Weight Value and Map Complexity in Theta*

Phuc T.H. Le and Ki-dong Lee

Robot lab, Department of Computer Engineering, Yeungnam University, Gyeongsan-si, South Korea

Abstract. A performance of an experiment to analyze the relationship between weight value and map complexity. That means there is a trial to find which weight value performs the best on maps with different complexity. And then in this article there is an application of hierarchical and dynamic weight method into Theta* path-finding algorithm. The dynamic weight value is picked from the first experiment. For different abstract map, a weight value is changed base on the complexity of the abstract map. This article contains a re-test of new implementation of Theta* and compare to the Mr. Nash's Theta* to collect the positive outcome. And by applying the dynamic weight method with different weight based on map complexity, the dynamic weight theta* perform better in time costing

1. Introduction

Path-finding algorithm is a process of finding a road between two given points – start and destination ones - in a random environment. In this study field, people used this technique in many other life fields such as: robotic, game design and network. In the past, path-finding algorithm had to solve two issues: find a way between two points in a map and the optimal shortest path. In the current day, people create one more problem: calculate an accurate path with in the short time. There are several path-finding algorithms have been introduced recently [1]. For example, Deep-First search, Breadth-First search, Best-First search, Dijkstra, A* and the most recent path-finding algorithm – Theta* [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 and 15]. While some of the algorithms search for the optimal path, others find a path within the shortest time. There are many implementing methods to reduce the time consuming such as using different types for database structures [6, 7, 8, 9, 10, 11 and 12]. In this paper, there is an application of dynamic weight to boost the speed of Theta* [3, 4 and 14]. But the found weight value will related to the map complexity of the environment. In order to complete our experiment, the experiment process firstly run different weight values on different maps with size 10x10 with specified maps' complexity. The weight value which gives a best outcome will be the value that is applied for typical map abstracts. Before us, there is no research on the relationship between weight value and map complexity.

2. Related Works

Dijkstra (DA) was invented by Edsger Dijkstra. By giving a start (source) point, DA can search a shortest path in the map [1]. DS is a type of path-finding algorithm that applies the comparison condition to find the path. However, the high cost of computation time is a weighted issue of Dijkstra.

A* is a search algorithm which is expanded from Dijkstra by applying a heuristic value [3]. The found path seems to be made by the non-human object [4]. The formula of A* is the formula (1)

$$F = G + H \quad (1)$$

where G is the distance from the start point to the current point and H is the estimated distance from the current point to the goal point.

Theta* - post-smoothing process of the A* algorithm - was created by Alex Nash in 2007 [4]. The problem of Theta* is that computation time is more than A* but the path is not always guaranteed to be the optimal one [14].

Weighted Theta* is a variant of Theta* algorithm [13]. In this algorithm, a new parameter called weight will be added to Theta*'s F formula (2)

$$F = G + W * H \quad (2)$$

, where G and H is the same as A* in formula (1), and W is a weight value ($0<W<+\infty$).

HPA* stands for hierarchical path-finding A* algorithm [15, 16]. This method reduces the complexity of the map by dividing the map into the smaller abstract maps. Botea et.al proved that HPA* saves up to 10 times faster than A* but the increasing in path-length is within 1% [11]. But the path from HPA* is similar to A* which looks unnatural.

In 2015, there is a research article which applied the dynamic weight and HPA* method into Theta* to reduce the time consuming [15].

However on these previous researches, there is no article defining a specific weight value on each type of map. Therefore, a performance of an analysis on the Theta* determines the relationship between weight value and map complexity. From the collected results, a conclusion of a new version of Theta* which can find the path in faster way comparing to the original Theta* is drawn.

3. Specific Weight - Dynamic Weight Theta*

The main idea of this article is to specify which weight value is good for a specific type of map. And when the abstract map matches to that type of map, then corresponding weight value is applied to that map. Moreover, a run on the dynamic weight theta* on different sizes map is made. The big map is divided into smaller abstract maps with size 10 points x10 points. Then for each abstract map, a scan is made for the map's environment to detect its complexity and the use of statistic data to determine the applied weight value for that map area. By doing this, the value is determined the best for a specific type of map. Then a performance is made on the dynamic weight Theta* in 12 cases.

