

Bridging the Gap Between Centralised and Decentralised Multi-Agent Pathfinding

Ko-Hsin Cindy Wang

The Australian National University & NICTA

1 Introduction

Multi-agent path planning involves navigating mobile units to their respective targets, while avoiding collisions with obstacles and other moving units in the shared environment. Since movement is fundamental to many task-performing intelligent agents, path planning has important real-life applications, including robotics, air traffic control, military operations, logistics, disaster rescue, and commercial games.

Even inside a two-dimensional obstacle space that has been discretized into a grid map, path planning for multiple circular units is much more challenging than the *single agent* case, where the size of the state space is bounded by the map size. Single-agent path planning in this setting can be tackled with a search algorithm such as A* [Hart *et al.*, 1968]. In multi-agent problems, however, both the number of states and the branching factor grow exponentially in the number of agents, as agents move simultaneously. Considering a state as the configuration of agents on the map, the combined state space of n agents on a map with m tiles has up to $\frac{m!}{(m-n)!}$ states, or $O(m^n)$, and a maximum branching factor of 5^n when only straight moves in the four cardinal directions, plus waiting, are allowed. Hence, despite its completeness and solution optimality guarantees, a global search such as centralized A* has little practical value in a multi-agent path planning problem, being intractable even for small collections of agents on relatively small maps. In robotics, a related problem on continuous motion planning for rectangular agents of variable size was shown to be PSPACE-hard by Hopcroft *et al.* (1984).

In addition to the inherent difficulty of the problem, in many real-life applications, such as computer games, solutions have to be computed in real time, using limited CPU and memory resources [Bulitko *et al.*, 2007a]. In the example of games, a search algorithm gets limited access to the CPU and memory resources, which are allocated with higher priority to other game modules such as the graphics engine. Yap [2002] recounts some of the problems with pathfinding in games that are frustrating to the player. A path planning algorithm able to scale up to many agents while using reasonably small resources can make a significant contribution to the overall quality of a game.

Scalability to larger problems has been achieved with decentralized approaches, which decompose an initial prob-

lem into a series of searches [Silver, 2005; Wang and Botea, 2008]. Even though their CPU and memory requirements are significantly lower, existing decentralized methods, including Silver's and Wang and Botea's, are incomplete and provide no criteria to distinguish between problems that can successfully be solved and problems where such algorithms fail. Further, no guarantees are given with respect to the running time, the memory requirements, and the quality of the computed solutions. Addressing such limitations is the central motivation for our recent and current work on identifying a class of problems and developing an algorithm that is complete on this class of problems, with guarantees of low-polynomial running time, memory requirements and solution length [Wang and Botea, 2009]. This is the starting point of a series of studies that we have planned on investigating the theoretical properties of multi-agent path planning problems. We also address the practical issues by developing algorithms that work well in practice and that are complete for well specified classes of problems.

2 Related Work

Often, single-agent path planning involves moving a single agent among a set of fixed obstacles whose locations are accurately known, from the start to the goal location. After reducing the continuous domain to a discrete search graph such as a grid map [Yap, 2002], a navigation mesh (e.g., [Tozour, 2002]), a graph with visibility points (e.g., [Rabin, 2000]), or a quadtree (e.g., [Yahja *et al.*, 1998]), the problem can be solved by performing an offline graph search such as A* [Hart *et al.*, 1968]. A* has been shown to be a very efficient algorithm for finding an optimal solution (Pearl 1984) to single-agent pathfinding.

Traditional multi-agent path planning takes two main approaches [Latombe, 1991]. A *centralised* method incorporates a global decision maker to plan paths for all units simultaneously, which is theoretically optimal but in practice has prohibitive complexity and cannot scale up to many units [Barraquand and Latombe, 1991]. On the other hand, a *decentralised* method can significantly reduce computations by decomposing the problem into several subproblems. A typical approach first computes paths individually, ignoring all other units, then handles the interactions along the way to avoid collisions. This is often much faster but yields suboptimal solutions and loses completeness.

Local Repair A* (LRA*) [Stout, 1996] is a method that applies A* directly to handle multiple units independently. With every collision, LRA* replans using expensive A* searches, exhibiting an intensive CPU usage [Bulitko *et al.*, 2007b], and generating cycles [Silver, 2005]. Other decentralized approaches include coordination [Leroy *et al.*, 1999] or using a prioritisation scheme to process each robot in a preassigned order, while treating others as moving obstacles [Erdmann and Lozano-Perez, 1986]. Jansen and Sturtevant (2008) use a direction map structure to share information about units' directions of travel, so later units can follow the earlier ones. The improved coherence leads desirably to reduced collisions. Ryan (2008) introduces a complete method that combines multi-agent path planning with hierarchical planning on search graphs with specific structures such as stacks, halls, cliques and rings. Silver's cooperative pathfinding method (2005) uses a reservation table and performs windowed forward searches on each unit, based on a true distance heuristic obtained from an initial backward A* search from each target.

