ACSIJ Advances in Computer Science: an International Journal, Vol. 3, Issue 2, No.8 , March 2014
ISSN : 2322-5157
www.ACSIJ.org

ACSIJ
WWW.ACSIJ.ORG

Map Abstraction with Adjustable Time Bounds

Sourodeep Bhattacharjee and Scott D. Goodwin
School of Computer Science, University of Windsor
Windsor, N9B 3P4, Canada
sourodeepbhattacharjee@gmail.com, sgoodwin@uwindsor.ca

Abstract

The paper presented here addresses the problem of path planning in real time strategy games. We have proposed a new algorithm titled Map Abstraction with Adjustable Time Bounds. This algorithm uses an abstract map containing non-uniformly sized triangular sectors; the centroids of the sectors guide the path search in the game map. In a pre-processing step we calculate an upper and lower time limit to plan paths for a given two dimensional grid map that is known beforehand. Depending on the time limits, we vary the size of the sectors to save search time or to improve path quality. We have experimented using maps from commercial games such as Dragon's Age: Origins and Warcraft III. In the worst case MAAT returns paths that are 8% less optimal. MAAT has an expensive pre-processing step which ultimately lowers the overhead CPU time consumed during game play by 1.1 milliseconds.

Keywords: Pathfinding, path-planning, map abstraction, hierarchical

1. Introduction

In this paper we have addressed the problem of path planning in real time strategy game (RTS) maps. In such maps, using a naïve A* search [1] takes more search time than desired [2]. For RTS games, high path planning time becomes a restricting factor when many mobile non-player characters are involved. Numerous improvements to A* search exist which reduce the search time. Hierarchical Pathfinding A* (HPA*) [3] involves placing 1 to 3 levels of hierarchical abstract maps on the game map to find quicker paths by sacrificing path quality. An improved version of HPA* was presented in the paper titled HPA* Enhancements [4]. Advanced sub-goaling algorithms such as LRTA* with sub-goaling [5] improve path quality and lower search time as well.

We have proposed an algorithm – Map Abstraction with Adjustable Time Bounds (MAAT) to determine an upper time limit and lower time limit that should be allocated to path-planning for our planning algorithm based on two dimensional grid maps where the maps are known/explored beforehand. In MAAT a single level abstract map consisting of sectors is placed on top of the game map. These sectors are split or merged to lower search time or increase path quality while attempting to keep the planning time (not including overhead time) within the new time limits. The new algorithm, however, has an expensive offline pre-processing step in which the new time limits are determined along with an initial abstract map of non-uniform sectors for a given map. Once the pre-processing is completed MAAT successfully lowers the overhead involved in modifying the abstract map by 1.1 milliseconds.

Experimental results show that MAAT performs faster than its predecessor Demand Sensitive Map Abstraction [6] and returns paths that are 8% closer to optimal, in the worst case, than HPA* (without path smoothing and refinement). Moreover MAAT stays within the new time bounds in an average of 84% path planning sessions.

2. Background

The current game industry endorsed time bounds for path planning is 1 millisecond to 3 milliseconds [7]. Many techniques are employed to address the high search time of A* search. One of the techniques suggests using abstract maps built from real game world maps and finding an abstract path from the abstract map. This abstract path is then refined into real path. It saves search time to use abstract maps, at the cost of lower path quality (longer paths) (Botea et al., 2004).

In a paper titled Near Optimal Hierarchical Pathfinding, the authors overlay an abstract map comprised of uniform sectors with entrances from one sector to another and within sectors, over the game map. Any path search request is first computed on the abstract map to identify the sectors that should contain the optimal path. The links of these sectors (given by intra and inter edges) give the abstract path. Then the abstract path is refined on the game map using A* search. The authors state that in the worst case the path received is 10% less optimal compared to the path returned by using a naïve A* search, without using path smoothing and refinement.

Demand Sensitive Map Abstraction (DSMA) [6] is another such technique where central points of abstract regions provide sub-goals for guiding the A* search in the real game map. Each sector in the abstract map in DSMA is associated with a demand. Demand of a sector is defined as the number of times the sector contained start and goal points plus the number of times it was expanded by the A* search on the abstract map. Conversely, demand of sectors decrease if they are not expanded in subsequent path planning.

