Finding Shortest Path of the Ambulance Routing: Interface of A* Algorithm using C# Programming

*Noraimi Azlin Mohd Nordin, *Zati Aqmar Zaharudin, *Mohd Azdi Maasar, *Nor Amalina Nordin,
*Department of Mathematics, Faculty of Computer and Mathematics Sciences, Universiti Teknologi MARA (UiTM),
40450 Shah Alam, Selangor, Malaysia
noraimi@tmsk.uitm.edu.my, zatiaqmar@tmsk.uitm.edu.my, azdimaasar@tmsk.uitm.edu.my, ring_ring86@yahoo.com

***Abstract*- In order to ensure the ambulance arrival to patient is within the targeted time, ambulance availability must be ensured and the time taken to arrive can be controlled. Optimal route that able to provide shortest distance must be determined. Thus, an interface is developed to help the ambulance providers in sending the ambulance to a specified ambulance station and emergency site. The interface for the ambulance routing is designed to give shortest path and shortest distance (in km). The results obtained were programmed using C# (pronounced as C sharp) software and the A* algorithm utilized as the engine to determine the shortest distance for the ambulance in the study area.**

Keywords- ambulance routing; A* algorithm; C# interface; Haversine formula

I. INTRODUCTION

The history of shortest path algorithm can be traced as early as 1968 in which the A* algorithm was established by Hart et al. [1], then, this algorithm is further discussed and extended by Nilsson in [2], Pohl in [3], Pearl in [4] and Patel in [5]. In 2006, study by Chen et al. [6] proved that A* algorithm faster and better than Dijkstra's algorithm because A* algorithm is an efficient algorithm to solve the shortest path. A* algorithm able to calculate the shortest path as Dijkstra's algorithm does. Lester [7] stated that A* algorithm essentially the same as Dijkstra's algorithm, distinctive by a heuristic that utilized in A* algorithm. Heuristic is defined as finding optimal solution. Heuristic is very fast, thus it can be easily adapted to various objectives such as minimizing total distance with a few changes in the constraint model [8]. Therefore, the A* algorithm able to compute faster than Dijkstra's algorithm.

As mentioned by Shukla et al. [9], before the entire computing algorithm can be applied, it was earlier shown that A* algorithm generates good result for the problem of path planning. It was also corroborated by Liang in [10], the A* algorithm can achieve better running time than other algorithm in the road network application. As reported by Taeg-Keun [11], A* algorithm is a kind of informed search and widely used in finding an optimal route such as travel distance, traffic condition and etc. This is because the location of starting and ending point are known. Usually, the A* algorithm is used in finding optimal path through a graph and it is central to many problems that includes route planning for a mobile robot [12]. This model then improved by Kala et al. [13], who reported that the problem of robotic path planning can be solved using A* algorithm. Meanwhile, according to Khantanapoka and Chinnasarn [14], A* algorithm can be used in increasing and searching for shortest distance from starting nodes to end nodes where the end nodes is optimal way. Thus, these studies will be a basis for our project located in Shah Alam, where we developed an interface in order to compute the shortest distance for the EMS ambulances by using A* algorithm.

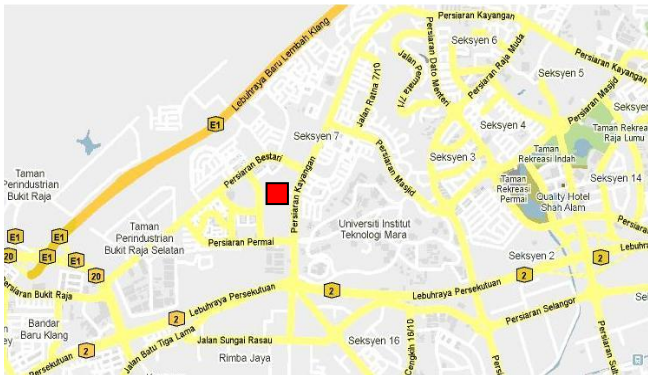
In our study area, Shah Alam, the road usually involved only a one way track. Ambulances in this area are facing many constraints such as traffic light, toll, and roundabout. If the ambulance has taken the wrong road, they will results to late arrival at the emergency sites. Therefore, this project focuses on developing an ambulance road network for the area under study specifically, Seksyen 7, Shah Alam, and to find a suitable mathematical model for estimating ambulance travelling distance. Thus, A* Algorithm model proposed by this study will be used to provide the route with the shortest total distance in our study area.

Therefore, two objectives are proposed in this paper, which are (i) to build an interface using C# software for finding shortest distance of the ambulances, and (ii) to find the minimum ambulance travelling distance by using (i). Previously, our study has implement the application of A* algorithm for ambulances at KKSA, therefore, as consecutive study, we compute the A* algorithm using the C# programming software.