In this paper, an application is created that user can pick a map containing obstacle points, a start point and goal point. The app is written in Java and tested on Intel Core 2 with 2GB Ram. In the app, a class in Java called Node, which has some properties: x, y represents the longitude and latitude position of the point in the grid map, color to present the status of the map such as: obstacle, goal, start, and path. In order to make these points are easier to see, there is an assignment of a certain size for a point (30 pixels). All of these points will be combined into a square 2D map size n. As in Fig. 1, the blue point has an arrow with the word "goal/ destination point" is a destination point. The green point has an arrow with the word "start point" is a start point. These points with white color are the free points that can be passed through. And the rest points in the map are these obstacles that can't be passed. This map is a point map type. And when the path is found by the application it is displayed a java window frame. On the result window, the found path is connected with series lines. All these points are belonged to the path found by the Theta* algorithm. At each case, two pieces of information ae collected: the computation time of the process from the beginning of the process to the end of the process in milliseconds, and the path-length of the solution path, which is count by the sum of each path point's Euclidean distance to their parents.

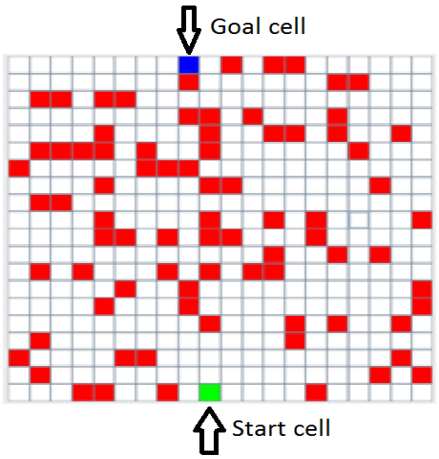

Figure 1. Example of Map

3.1 Weight on Specific Map Type

The implemented dynamic weight Theta* is called as "DW Theta*." And a base for comparison is the original Theta*, "Author Theta*" from Nash et.al. [3, 4 and 14] On general, 30 different maps are used to evaluate the weight value. The main purpose of these activities is to figure out which what weighted value makes the algorithm perform the fastest in speed in which type of map complexity. In this article, the time unit is millisecond. At the beginning of our experiment, the "DW Theta*" runs on 6 cases. In all the six cases the map sizes are always 10 points x 10 points, but the percentages of obstacle are increased from 0% to 50%. For each case, a test the algorithm is made on 5 different maps.

In the next experiment, there is a pick of many different types of map with same size (100 points: 10x10). The tested types of map includes: maps with 0% obstacles, maps with 10% obstacles, maps with 20% obstacles, maps with 30% obstacles, maps with 40% obstacles and maps with 50% obstacles. These test weight values are: 1, 2, 3, 5, 10 and 100. On each map the weighted Theta* run with the same weight value from the beginning of the search to the end. The repeat this same test with the same map but with the different weight values is made. Then a collection of the data is made to determine which weight value is the best for this type of map. The test result can be found from these figures. The best weight value in case 1, 2 and 3 is 5. The best one in case 4 is 3. The best one in case 5 is 2 and best one in case 6 is 1. That means if the complexity of the abstract map is less than 30%, we use the weight value equals to 5. If the complexity of the abstract map is more than or equals to 30% but less than 40%, we use the weight value equals to 3. If the complexity of the abstract map is more than or equal to 40% but less than 50%, we use the weight value equals to 2. And if the complexity is greater than or equals to 50% the weight value is equal 1.

3.2 Dynamic weighted Theta*

In this experiment, square maps with size n x n, which is divided by 10 and has 0 in reminder, are used to perform the dynamic weight theta* and divide the input map into smaller abstract map size (100 points: 10x10). For

example, if users input the map size 20x20(400 points), they have 4 abstract maps size 10x10(100 points each). The weight value of the abstract map is decided based on the complexity of that map. The complexity of the map is calculated by counting the percentage of number of obstacles on the total points of the abstract map. The two bellowing algorithms are tested with different cases: case 1 - map 20x20 with 30% obstacles, case 2 - map 30x30 with 30% obstacles, case 3 - map 40x40 with 30% obstacles, case 4 - map 50x30 with 30% obstacles, case 5 - map 20x20 with 40% obstacles, case 6 - map 30x30 with 40% obstacles, case 7 - map 40x40 with 40% obstacles, case 8 - map 50x30 with 40% obstacles, case 9 - map 20x20 with 50% obstacles, case 10 - map 30x30 with 50% obstacles, case 11 - map 40x40 with 50% obstacles and case 12 - map 50x30 with 50% obstacles.

