A Comparative Study of the Algorithms for Path finding to Determine the Adversary Path in Physical Protection System of Nuclear Facilities

D. Andiwijayakusuma[1], A. Mardhi[1], I. Savitri[2], T. Asmoro[2]
[1]Center for Nuclear Reactor Technology and Safety – BATAN
[2]Center for Informatic and Nuclear Strategic Zone Utilization - BATAN

Nuclear Serpong Area, PUSPIPTEK complex Building 80th, South-Tangerang, Banten, INDONESIA.

Corresponding author’s e-mail address: dinan@batan.go.id

Abstract. The shortest path algorithm is the process of finding a path on a weighted graph that minimizes the number of the weights of the path-forming side. Thus, the resulting path is the distance that has the least weight or distance. In physical protection systems (PPS), the implementation of the shortest path algorithm is determining the most vulnerable adversary path to penetrate the physical protection system of a nuclear facility. The paths between points at a facility passed by the adversary will form a directed and weighted graph. The weights on the graph represent the ability of a protective device (e.g. CCTV, sensor, alarm etc.). The formed graph will be processed using algorithms to determine the most vulnerable path. In this paper, we study and examine the performance of two types of algorithms namely Dijkstra and A*. The purpose of this study is to compare the two algorithms which we will use it to determine the most vulnerable path related to the physical protection system of nuclear facilities. Our results show that the A* algorithm provides more efficient path results than the Dijkstra algorithm.

Introduction

The Physical Protection System (PPS) plays a role in ensuring the security of any nuclear facilities from attempts to theft, sabotage and other illegal action. One way to find out the effectiveness of PPS design is by conducting an evaluation through enemy path analysis. The most vulnerable path is defined as the best path for the adversary to penetrate the system. If the evaluation of the effectiveness of PPS design reveals any vulnerability, the system design must be redesigned to correct the vulnerabilities and another analysis of the redesigned system is performed [1]. Pathfinding is the process of finding the best path or shortest path from one point to another. Shortest Path is an optimization effort of pathfinding. Any route found on pathfinding will be searched for the shortest route. The shortest route search is determined by the accumulated vector quantity to reach the destination node. The accumulated cost that has the minimum value is the shortest route from the graph. There are many algorithms that can be used to find the shortest path, the most used is Dijkstra's algorithm [2]. Other algorithms include the Greedy algorithm, the A* (A-Star) algorithm[3], the Genetic algorithm, the Bellman-Ford algorithm, the Flyod Warshall algorithm, the Ant Colony algorithm, the Branch and Bound algorithm, the Prim algorithm and others. Each algorithm can be
implemented with various programming languages so that the shortest distance calculation can be done. But the problem is: if the selected Dijkstra is the algorithm, then this algorithm has correctly determined the shortest distance that can be done? For these purposes Dijkstra's algorithm will compare the result with another algorithm to calculate the shortest path. In this paper we examine two algorithm, they are dijkstra and A* algorithm. We will be described the characteristics of those two algorithms. The results of this study will be used to determine the most vulnerable path related to the physical protection system of nuclear facilities.

Dijkstra’s Algorithm
Dijkstra’s Algorithm is one of the most famous algorithms in computer science. Back before computers were a thing, around 1956, Edsger Dijkstra came up with a way to find the shortest path within a graph whose edges were all non-negative value. To this day, almost 50 years later, his algorithm is still being used in things such as “link-state routing”. Dijkstra's algorithm implements the graph theory. Therefore, Dijkstra's algorithm can be explained by using node symbol (vertex, node) with edge or arc, and since this algorithm uses directed graph with certain weights, it is called network algorithm. The network weight in this case is the distance and which is then used to determine and calculate the shortest path. Network weights can be replaced there is a probability of being detected by security devices, i.e. CCTV, sensors, alarms and other weights.

The sequence of Dijkstra's algorithm steps is as follows as from reference[4]:
1. Give a weighted value (for example: distance, probability of detection by sensor, etc.) from one node to another node, then specify value 0 at the beginning node and infinite value on another node or unfilled node.
2. Determine all unassigned nodes and set the starting node as the departure node.
3. From the departure node, consider the neighbor's unspoiled nodes and calculate the weighted value from the departure node. In this 3rd sequence, for example, the node A to node B has a weighted value of 5 and from node B to node C is 7, then the total weighted value to node C passes node B is 5 + 7 = 12. If this weighted value is smaller than previously recorded, delete the old data and save the value to a new weighted value.
4. When you are finished considering each weighted value to the neighboring node, mark the node that has been touchable as the touchable node. The touchable node will never be re-checked, and the saved value is the last and the least weighted value.
5. Determine the unspoiled node with the smallest distance from the departure node, as the next departure node, and continue with the return to 3.

A* Algorithm
The A-Star or A* algorithm is one of the search algorithms that analyze inputs, evaluates a number of possible paths and generates solutions. The A* algorithm is a computer algorithm that is used extensively in graph traversal and path finding and efficient path planning processes around points called nodes (Reddy, 2013). The A * algorithm uses the lowest cost path to the node that makes it the best first search algorithm. Using the formula as follows:

\[ f(x) = g(x) + h(x) \]  \hspace{1cm} (I)

where:
- \( g(x) \) is the total distance from the original position to the current location.
- \( h(x) \) is a heuristic function used to estimate the distance from the current location to the destination location.