II. METHODOLOGY

A. Data Collection

All data location and information related to the routing of ambulances are collected from the Klinik Kesihatan Shah Alam (KKSA). Based on the data collected, a road network is developed, where all major routes used to send patient to designated hospital are constructed in the developed road network. Hence, the A* algorithm is used to find routes that persist the shortest total distance; from KKSA to ambulance demand location. Figure 1 depicts the location of KKSA.

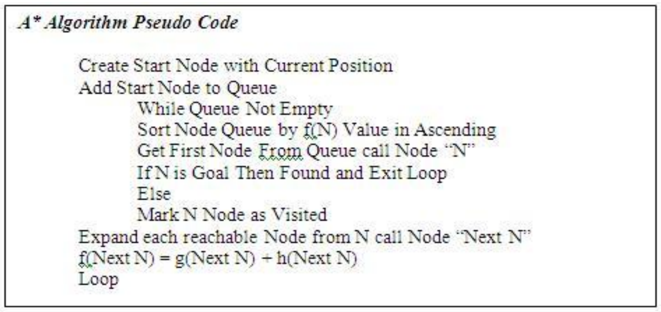


🟥	Klinik Kesihatan Shah Alam (KKSA)

Figure 1. Location of KKSA

B. A Algorithm Model*

A* algorithm pseudo code can be described as the following, based by Khantanapoka and Chinnasarn [14] as illustrates in Figure 2:



A Algorithm Pseudo Code*

```
Create Start Node with Current Position
Add Start Node to Queue
        While Queue Not Empty
        Sort Node Queue by f(N) Value in Ascending
        Get First Node From Queue call Node "N"
        If N is Goal Then Found and Exit Loop
        Else
        Mark N Node as Visited
Expand each reachable Node from N call Node "Next N"
f(Next N) = g(Next N) + h(Next N)
Loop
```

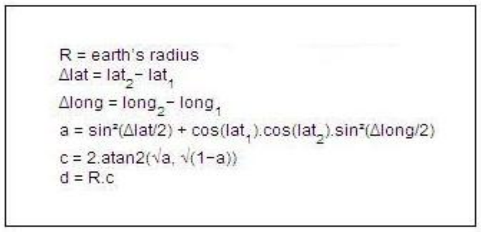
Figure 2: A* Algorithm Pseudo Code

The A* algorithm is applied to determine the shortest path from KKSA to specified incident site. The shortest path obtained is based on incident site and the nearest ambulance location. It works by maintaining a pair of lists, one containing locations on the tile map which *N* is the path and another one contains the locations. Then, A* algorithm will continues the loop if there is still next steps that could be consider to be available next step and considers its neighbors. Further details on A* algorithm implementations and workflows in area of study, can be found in [15].

Difficult parts in solving A* algorithm is to create a good heuristic function to determine *h(n)*. Castor [16] stated that heuristic function is used to find an estimate of how long it will take to reach the destination node from start node. A heuristic function differs from an algorithm where heuristic is more of estimation and is not necessarily provably correct. An algorithm is a set of steps which can be proven to stop on a particular given set of input. Heuristic function in A* algorithm is an arbitrary, where the better heuristic is, the faster and more accurate the solution will be. However, even deciding a good heuristic, the problem is still exists. Even with a shortest path example, the heuristic can change, depending on the implementation of the search, and how easy or complicated the heuristic function is going to be. Therefore, the heuristic used in this research is Haversine Formula.

C. Haversine Formula

Haversine is the name of trigonometric function. According to [17], to presuming a spherical Earth with radius, the locations of the two points in spherical coordinates are longitude and latitude. Sinnott [18] mentioned that in navigation, haversine formulas become an important equation using the great-circle distance between 2 point on a sphere from their longitudes and latitudes. This formula is used to calculate the distance between longitude and latitude in kilometres or miles. The haversine formula for calculating distance is as follows:



R = earth's radius
$\Delta lat = lat_2 - lat_1$
$\Delta long = long_2 - long_1$
$a = \sin^2(\Delta lat/2) + \cos(lat_1).\cos(lat_2).\sin^2(\Delta long/2)$
$c = 2.\text{atan2}(\sqrt{a}, \sqrt{(1-a)})$
$d = R.c$

Figure 3: Haversine Formula

where :

R	:	radius of the earth
lat	:	Latitude
long	:	Longitude
Δ*lat*	:	change in latitude
Δ *long*	:	change in longitude
a	:	the square of the half of the straight line distance between two points
c	:	the great circle distance in radians
d	:	Distance

This method gives exact results of the distance. According to Dave and Richard in [19], we take the radius, *R* equals to 6367 km and also equal to 3965 miles where this is the circumference divided by 2π. For converting decimal degrees to radians, we must multiply the number of degrees by π/180. To express *c* in decimal degrees, we multiply the number of radians by 180/π, but be sure to multiply the number of radians by *R* to get *d*. In addition to this, according to Long [20], the haversine distance is more accurate than other formulas in order to find the shortest distance. Thus, for our research, haversine formula will be utilized and incorporated into the Microsoft Visual C# 2008 Express Edition.