After determining which value is the best for each type of map's complexity, a test on "DW Theta*" is made once again to compare with the "author Theta*" on different 12 cases as which are listed on the previous paragraph. And from all 12 cases, the positive data is collected to draw some conclusions.

Table 1. Time consuming in the first 6 cases

	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6
Author Theta*	6.66	10.73	5.69	18.18	1.15	2.82
DW Theta*	1.81	2.63	0.82	2.62	0.80	2.71

Table 2. Time consuming in the last 6 cases

	Case 7	Case 8	Case 9	Case 10	Case 11	Case 12
Author Theta*	4.6	8.96	0.35	1.36	1.93	1.97
DW Theta*	3.88	5.36	0.28	0.75	1.32	0.85

Table 3. Path-length in the first 6 cases

	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6
Author Theta*	105.81	332.55	264.96	771.28	121.29	265.93
DW Theta*	183.9	365.84	424.39	1201.98	142.17	318.17

Table 4. Path-length in the last 6 cases

	Case 7	Case 8	Case 9	Case 10	Case 11	Case 12
Author Theta*	595.95	525.61	131.14	373.36	316.94	811.82
DW Theta*	950.54	1640.24	197.16	547.53	405.16	1034.81

4. Experiment Results & Discussion

At these figures 2 to 7, the x-axis represents for the time-consuming, which is calculated in seconds; and the y-axis is used to express the weight values. From these figures 2 to 7, a conclusion is made to determine that the weight value which equals 5 makes the theta* perform faster than other weight value with maps having a percentage of obstacles less than or equal 20%. And for the maps having a percentage of obstacles more than 20% but less than or equal 40%, the best weight value is 2. And finally, for the maps having a percentage of obstacles more than 40%, the best weight value is 1. There is no consideration for these maps with percentage of obstacles which are higher than 60% because these cases are hard to find a solution path. If there is an addition of a weight value which is greater than one into the calculation of F value the runtime will perform faster but the path is not optimal because of the increasing in its path-length. And if there is an addition of weight value which is smaller than one into the calculation of F value the runtime will perform slower but the path is closer-to-optimal because of the increasing in accuracy.

This experiments looks like the part of Nash et. al's research [4 and 14]. But instead of using the stable and unchanged weight value, there is a use of the dynamic weight values which depend on the map's complexity. There is an exchange between the path-length and time cost. In this paper, the dynamic weight is applied on different abstract maps but generally all these weight values are greater than 1. When the applied "DW Theta*" algorithm is compared to the original Theta* - "author Theta*." The "DW Theta*" algorithm performs faster than the "author Theta*" in all 12 cases, as it is on table 1 and table 2. On the other hand, these solutions have their path-length be longer than the "author Theta*" as table 3, Table 4. Nash et.al mentions that Theta* - "author Theta*" performs faster than weighted Theta*. From the experiment, our Theta* - a dynamic weight Theta* performs faster the original Theta*. To explain for this fact, the complexity of A*, which is $O(b^d)$, where b is the average number of successors per state and d is the length of the solution, is analyzed. The number of successors in weighted A* is reduced because when if user applies the weight, they limit many unsuitable cells. This fact cause the b in weight A* is less than the A*. This method might cause a drawback that it has an ability to eliminate the cell or point that should be included on the solution path. Therefore the d value might increase a lot and make the time.