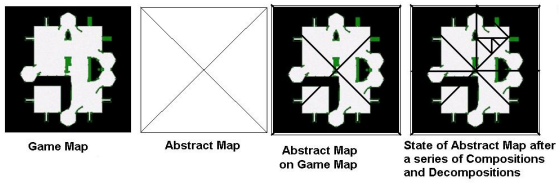


Fig 1. Illustration of non-uniform abstract map

Depending on the time taken to find a real path the sectors are either decomposed or composed non-uniformly across the map, shown in figure 1. If the search time of the previous path went above 3 milliseconds, the highest demand sector was decomposed and if the search time went below 1 millisecond, two neighbor sectors with the lowest collective demand were composed or merged.
This paper suggests a new algorithm – Map Abstraction with Adjustable Time Bounds (MAAT), inspired from the key ideas behind DSMA and addresses the need of transparent policies used in DSMA.

3.Proposed Methodology

3.1 Overview of MAAT

In this section we will discuss the high level A* search on the abstract map and the low level A* search on the game map and the concepts of decomposition and composition. Consider the abstract map in figure 2 where the triangles represent sectors on the map. Sub-goals for the low level A* search are provided by the grids below that coincide with the centroid of the sectors, shown in black squares within the sectors. A group of these central grids is returned by the A* search on the abstract map. The low level A* search connects these grids depending on the map data.
If during the low level search i.e. the A* search on the game map, a centroid-coinciding grid (say g2) is not reachable from one grid (g1) due to obstacles on that grid, then the A* search attempts to connect g1 to the next grid - g3 on the abstract path, where the abstract path is given by the grids g1-> g2 -> g3.

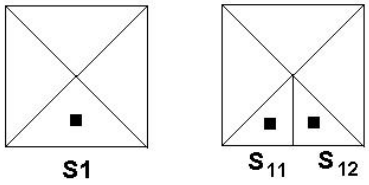


Fig 2. Decomposition and composition

3.2 Pre-processing

This step is employed to find attainable time bounds for the time taken by A* search to find a path (Abstract path + Complete path). This step also returns an initial customized abstract map with non-uniform sectors. This customized abstract map ensures fewer decompositions and compositions after pre-processing.

The pre-processing begins with a game map and an abstract map overlaid on it. N-number of start-goal positions are generated for the map and 1ms -3 ms is taken as a standard time bound from which an attainable time bound is to be extrapolated. If the search time for a certain start goal pair went above 3 milliseconds the *most time consuming* sector in the path is decomposed and the path is recalculated to check the validity of the decomposition. That is if the time after decomposition is not lower than previous search time, the decomposition is reverted. Otherwise, we record the new time along with the previous time and the action taken (composition / decomposition). We also associate a quality with the new search time. This quality is marked as “good” if the new search time is closer to 3 ms and “bad” if it is closer to the previous search time.

Similarly, if the search time goes below 1 millisecond, we compose two least time consuming neighbor sectors and check for the validity and quality of the action. An illustration of relative quality is shown in figure 3, where T1 is the previous time taken and T2 is the new search time.