D. Microsoft Visual C# 2008 Express Edition

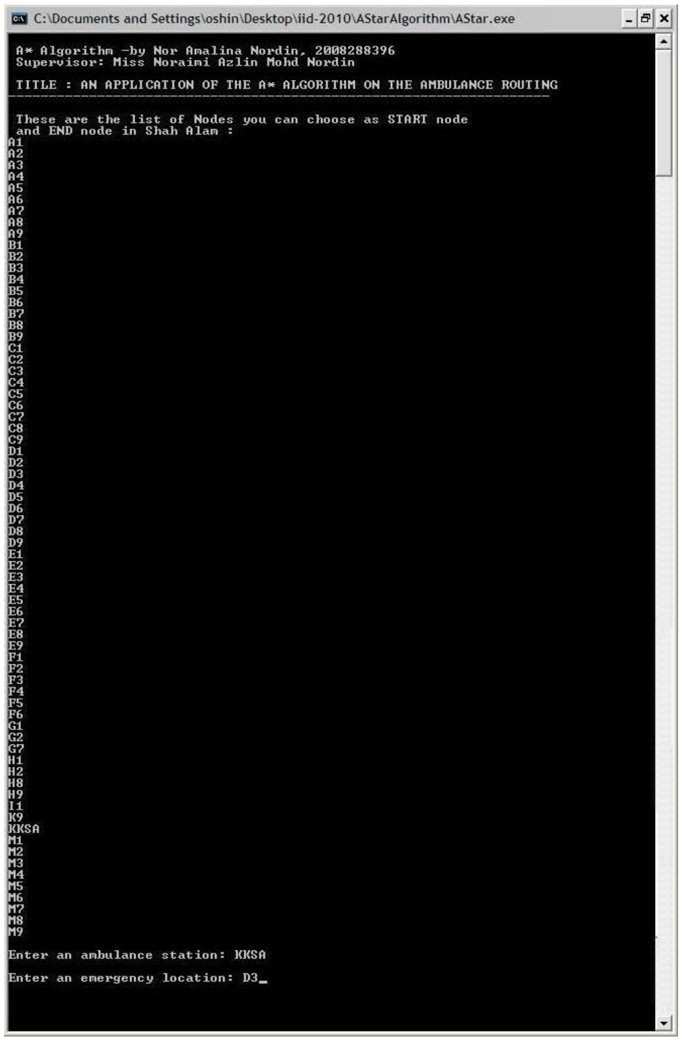
Microsoft Visual C# 2008 Express Edition or mainly known as C# software was created in Microsoft in the late

1990 and it was built by Anders Hejlsberg [21]. C# is directly related to C, C++ and Java. The reason of using C# is because it contains many innovative features than C++. Lippert [22] did wrote the base structure of the A* algorithm but did not provide a complete running sample. On the other hand, Macaferi's [23] implement coding from C# and proved that A* algorithm is really powerful algorithm compared to other algorithms. Further review on application of A* algorithm in C# programming can be seen in [16], [22], [20] and [18].

III. IMPLEMENTATION

Utilizing C# programming may reduce calculation time for the shortest distance compared to calculating manually. On top of that, this system provides information on the shortest distance and shortest path for an ambulance in order to arrive at the specified incident site.