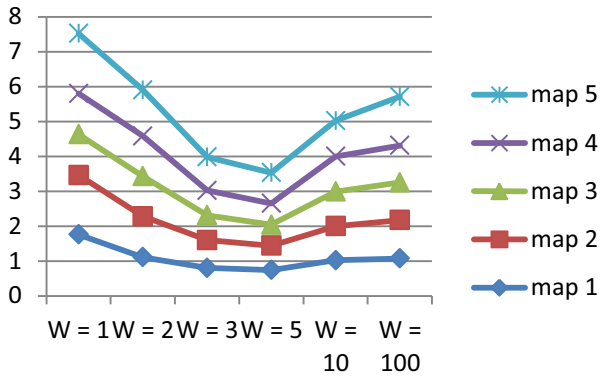

Figure 2. Time costing in case 1 with 5 different maps

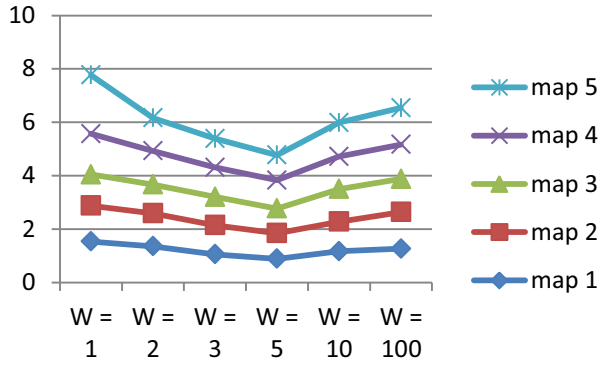

Figure 3. Time costing in case 2 with 5 different maps

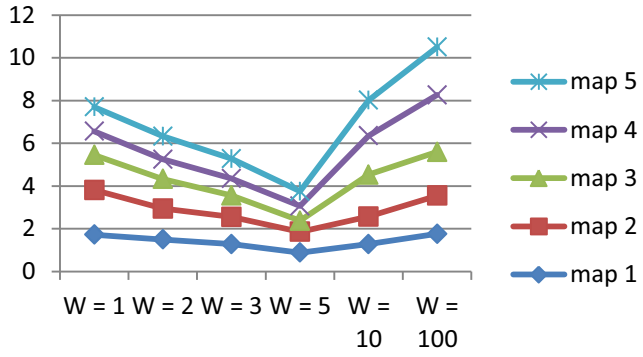

Figure 4. Time costing in case 3 with 5 different maps

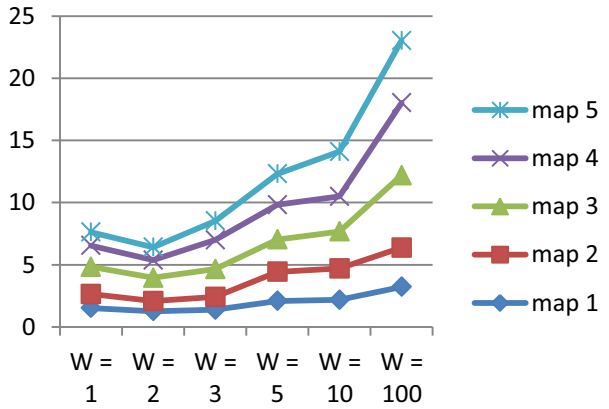

Figure 5. Time costing in case 4 with 5 different maps

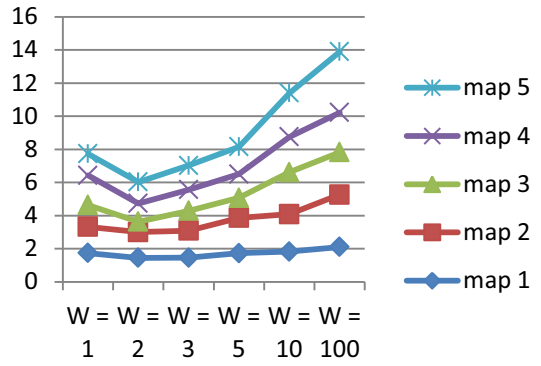

Figure 6. Time co sting in case 5 with 5 different maps

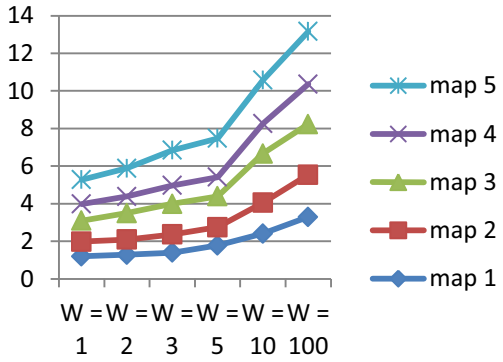

Figure 7. Time costing in case 6 with 5 different maps

5. Conclusion and Future Works

In future work, the application of dynamic weight from 0 to infinite on different map areas based on the percentage of blocks based on the idea of Mr. Wilt in his article [13] is considered. Moreover, the idea of combining dynamic weight theory with the hierarchical method can be applied to these variants of Theta*, such as AP- Theta* or lazy Theta*. As the result of these experiments, weight value makes the Theta* performs well on specific map's complexity is determined. In additional, in this article, an introduction of a way to increase the performance by applying weighted method and hierarchical method from A* to Theta* using the dynamic weight value based on the map complexity is stated. The applied weight values are from 1 to infinite. The new method is tested in many cases to collect data to draw some strong conclusion that the implementation makes the Theta* path-finding process run faster. Applying weight method to heuristic searching algorithms is making a trade-off between path-length and time-consuming. So when dynamic weight method is implemented to Theta*, the same result is showed. The explanation of the trade-off is displayed in the complexity of its – $O(b^d)$, which depends on both the number of expansion neighbor and the length of solution path.

References

1. T.H. Cormen, C.E. Leiserson, and R.L. Rivest, , *Introduction to Algorithm*, McGraw-Hill Companies, Inc., ch.3, (2009)
2. S.E. Putri, and N. Normalina, "Implementation and Analysis of Depth-First Search (DFS) Algorithm for Finding the Longest Path." In International Seminar on Operational Research (InteriOR), (2011)
