putra Malaysia (UPM). She obtained her PhD in Electrical, Electronic & System Engineering from Universiti Kebangsaan Malaysia (UKM). She is a senior lecturer at the Department of Electrical and Electronic Engineering, UPM. Her research interest includes intelligent control system, control system design, pattern recognition, image & signal processing, biomedical engineering, system modelling, rehabilitation and assistive robotics.

Haslina Jaafar is a senior lecturer at the Department of Electrical and Electronic Engineering, Faculty of Engineering, University of Putra Malaysia (UPM). She received her BE in Electrical and Electronic Engineering and Systems and MS degree in microelectronics from National University of Malaysia (UKM) in 1999 and 2001, respectively, and her PhD degree in Microelectro-mechanical systems from the University Science Malaysia (USM) in 2014. Her current research interests include MEMS, flexible electronics and nanotechnology. She is a senior member of IEEE, active members of Electron Devices Society (EDS) and Circuits and Systems Society (CAS).