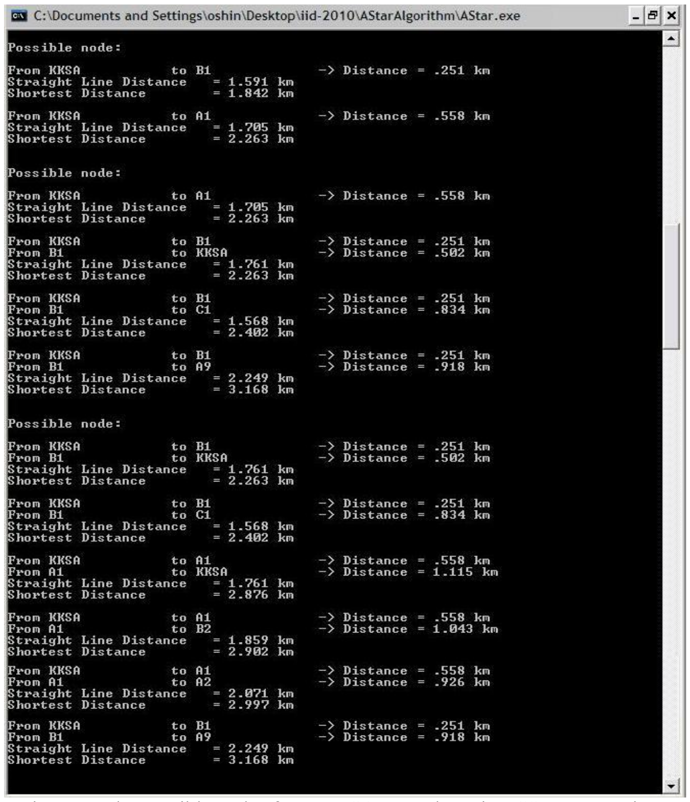
The implementation of the C# programming is as shown in the next Figure 4, 5, 6, 7 and 8. All the figures show the developed interface of the A* Algorithm in Microsoft Visual C# 2008 Express Edition from node KKSA to D3.



```
C:\Documents and Settings\oshin\Desktop\iid-2010\AStarAlgorithm\AStar.exe

 A* Algorithm -by Nor Amalina Nordin, 2008288396
 Supervisor: Miss Noraimi Azlin Mohd Nordin

 TITLE : AN APPLICATION OF THE A* ALGORITHM ON THE AMBULANCE ROUTING
 -----------------------------------------------------------------------

 These are the list of Nodes you can choose as START node
 and END node in Shah Alam :
A1
A2
A3
A4
A5
A6
A7
A8
A9
B1
B2
B3
B4
B5
B6
B7
B8
B9
C1
C2
C3
C4
C5
C6
C7
C8
C9
D1
D2
D3
D4
D5
D6
D7
D8
D9
E1
E2
E3
E4
E5
E6
E7
E8
E9
F1
F2
F3
F4
F5
F6
G1
G2
G7
H1
H2
H8
H9
I1
K9
KKSA
M1
M2
M3
M4
M5
M6
M7
M8
M9

Enter an ambulance station: KKSA

Enter an emergency location: D3_
```

Figure 4. The interface from KKSA to D3

Figure 4 shows the main interface of our program. Here, all the nodes involved are listed from A1 to M9. At the bottom of the interface, user needs to enter their designated starting and ending destination for the system to calculate the shortest distance and hence, provide the shortest path. From Figure 4, the user enters the start node as KKSA and end node as D3.



```
C:\Documents and Settings\oshin\Desktop\iid-2010\AStarAlgorithm\AStar.exe

Possible node:

From KKSA            to B1            -> Distance = .251 km
Straight Line Distance    = 1.591 km
Shortest Distance         = 1.842 km

From KKSA            to A1            -> Distance = .558 km
Straight Line Distance    = 1.705 km
Shortest Distance         = 2.263 km


Possible node:

From KKSA            to A1            -> Distance = .558 km
Straight Line Distance    = 1.705 km
Shortest Distance         = 2.263 km

From KKSA            to B1            -> Distance = .251 km
From B1              to KKSA          -> Distance = .502 km
Straight Line Distance    = 1.761 km
Shortest Distance         = 2.263 km

From KKSA            to B1            -> Distance = .251 km
From B1              to C1            -> Distance = .834 km
Straight Line Distance    = 1.568 km
Shortest Distance         = 2.402 km

From KKSA            to B1            -> Distance = .251 km
From B1              to A9            -> Distance = .918 km
Straight Line Distance    = 2.249 km
Shortest Distance         = 3.168 km


Possible node:

From KKSA            to B1            -> Distance = .251 km
From B1              to KKSA          -> Distance = .502 km
Straight Line Distance    = 1.761 km
Shortest Distance         = 2.263 km

From KKSA            to B1            -> Distance = .251 km
From B1              to C1            -> Distance = .834 km
Straight Line Distance    = 1.568 km
Shortest Distance         = 2.402 km

From KKSA            to A1            -> Distance = .558 km
From A1              to KKSA          -> Distance = 1.115 km
Straight Line Distance    = 1.761 km
Shortest Distance         = 2.876 km

From KKSA            to A1            -> Distance = .558 km
From A1              to B2            -> Distance = 1.043 km
Straight Line Distance    = 1.859 km
Shortest Distance         = 2.902 km

From KKSA            to A1            -> Distance = .558 km
From A1              to A2            -> Distance = .926 km
Straight Line Distance    = 2.071 km
Shortest Distance         = 2.997 km

From KKSA            to B1            -> Distance = .251 km
From B1              to A9            -> Distance = .918 km
Straight Line Distance    = 2.249 km
Shortest Distance         = 3.168 km
```

