Proceedings of the Seventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment

Ultra-Fast Optimal Pathfinding without Runtime Search

Adi Botea
NICTA* and the Australian National University

Abstract

Pathfinding is important in many applications, including games, robotics and GPS itinerary planning. In games, most pathfinding methods rely on runtime search. Despite numerous enhancements introduced in recent years, runtime search has the disadvantage that, in bad cases, most parts of a map need to be explored, causing a time performance degradation. In this work we explore a significantly different approach to pathfinding, eliminating the need for runtime search. Optimal paths between all pairs of locations are pre-computed. Since straightforward ways to store pre-computed paths are prohibitively expensive even for maps of moderate size, pre-computed data are compressed, reducing the memory requirements dramatically. At runtime, pathfinding is very fast, as it requires visiting only the locations on an optimal path. In each location, a quick computation provides the next move along the optimal path. We demonstrate the effectiveness of this approach on Baldur's Gate game maps. The compression factor reaches two orders of magnitude, bringing the memory requirements down to reasonable values. Compared to A* search, the runtime speedup reaches and even exceeds two orders of magnitude. When averaged over paths of similar cost, the speedup reaches a value of 700 in our experiments.

Introduction

Pathfinding has many applications, including games, robotics and GPS itinerary planning. Runtime search is used in most pathfinding techniques. One popular method to obtain a search space is to discretize an initial map into a grid map. Then, a search algorithm, such as A*, enhanced with an admissible heuristic function, such as the Manhattan or the Octile heuristic, can be used for pathfinding.

Numerous enhancements to pathfinding (e.g., on grid maps) have been introduced in recent years. These include reducing the size of the search space with hierarchical abstraction (Botea, Müller, and Schaeffer 2004; Sturtevant and Buro 2005), exploiting symmetry (Harabor and Botea 2010; Harabor and Grastien 2011), and building accurate heuristics at the cost of using extra-memory (Björnsson and Halldórsson 2006; Cazenave 2006; Sturtevant et al. 2009). Despite the success of such enhancements, it is often the case that solving a pathfinding instance requires visiting many locations on a map, resulting in a speed degradation.

We introduce *compressed path databases* (CPDs), a pathfinding approach significantly different from traditional approaches based on runtime search. Our goal is to improve the speed dramatically using pre-processing and additional memory to efficiently store the pre-processing results. While many games feature dynamic environments, and mobile units with different sizes and terrain traversal capabilities, we focus on static environments and single-agent pathfinding. We discuss briefly the case of dynamic obstacles (other mobile units or changes in environment) as well.

During pre-processing, optimal paths between any two locations on a map are computed. If pre-computed data are stored in a naive way, the memory requirements are quadratic in the size of the original map, being prohibitively large even for maps of moderate size. To address this, we introduce a technique for compressing path information, reducing memory requirements dramatically.

With CPDs in use, runtime pathfinding is very fast. It only requires visiting each location along an optimal path from the start to the target location. In each location, a quick computation provides the next location where to move along an optimal path. The first-move lag, which measures the time needed before making the first move on a path, is extremely small, since deciding the next move is independent of deciding the rest of the moves. Computing the next move is faster even than in real-time search, as the latter relies on a small local search to compute a move.

Similarly to Sankaranarayanan, Alborzi, and Samet (2005), our work exploits the idea that, often, the shortest paths from a current node to any target in a remote area share a common first move. These authors call that property *path coherence*. There are significant differences between our work and the approach taken by Sankaranarayanan, Alborzi, and Samet (2005). First off, Sankaranarayanan, Alborzi, and Samet compress a map using a quad-tree decomposition. Our method decomposes a map into a list of rectangles that can have arbitrary sizes and placements, which can potentially require significantly fewer rectangular blocks to cover the map. We will show that, in our experiments, CPDs are significantly more

*NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.