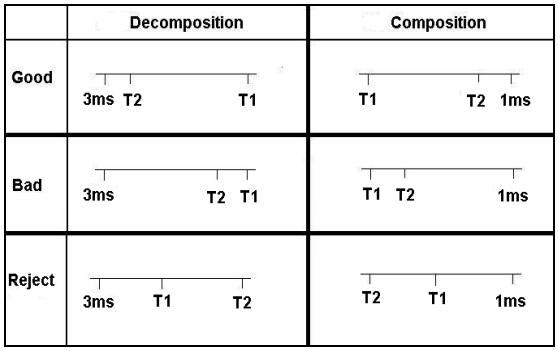


Fig 3. Quality of new search time

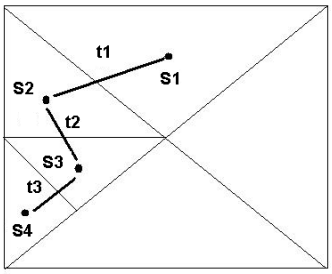


Fig 4. Most time consuming sector

We define most time consuming sector as the common sector in the set of abstract sectors for which the A* search on the real map takes maximum time to return an actual path. Consider figure 4, where S1, S2 and S3 are the centers of the triangular regions and t1, t2 and t3 are time taken by A* search to find real path between S1-S2, S2-S3 and S3-S4 respectively. If t1+ t2 is greater than t2 + t3 then the most time consuming sector is the common sector taking the maximum collective time, the sector containing S2 in this case. This sector is decomposed if the search time went above 3 milliseconds. Conversely, S3 and S4 form the least time consuming sectors (composed / merged when needed).

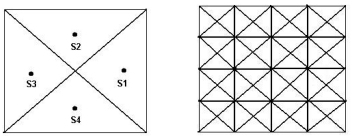
This pre-processing step continues until all the random start-goal positions are exhausted. The new upper time limit is calculated by taking the average of search times with "good" quality for all decompositions. While the new lower time limit is calculated by taking the average of all search times with "good" quality for all compositions. This time range can be considered as the attainable best in the given set of start-goal points (without any random obstacles in the map) as they are average of all search times that were closer to the desirable range. Taking more start-goal points (increasing the value on N) will yield statistically reliable results. These new time limits can be associated with the map and machine they were run on and can be adapted for different systems and maps.

3.3 Online Search

The online search is a simple operation where most time consuming sectors are decomposed if the search time when above the new upper time limit. On the other hand, if the search time went below the new lower time limit, two of the lowest time consuming neighbor sectors are composed or merged.

The decomposition will ensure lower search time as new sub goals (centers of sectors) are generated that are closer to the current position, thereby limiting A* search time on the real map. While the composition action will ensure that the A* search finds better paths, at the cost of more search time by making the sub-goals distant. To sum up we are claiming that if no sub-goals are used, the path returned by A* search will be optimal and as we keep adding more sub-goals, the path loses optimality but the search time is reduced.

During both the pre-processing and online search, if the start and goal lie in the same sector and the search time falls below 1 millisecond, no composition action will take place as we are receiving the optimal path returned by the A* search on the game map. Moreover if the abstract map has only the basic configuration consisting of four sectors, all at level 0 then we pass the request to the real map A* search, as not much time will saved in this trivial case, as shown in figure 5 (a). The maximum decomposition we allow is shown figure 5 (b) where all sectors are at level 4 (i.e. derived by decomposing the base sector 4 times). The maximum level of decomposition can be altered in future.



(a) (b)

Fig 5. Basic configuration and maximum decomposition

4. Experiments

All the maps used for experiments were provided by Nathan Sturtevant [8]. The experiments are also designed according to the specifications mentioned in (Sturtevant, 2012). We have used grid based maps from games like Dragon's Age origins (10 maps, size ranging from 256 x 260 grids to 492 x 512 grids), Warcraft III (10 maps all of

size 512 x 512 grids) and artificially created maps with random obstacles. The map data was formatted into a format in which all traversable terrain were uniform, after replacing shallow water, trees and water bodies with normal ground. All maps have octile navigation and crossing a diagonal has weight √2 while cardinal directions have a cost of 1. Moreover, a diagonal movement is allowed only if an alternative cardinal movement exists to ensure that a diagonal movement is not possible across two obstacles touching each other at the corners.

Random maps are created by randomly introducing obstacles into the map starting with 10% up to a maximum of 40% by a step of 5%. Maps containing more than 40% obstacles are left with very little traversable terrain and hence do not provide challenging pathfinding problems. For each map we have generated 1000 random pairs of start and goal positions always ensuring that the points are connected by successfully running an A* search between them.