3. A. Nash, S. Koenig, and A. Felner, "Theta*: Any-Angle Path Planning on Grids," *In Proceeding of the AAAI Conference on Artificial Intelligent*, pp. 1177-1183, (2007)
4. K. Daniel, , A. Nash, S. Koenig and A. Felner, "Theta*: Any-Angle Path Planning on Grids," Journal of Artificial Intelligence Research 39, pp. 533-579, (2010)
5. T. Li, L. Qi, and D. Ruan, "An Efficient Algorithm for the Single-Source Shortest Path Problem in Graph Theory," *Proc. of 3rd International Conference on Intelligent System and Knowledge Engineering*, **volume 1**, pp. 152-157, (2008)
6. C. Xi, F. Qi, and L. Wei, "A New Shortest Path Algorithm based on Heuristic Strategy," Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on, **volume 1**, pp. 2531-2536, (2006)
7. K. Magzhan, and H.M. Jani, "A Review and Evaluation of Shortest Path Algorithms," International journal of scientific & technology research, **volume 2**, issue 6, (2013)
8. J. Freeman, "Parallel Algorithm for Depth-First Search," Technical Reports (CIS), University of Pennsylvania, pp. 428, (1991)
9. X. Cui, and H. Shi, "A*-based Path finding in Modern Computer Games," International Journal of Computer Science and Network Security, **volume 11**, No. 1, pp. 125-130, (2011)
10. A. Botea, and M. Muller, "Near Optimal Hierarchical Path-Finding," Journal of Game Development, **volume 1**, pp. 7-28, (2004).
11. R.M. Jansen, and M. Buro, "HPA* Enhancements," *AAAI Conference on* Artificial Intelligence and Interactive Digital Entertainment, pp. 84-87, (2007)
12. A.E. Hansen, and R. Zhou, "Anytime Heuristic Search," Journal of Artificial Intelligent Research 28, pp. 267-297, (2007)
13. C. Wilt, and W. Ruml, "When does Weighted A* Fail?" Proceeding *of the Symposium on Combinatorial Search (SoCS-12)*, pp. 137-144, (2012)
14. A. Nash, "Any-Angle Path Planning," Doctor of Philosophy Thesis (Computer Science) at the Faculty of the USC Graduate School University of Southern California. , (2012)
15. Phuc, Le T.H., "Applying Dynamic Weight on Theta Star Path-finding Algorithm in 2D Grid Map," *Proceeding of International Conference on Intelligent Computing, Electronics Systems and Information Technology (ICESIT-15),* (2015)