Figure 5. The possible nodes from KKSA to D3 by using C# programming

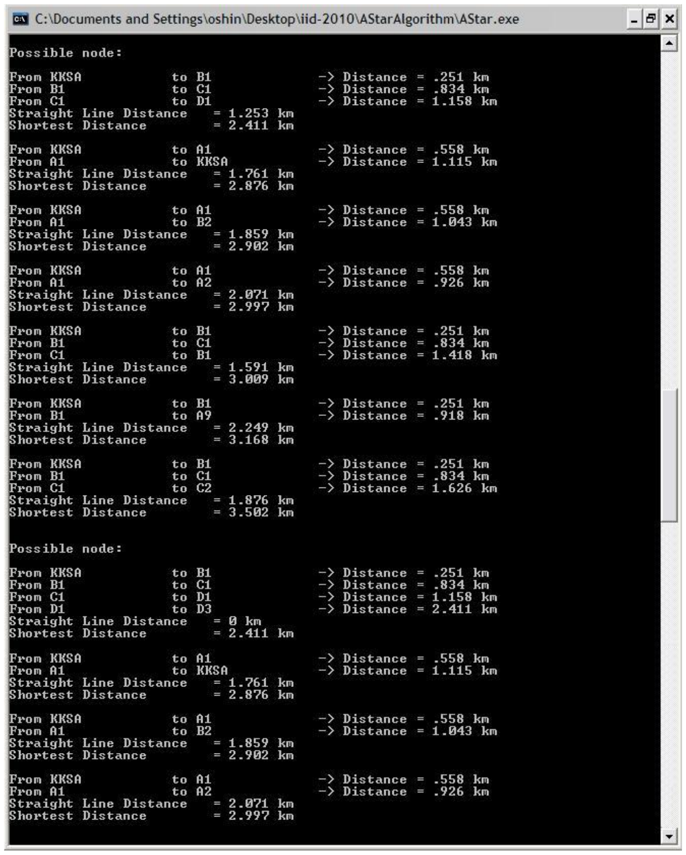
Calculation parts in our C# programming are shown thoroughly in Figure 5. Here, the word 'Possible Node' can be seen. Possible node means it will actually calculate the possible nodes which is the nearest to the starting node, that is KKSA. Thus, the nearest possible nodes from KKSA are B1 and A1. According to [15], hence, our *g(n)* is "Distance", *h(n)* is " Straight Line Distance" while *f(n)* is "shortest Distance". From the first possible node, we can see the shortest distance within KKSA to B1 and KKSA to A1. Therefore, we choose KKSA to B1 because it contains the smallest shortest distance (1.842 km).

The same procedure will be followed in finding the next shortest path. Thus, from figure 5 we can see that this program calculate all possible nodes from B1 and A1. The program calculates only from B1 to C1 and A9. This is due to the reason of there are no other possible nodes that can be obtained from A1. Hence, it will only calculate all the possible paths from B1, which means C1 and A9. Therefore, we choose B1 to C1 because it provides the shortest distance (2.842 km).

Continuation of calculation for finding the shortest path is shown in Figure 6, 7 and 8 respectively. Figure 6 illustrates the continuation of our calculation path of all possible nodes in the interface. Here, we can see that the fourth and fifth possible nodes that may be chosen as the next shortest path. In this figure, the nodes calculated are between D1 and C2. Since D1 has shorter distance value, thus D1 is chosen as our next best shortest path with distance of 2.411 km.

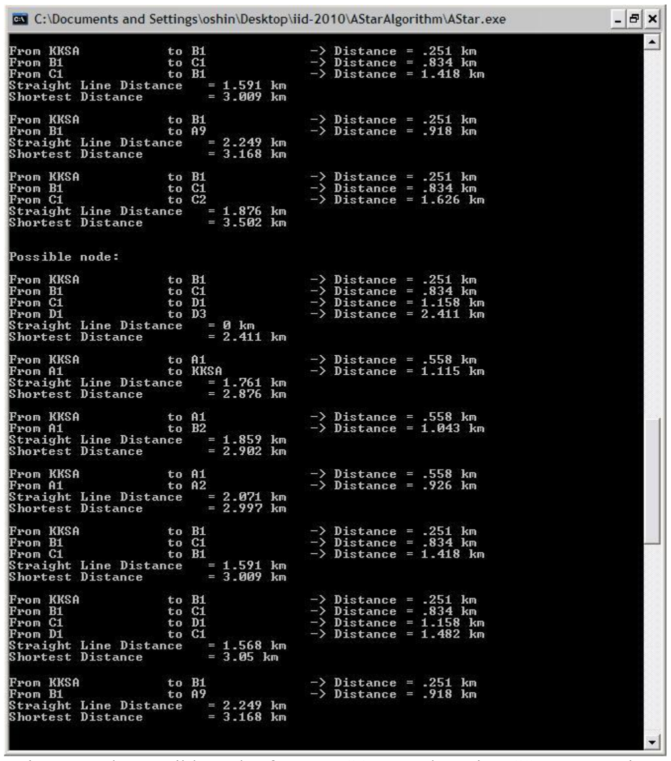
Since D3 is our final node or better known as destination node, thus the program will stop calculating and directly gives the final value of the shortest distance in Figure 7.