To measure the performance of MAAT we have compared it to DSMA and our implementation of HPA* (with three levels of abstraction and without path smoothing and refinement). We have measured the average number of nodes expanded, running time and average path length of the algorithms. To measure the solution quality we have used the percent error metric (Botea et al., 2004) given by the Eq (1).

$$Percent\ Error = \frac{Path\ Length - Optimal\ Path\ Length}{Optimal\ Path\ Length} \times 100 \quad (1)$$

5. Results

The graph in figure 6 (a) shows the number of nodes expanded by the HPA*, DSMA and MAAT. This includes the nodes in abstract map. We can see that MAAT expands more nodes than HPA* but less than DSMA. The reason is that HPA* has more nodes in the abstract map which reduces the nodes expanded in the game map. MAAT has more nodes on the abstract map than DSMA as a result of which MAAT expands fewer nodes on the game map.

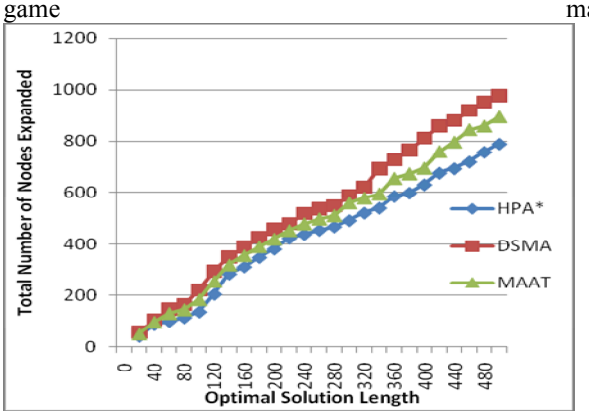


Fig 6 (a).

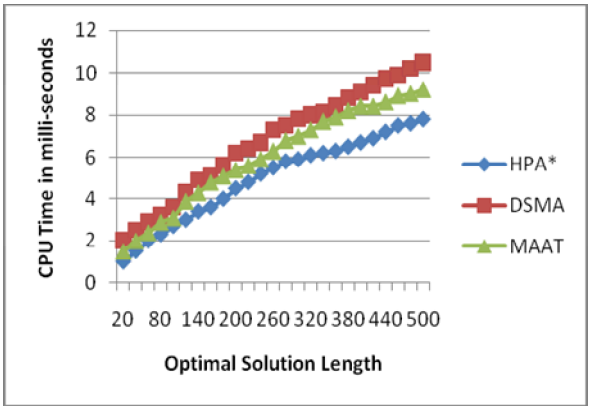


Fig 6 (b). Comparison of nodes expanded and CPU time

The graph in figure 6 (b) indicates the CPU time consumed by the three algorithms to find complete paths. This time does not include the pre-processing step of MAAT, the composition and decomposition queue management of DSMA or the start-goal insertion of HPA*. The results indicate that MAAT takes less time than DSMA but more than HPA* which is consistent with the results on nodes expanded.

We have also found that MAAT takes an average of 0.2 milliseconds in composition or decomposition during online search while DSMA takes an average of 1.3 milliseconds in composition or decompostion operation. The reason is that it stores information regarding sectors in queues and arranges them according to rise and fall in demands associated with them and this operation is time consuming given the current implementation.

Figure 7 shows the graph of error percentage showing that for shorter paths MAAT gives better solution quality. This is due to the fact that MAAT uses naïve A* search if the abstract path goes through basic sectors at level 0. MAAT has consistently better path quality compared to HPA*. This gain in path quality comes at the cost of expensive pre-processing. However, the pre-processed results can be shipped with the game and hence will not affect game play.

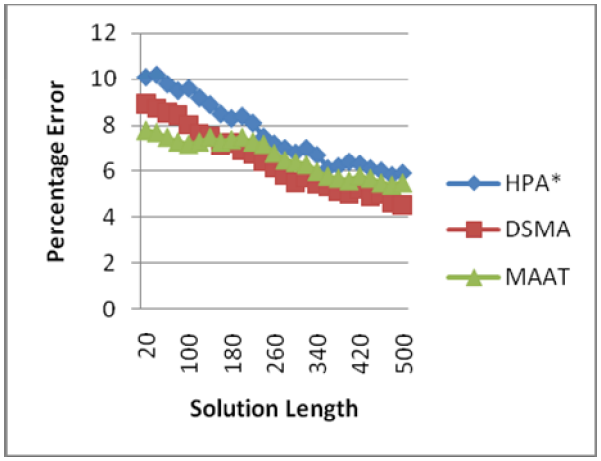


Fig 7. Solution quality

Both DSMA and MAAT aim at trading off path quality to search time and vice versa in real time. However the results indicate that MAAT performance is better as it gives better paths than HPA* by altering the granularity of the abstract map non-uniformly. MAAT also takes less overhead time and less search time compared to DSMA. Moreover, MAAT does not have the high storage requirements of DSMA.