The heuristic function is clearly different because it is a mere estimate compared to its original value. The higher the heuristic accuracy, the faster and better the destination location is found and with a better level of accuracy. The function \( f(x) = g(x) + h(x) \) is the current estimate from the closest distance to the destination.
Adversary Path Representation
Here is an example of a sabotage scenario with the adversary path to penetrate the hypothetical nuclear facility.

In Figure 1 shown that adversary can move nodes in one area but must remain through each area to reach the target, which means it is a sequential diagram. This concept is same as in Adversary Sequence Diagram (ASD) but the network diagram may represent the connection better with some of advantage in find the most vulnerable path and determine the critical point detection [6]. The vertices represent the physical location such as limited area, protected area, outer door and wall, or the adversary milestone such as start point and target. The existence of an edge in between two vertices may represent the connection between two physical locations or method of adversary action such as penetrate fence and outer door. The safeguard elements properties such as probability of detection in sensors and delay time of delay elements represented by a number of network capacity or weight value.
in the edge. The best path analysis for the enemy, which means the most vulnerable path for the security system, will be calculated by shortest path algorithms.

**Results and Discussions**

Figure 1 shows a hypothetical nuclear facility. The facility is divided into 6 areas: Outer Area, Limited Area, Protected Area, Controlled Building Area, Controlled Building, and Target. The adversary needs 10 steps and must enter every area of the facility. We try to simplify the diagram into 5 areas by being assigned several nodes as an alternative path that will be selected as the best path for the adversary to penetrate the security system.

![Diagram of the hypothetical facility](image1)

Figure 1. Hypothetical nuclear facility.

We modelled the map in a two-dimensional grid representing the area of the hypothetical facility. In this modelling does not consider the detection sensor tool for weighting values. In map modelling there appears to be a gap such as in a maze, assuming that the gap is the point with the lowest weighted value. Next is a simulation to find the shortest path the enemy can pass to reach the target using Dijkstra and A* algorithms.

![Simulation models using computer code](image2)

Figure 3. Computer modeling with a sabotage scenario on a hypothetical facility.

![Pathfinding results using Dijkstra Algorithm](image3)

![Pathfinding results using A* Algorithm](image4)

Figure 4. Pathfinding results using Dijkstra and A* algorithm.

From Figure 4 above, it appears that the adversary path using the A-star algorithm is slightly different from the adversary path using Dijkstra's algorithm. The simulation to find the shortest path is performed ten times for each algorithm and the average simulation results are shown in Table 1. The addition of barrier weight in the form of detection of sensor or CCTV camera has not been applied in the simulation. The gap in the barrier is assumed to be the weakest point and allows the enemy to pass through the gap to the target.
Table 1. The results of simulation calculations using Dijkstra's Algorithm and A* Algorithm.

| Algorithm         | Ave. Length | Ave. Computer Time(ms) | Ave. Operations |
|-------------------|-------------|------------------------|-----------------|
| Dijkstra          | 67.46       | 18900                  | 1829            |
| Dijkstra Bi-      | 67.46       | 26500                  | 628             |
| directional       |             |                        |                 |
| A*                | 68.04       | 10000                  | 1036            |
| A* Bi-directional | 68.04       | 9100                   | 452             |

The results in Table 1 show that the average path length using the Dijkstra algorithm is relatively the same as using the A-Star algorithm. The average difference in track length is less than 1%, either using the general Dijkstra/A* or bi-directional. However, the required operation using the Dijkstra algorithm is relatively greater than using the A* algorithm. The difference occurring to the required operation is approximately 43.35% in the general Dijkstra/A* and about 28.03% in Dijkstra/A* bi-directional. This happens because of the heuristic function that affects the speed of the shortest path search process. The more accurate the value of the heuristic function, the faster the process of finding the shortest path to the target. The computation time used in the simulation shows that the A* algorithm is faster determining the shortest path, either A* regular or bi-directional. This is of course because the operation required is less than the Dijkstra algorithm. Overall, we can conclude that the simulation to find the shortest path using the A* algorithm is more effective than using Dijkstra's algorithm.

Conclusion
In this study, we have studied about two shortest path algorithms and their comparison. There are advantage and disadvantage in algorithms. To find the running time of each algorithm, we used one code for comparing the length, running time (in Microseconds) and operations. After running the same program on ten different runs, we calculated the average length, average running time and operations for each algorithm. From the results, we can conclude A* algorithm is more efficient algorithm to find out the shortest path compare to Dijkstra Algorithm.

Acknowledgments
Authors wishing to thanks to Center for Nuclear Reactor Technology and Safety - BATAN, and Indonesian Ministry of Research, Technology and High Education (KEMENRISTEKDIKTI) FY 2018 for financial support through the research.

References
[1] A.A. Wadoud, A.S. Adail, A.A. Saleh, Physical protection evaluation process for nuclear facility via sabotage scenarios, Alexandria Engineering Journal, 2017.
[2] Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik 1, 269–271 (1959).
[3] Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics SSC4 4(2), 100–107 (1968).
[4] D. Dwijono, Comparison of Shortest Distance Calculation Between Dijkstra’s Algorithm and Linear programming, Jurnal Teknik dan Ilmu Komputer, Vol. 06 No. 24, Okt-Des 2017.
[5] Terao, N., Suzuki, M., A Probabilistic Extension of the EASI Model, Journal of Physical Security, vol.7, issue 2, 2014, p.12-29
[6] A. Mardhi, Master thesis: Development of Analytical Software for Assessing Physical Protection System, Dept. of Nuclear Engineering, Chulalongkorn University, 2015.