At the end of the calculation, the program will provide the shortest distance obtained as in Figure 8. This program shows the possible nodes for the shortest distance with their possible nodes and distance in kilometers based on A* algorithm. To end with, the program will ask user either the user want to try other routes by the word "Do you want to try A* Algorithm again? ". If the user wanted to find other shortest node, they can choose "Yes" and the program will start the main interface again, but if they refuse to continue, they may choose "No" and the program will automatically stop and terminated.



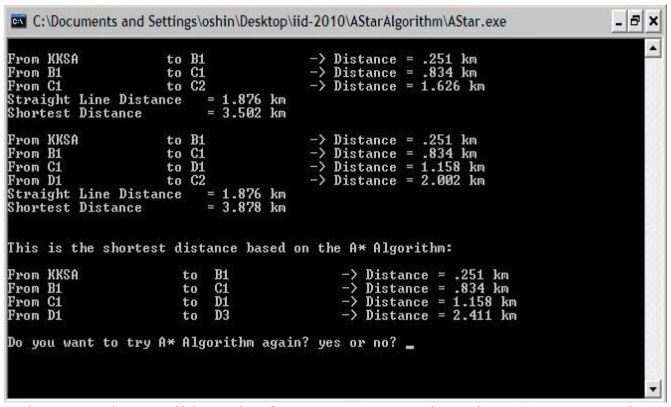
```
C:\Documents and Settings\oshin\Desktop\iid-2010\AStarAlgorithm\AStar.exe

Possible node:

From KKSA           to B1              -> Distance = .251 km
From B1             to C1              -> Distance = .834 km
From C1             to D1              -> Distance = 1.158 km
Straight Line Distance    = 1.253 km
Shortest Distance         = 2.411 km

From KKSA           to A1              -> Distance = .558 km
From A1             to KKSA            -> Distance = 1.115 km
Straight Line Distance    = 1.761 km
Shortest Distance         = 2.876 km

From KKSA           to A1              -> Distance = .558 km
From A1             to B2              -> Distance = 1.043 km
Straight Line Distance    = 1.859 km
Shortest Distance         = 2.902 km

From KKSA           to A1              -> Distance = .558 km
From A1             to A2              -> Distance = .926 km
Straight Line Distance    = 2.071 km
Shortest Distance         = 2.997 km

From KKSA           to B1              -> Distance = .251 km
From B1             to C1              -> Distance = .834 km
From C1             to B1              -> Distance = 1.418 km
Straight Line Distance    = 1.591 km
Shortest Distance         = 3.009 km

From KKSA           to B1              -> Distance = .251 km
From B1             to A9              -> Distance = .918 km
Straight Line Distance    = 2.249 km
Shortest Distance         = 3.168 km

From KKSA           to B1              -> Distance = .251 km
From B1             to C1              -> Distance = .834 km
From C1             to C2              -> Distance = 1.626 km
Straight Line Distance    = 1.876 km
Shortest Distance         = 3.502 km


Possible node:

From KKSA           to B1              -> Distance = .251 km
From B1             to C1              -> Distance = .834 km
From C1             to D1              -> Distance = 1.158 km
From D1             to D3              -> Distance = 2.411 km
Straight Line Distance    = 0 km
Shortest Distance         = 2.411 km

From KKSA           to A1              -> Distance = .558 km
From A1             to KKSA            -> Distance = 1.115 km
Straight Line Distance    = 1.761 km
Shortest Distance         = 2.876 km

From KKSA           to A1              -> Distance = .558 km
From A1             to B2              -> Distance = 1.043 km
Straight Line Distance    = 1.859 km
Shortest Distance         = 2.902 km

From KKSA           to A1              -> Distance = .558 km
From A1             to A2              -> Distance = .926 km
Straight Line Distance    = 2.071 km
Shortest Distance         = 2.997 km
```