Experiments on the current system indicated that in the pre-processing stage MAAT found the new average lower time limit to be 0.8 ms and the average upper time limit to be 8.4 milliseconds. In an average of 84% cases MAAT was able to stay within these limits, derived from graph in figure 6 (b). On the other hand DSMA was able to stay in the range of 1 ms to 3 ms in an average of 33% cases. This does not indicate that DSMA has poor performance. It show that DSMA struggles with time consuming composition and decomposition operations trying to attain the pre-defined limits. MAAT, on the other hand, saves the overhead time by 1.1 milliseconds, using new time limits.

6. Conclusion

We have proposed a new pathfinding algorithm titled Map Abstraction with Adjustable Time Bounds (MAAT). MAAT assigns a lower and upper time limit for a given map and system using the industry endorsed 1ms – 3ms range as a seed range. The new time limits can be attained by MAAT in 84% cases by altering the granularity of the abstract map non-uniformly and in real time. Experimental results show that MAAT returns better paths than HPA* and takes less time compared to its predecessor - DSMA. It involves using a pre-processing step which is calculated offline. This step is expensive but it will not affect game play. MAAT takes 1.1 milliseconds less overhead time compared to DSMA during online search. Moreover MAAT has a transparent policy which is easy to implement and can be modified in future.

Future work can involve running MAAT for more iterations in the pre-processing step and on different systems so find an average system-independent time range. Another direction of effort can be put in modifying MAAT and the idea of non-uniform abstract maps to three dimensional game worlds comprised of multiple levels.

References

[1] Hart, P.E. et al, 1968. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2), pp.100--107.

[2] Sturtevant, N. & Buro, M., 2005. Partial pathfinding using map abstraction and refinement. In *Proceedings of the National Conference on Artificial Intelligence*. Ch. 3. p.1392.

[3] Botea, A. et al, 2004. Near optimal hierarchical path-finding. *Journal of game development*, 1(1), pp.7-28.

[4] Jansen, R.M. & Michael, B., 2007. HPA* enhancements. In *Proceedings of the Third Artificial Intelligence and Interactive Digital Entertainment Conference, Stanford, California, USA.*, 2007.

[5] Hernandez, C. & Baier, J.A., 2011. Fast subgoaling for pathfinding via real-time search. In *Proceedings of the 21th International Conference on Automated Planning and Scheduling (ICAPS)*., 2011.

[6] Bhattacharjee, S. & Goodwin, S.D., 2013. Pathfinding by Demand Sensitive Map Abstraction. In *Proceedings of 26th Canadian Conference on Artificial Intelligence*.

[7] Bulitko, V. et al, 2007. Graph abstraction in real-time heuristic search. *JAIR*, 30, pp.51-100.

[8] Sturtevant, N.R., 2012. Benchmarks for grid-based pathfinding. *Computational Intelligence and AI in Games, IEEE Transactions on*, 4(2), pp.144-48.

First Author Sourodeep Bhattacharjee has completed Bachelor of Technology in Computer Science (2010) from West Bengal University of Technology and Master of Science in Computer Science

(2012) from University of Windsor, Windsor, Ontario. He is currently employed as Senior Research Fellow with CSIR-Cetral Mechanical Engineering Research Institute – Embedded Systems Laboratory . His research interests revolve around machine learning, artificial intelligence, and energy informatics.

Second Author Dr. Scott Goodwin is professor with School of Computer Science, University of Windsor, Ontario, Canada. His current research interests is Artificial Intelligence in Games.