Fast Grid-Based Path Finding for Video Games

William Lee and Ramon Lawrence

University of British Columbia

Abstract. Grid-based path finding is required in many video games and virtual worlds to move agents. With both map sizes and the number of agents increasing, it is important to develop path finding algorithms that are efficient in memory and time. In this work, we present an algorithm called DBA* that uses a database of pre-computed paths to reduce the time to solve search problems. When evaluated using benchmark maps from Dragon Age™, DBA* requires less memory and time for search, and performs less pre-computation than comparable real-time search algorithms. Further, its suboptimality is less than 3%, which is better than the PRA* implementation used in Dragon Age™.

1 Introduction

As games have evolved, their size and complexity has increased. The virtual worlds and maps have become larger as have the number of agents interacting. It is not uncommon for hundreds of agents to be path finding simultaneously, yet the processing time dedicated to path finding has not substantially increased. Consequently, game developers are often forced to make compromises on path finding algorithms and spend considerable time tuning and validating algorithm implementations.

Variations of A* and PRA* are commonly used in video games [9]. The limitation of these algorithms is that they must plan a complete (but possibly abstract) path before the agent can move. Real-time algorithms such as kNN LRTA* [3] and HCDPS [8] guarantee a constant bound on planning time, but these algorithms often require a considerable amount of pre-computation time and space.

In this work, we propose a grid-based path finding algorithm called DBA* that combines the real-time constant bound on planning time enabled by using a precomputed database as in HCDPS [8] with abstraction using sectors as used in PRA* [9]. The result is a path finding algorithm that uses less space than previous real-time algorithms while providing better paths than PRA* [9]. DBA* was evaluated on Dragon Age™maps from [10] with average suboptimality less than 3% and requiring on average less than 200 KB of memory and between 1 and 10 seconds for pre-computation.

2 Background

Grid-based path finding is an instance of a heuristic search problem. The algorithms studied in this paper, although potentially adaptable to general heuristic search, are specifically designed and optimized for 2D grid-based path finding. States are vacant square grid cells. Each cell is connected to four cardinally (i.e., N, E, W, S) and four

O. Zaïane and S. Zilles (Eds.): Canadian AI 2013, LNAI 7884, pp. 100–111, 2013.
© Springer-Verlag Berlin Heidelberg 2013

diagonally neighboring cells. Out-edges of a vertex are moves available in the cell. The edge costs are 1 for cardinal moves and 1.4 for diagonal moves. Standard octile distance is used for the heuristic.

Algorithms are evaluated based on the quality of the path produced and the amount of time and memory resources consumed. *Suboptimality* is defined as the ratio of the path cost found by the agent to the optimal solution cost minus one and times 100%. Suboptimality of 0% indicates an optimal path and suboptimality of 50% indicates a path 1.5 times as costly as the optimal path.

An algorithm is also characterized by its *response time*, which is the maximum planning time per move. The *overall time* is the total amount of time to construct the full solution, and the *average move time* is the overall time divided by the number of moves made. Memory is consumed for node expansions (e.g. open and closed lists in A*), for storing abstraction information (e.g. regions), and for storing computed paths. *Per agent memory* is memory consumed for each search agent. *Fixed memory* is memory consumed that is shared across multiple, concurrent path finding agents.

2.1 A*

A* [5] is a common algorithm used for video game path finding as it has good performance characteristics and is straightforward to implement. A* paths are always optimal as long as the heuristic function is admissible. However its overall time depends on the problem size and complexity, resulting in highly variable times. The biggest drawback is that its response time is the same as its overall time as a complete solution is required before the agent moves. A*'s memory use is variable and may be high depending on the size of its open and closed lists and the heuristic function used.

2.2 PRA*

The PRA* variant implemented in Dragon Age [9] was designed to improve on A* performance for path finding. Response time and per agent memory are reduced by abstracting the search space into sectors and first computing a complete solution in the abstract space. The abstraction reduces the size of the problem to be solved by A*. The abstraction requires a small amount of fixed memory. It also results in solutions that are suboptimal (between 5 to 15%). The small trade-off in space and suboptimality is beneficial for faster response time.

2.3 Real-Time Algorithms

There have been several real-time algorithms proposed that guarantee a constant bound on planning time per action (response time) including TBA* [1], D LRTA* [4], kNN LRTA* [3], and HCDPS [8]. TBA* is a time-sliced version of A* that exhibits the same properties as A* with the added ability to control response time and per move planning time. All of the other algorithms rely on some form of pre-computation to speed up online search.