Figure 6. The possible nodes from KKSA to D3 by using C# programming (continue)



```
C:\Documents and Settings\oshin\Desktop\iid-2010\AStarAlgorithm\AStar.exe

From KKSA           to B1              -> Distance = .251 km
From B1             to C1              -> Distance = .834 km
From C1             to B1              -> Distance = 1.418 km
Straight Line Distance    = 1.591 km
Shortest Distance         = 3.009 km

From KKSA           to B1              -> Distance = .251 km
From B1             to A9              -> Distance = .918 km
Straight Line Distance    = 2.249 km
Shortest Distance         = 3.168 km

From KKSA           to B1              -> Distance = .251 km
From B1             to C1              -> Distance = .834 km
From C1             to C2              -> Distance = 1.626 km
Straight Line Distance    = 1.876 km
Shortest Distance         = 3.502 km


Possible node:

From KKSA           to B1              -> Distance = .251 km
From B1             to C1              -> Distance = .834 km
From C1             to D1              -> Distance = 1.158 km
From D1             to D3              -> Distance = 2.411 km
Straight Line Distance    = 0 km
Shortest Distance         = 2.411 km

From KKSA           to A1              -> Distance = .558 km
From A1             to KKSA            -> Distance = 1.115 km
Straight Line Distance    = 1.761 km
Shortest Distance         = 2.876 km

From KKSA           to A1              -> Distance = .558 km
From A1             to B2              -> Distance = 1.043 km
Straight Line Distance    = 1.859 km
Shortest Distance         = 2.902 km

From KKSA           to A1              -> Distance = .558 km
From A1             to A2              -> Distance = .926 km
Straight Line Distance    = 2.071 km
Shortest Distance         = 2.997 km

From KKSA           to B1              -> Distance = .251 km
From B1             to C1              -> Distance = .834 km
From C1             to B1              -> Distance = 1.418 km
Straight Line Distance    = 1.591 km
Shortest Distance         = 3.009 km

From KKSA           to B1              -> Distance = .251 km
From B1             to C1              -> Distance = .834 km
From C1             to D1              -> Distance = 1.158 km
From D1             to C1              -> Distance = 1.482 km
Straight Line Distance    = 1.568 km
Shortest Distance         = 3.05 km

From KKSA           to B1              -> Distance = .251 km
From B1             to A9              -> Distance = .918 km
Straight Line Distance    = 2.249 km
Shortest Distance         = 3.168 km
```

Figure 7. The possible nodes from KKSA to D3 by using C# programming (continue)



```
C:\Documents and Settings\oshin\Desktop\iid-2010\AStarAlgorithm\AStar.exe

From KKSA           to B1              -> Distance = .251 km
From B1             to C1              -> Distance = .834 km
From C1             to C2              -> Distance = 1.626 km
Straight Line Distance    = 1.876 km
Shortest Distance         = 3.502 km

From KKSA           to B1              -> Distance = .251 km
From B1             to C1              -> Distance = .834 km
From C1             to D1              -> Distance = 1.158 km
From D1             to C2              -> Distance = 2.002 km
Straight Line Distance    = 1.876 km
Shortest Distance         = 3.878 km

This is the shortest distance based on the A* Algorithm:

From KKSA              to  B1              -> Distance = .251 km
From B1                to  C1              -> Distance = .834 km
From C1                to  D1              -> Distance = 1.158 km
From D1                to  D3              -> Distance = 2.411 km

Do you want to try A* Algorithm again? yes or no? _
```

Figure 8. The possible nodes from KKSA to D3 by using C# programming (continue)

Thus, as shown in the Figure 4, 5, 6, 7 and 8, this is the results obtained after the data is computed into C# Programming. By using C# programming, the result shows that the shortest distance from KKSA to D3 is achieved via path KKSA → B1 → C1 → D1 → D3, with the total distance of 2.411 kilometers. The value of the distance is a little bit different compared to manual calculation since in C# Programming, the longitude and latitude were used to determine the distance for each node and straight line distance.

Moreover, under this C# Programming, the ambulance starting node is not limited to starting node at ambulance station or KKSA, but it can also start from any other nodes keyed in by the users.

IV. Conclusion

As a conclusion, this application of A* algorithm with C# programming proposed system aims to enhance the quality of the EMS. In addition, it may help to improve the efficiency of the EMS ambulance management. This system can provide information on the shortest distance and shortest path for an ambulance in order to arrive on time to the incident site. Moreover, the ambulance will be send from any wanted starting node (the ambulance location) to a specified destination node (incident site).

IV. Acknowledgment

The authors would like to thank everyone involved directly and indirectly throughout completion of this paper. We would also like to thank Assoc. Prof. Dr. Adibah Shuib for her comments regarding this project.

References

[1] P. E. Hart, N. J. Nilsson, & B. Raphael, "A Formal Basis for the Heuristic. Determination of Minimum Cost Paths," IEEE Transactions on Systems Science and Cybernetics SSC4 4 (2), 1968, pp. 100–107.

[2] J. N. Nilsson, Problem-Solving Methods in Artificial Intelligence. New York: McGraw Hill, 1971.

[3] I. Pohl, "Heuristic Search Viewed as Path Finding in a Graph. Artif.Intell, vol. 1, 1970, pp. 193-204.

[4] J. Pearl, Heuristic Intelligence Search Strategies for Computer Problem Solving. Reading, Ma: Addison-Wesley, 1984.

[5] A. Patel, (2009). *A* Algorithm*. Retrieved from http://www-cs-students. stanford. edu /~ amitp/

[6] Q., Chen, X., Fei & W., Li, (2006). A New Shortest Path Algorithm based on Heuristic Strategy, Proceedings of the 6th World Congress on Intelligent Control and Automation, June 21-23, 2006, Dalian, China. Corporatise Ambulance Services, says MMA. (2006, September 28). Retrieved from http://malaysianmedicine. blogspot.com/2006_09_01_archive.html

[7] P., Lester, (2005). A* Pathfinding for Beginners. Retrieved from http://www. polic yalmanac.org/games/aStarTutorial.htm

[8] Y. Caseau, & F. Laburthe, "Heuristic for Large Constrained Vehicle Routing Problems," Journal of Heuristics, vol. 5, 1999, pp. 281-303

[9] A., Shukla, R., Tiwari, & R., Kala, (2008). Mobile robot navigation control in moving obstacle environment using A* algorithm. Intelligence System engineering systems through artificial neural networks ASME Publications, 18, 113-120.

[10] D., Liang (2005). Fast Shortest Path Algorithm for Road Network and Implementation Honour Projects Carleton University, 5-13.

[11] W., Taeg-Keun, (2007). Efficient Modified Bidirectional A* Algorithm for Optimal Route-Finding*. Dept. of Computer Science, Kyungwon University Seongnam-Si, Gyeonggi-Do, Korea. Retrieved from http://www.springerlink.com/content/w68p612q2m155413/?p=24f1169cafa9410a88f326a7e1e1741d&pi=0

[12] X., Yanyan, Y., Weiya, & S., Kaile, (2009). The BDD-Based Dynamic A* Algorithm for Real-Time Replanning*. Supported by the Natural Science Foundation of China. Retrieved from http://www.springerlink.com/ content/n0561782w7008m03/fulltext.pdf

[13] R., Kala, A., Shukla, & R., Tiwari, (2010). Fussion of Probabilistic A* Algorithm and Fuzzy Inference System for Robotic Path Planning. Indian Institute Technology and Management Gwalior, Gwalior, India.

[14] K., Khantanapoka, & K., Chinnasarn, (2009). Pathfinding of 2D & 3D Game Real- Time Strategy with Depth Direction A* Algorithm for Multi-Layer. 2009 Eighth International Symposium on Natural Language Processing. Bangkok, Thailand.

[15] N. A., Mohd Nordin, N., Kadir, Z., A., Zaharudin, & N., A., Nordin, (2011). An Application of the A* Algorithm on the Ambulance Routing. Proceedings of the 2011 IEEE Colloquium on Humanities, Science and Engineering Research (CHUSER 2011), 5 & 6 December 2011, Hotel Parkroyal, Penang.

[16] T., Castor, (2007). *A* algorithm implementation in C#*. Retrieved 15 January 2010, from http:// www . code project. com/ KB/ recipes/ Path Finder. aspx

[17] R., W., Sinnott, (1984). Virtues of the Haversine. *Sky and Telescope*, 68(2), 159.

[18] C., Sarath, (2010). *Calculate distance between two points on globe from latitude and Longitude coordinates*. Retrieved 25 July 2010, from http://www.consultsarath.com/content/artIcles/KB000010-calculate-distance-between-twopoints-on-the-globe-from -latitudecoordinate.aspx

[19] P., Dave, & P., Richard, (1999). Deriving the Haversine Formula. Retrieved 25 July 2010, from http://mathforum.org/dr.math/

[20] S., Long, (2008). Haversine Formula in C# and in SQL. Retrieved 17 July 2010, from http:// megacode3. Wordpress .com/2008 /02/05/ haversine-formula-in-c/

[21] S., Herbert, (2001). *C#: A beginner guide*. Obsone:McGraw Hill.

[22] E., Lippert, (2007). *Path Finding Using A* in C# 3.0*. Retrieved 17 July 2010, from http:// blogs. msdn.com/b/ /ericlippert /achieve /2007 /10/02/ path-finding-using-a-in-c-3-0.aspx

[23] L., Macaferi's, (2009). A* Pathfinding in C#-Part 1, Part 2 and Part 3. Retrieved 17 July 2010, from http://www.leniel.net/2009/06/astar-pathfinding-search-in-csharp.html