3 Outline of Progress and Future Plan

From my BSc Honours year through the first year of my PhD, the initial focus of our research was on optimizing computation speed and memory usage on large-scale multi-agent path planning problems (large in both the number of units and the map size). Our FAR algorithm [Wang and Botea, 2008] introduced an efficient search graph structure that implements a flow annotation idea inspired by real-life road networks. The movement along a given row (or column) is restricted to only one direction, avoiding head-to-head collisions. The movement direction alternates from one row (or column) to the next. After building the search graph, an A* search is independently run for each mobile unit. During plan execution, a heuristic procedure breaks deadlocks online, and repairs plans locally. We subsequently addressed the importance of guaranteeing completeness by identifying a tractable subclass of problems on grid maps. We also provided an algorithm for solving it, called MAPP (multi-agent path planning), that brings a blank to the front of a blocked unit by sliding other units along an alternate path. Formal proofs of MAPP's low polynomial running time and solution length in the worst case are given in [Wang and Botea, 2009].

Currently, we are pursuing this tractability study by evaluating the properties of our identified class and the performance of our algorithm. Research projects stemming from this study include finding ways of obtaining different tractable classes systematically, and using tractable classes to solve multi-agent path planning problems efficiently.

In future work, we plan to look into the complexity of the grid map problems that we have been working on. We will further study the trade-off between completeness and efficiency, and balance the right amount of computation for best performance. Unlike the single-agent problem, the quality of solutions with multiple agents are not solely based on shortest path lengths. Hence we also plan to investigate makespan-optimal cost functions too. Lots of open problems remain: we could extend our work to allow additional degrees of free-

dom, such as allowing diagonal moves; or handle more general cases, like units of different speed and sizes. Last but not least, I will actively seek applications (e.g. robots) for implementing our algorithms in real life.

References

- [Barraquand and Latombe, 1991] J. Barraquand and J.-C. Latombe. Robot Motion Planning: A Distributed Representation Approach. *IJRR*, pages 628–649, 1991.
- [Bulitko *et al.*, 2007a] V. Bulitko, Y. Bjornsson, M. Luvstrek, J. Schaeffer, and S. Sigmundarson. Dynamic Control in Path-Planning with Real-Time Heuristic Search. In *Proceedings of ICAPS*, pages 49 – 56, 2007.
- [Bulitko *et al.*, 2007b] V. Bulitko, N. Sturtevant, J. Lu, and T. Yau. Graph Abstraction in Real-time Heuristic Search. *JAIR*, 30:51–100, 2007.
- [Erdmann and Lozano-Perez, 1986] M. Erdmann and T. Lozano-Perez. On Multiple Moving Objects. In *ICRA*, pages 1419–1424, 1986.
- [Hart *et al.*, 1968] P. Hart, N. Nilsson, and B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. on Systems Science and Cybernetics*, 4(2):100–107, 1968.
- [Hopcroft *et al.*, 1984] J. E. Hopcroft, J. T. Schwartz, and M. Sharir. On the complexity of motion planning for multiple independent objects: PSPACE-hardness of the “warehouseman’s problem”. *IJRR*, pages 76–88, 1984.
- [Jansen and Sturtevant, 2008] R. Jansen and N. Sturtevant. A New Approach to Cooperative Pathfinding. In *AAMAS*, pages 1401–1404, 2008.
- [Latombe, 1991] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.
- [Leroy *et al.*, 1999] S. Leroy, J.-P. Laumond, and T. Siméon. Multiple Path Coordination for Mobile Robots: A Geometric Algorithm. In *IJCAI*, pages 1118–1123, 1999.
- [Rabin, 2000] S. Rabin. A* Speed Optimizations. In *Game Programming Gems*, pages 272–287. 2000.
- [Ryan, 2008] M. R. K. Ryan. Exploiting Subgraph Structure in Multi-Robot Path Planning. *JAIR*, 31:497–542, 2008.
- [Silver, 2005] D. Silver. Cooperative Pathfinding. In *AIIDE*, pages 117–122, 2005.
- [Stout, 1996] B. Stout. Smart Moves: Intelligent Pathfinding. *Game Developer Magazine*, October/November 1996.
- [Tozour, 2002] P. Tozour. Building a Near-Optimal Navigation Mesh. In *AI Game Programming Wisdom*, pages 171–185. 2002.
- [Wang and Botea, 2008] K.-H. C. Wang and A. Botea. Fast and Memory-Efficient Multi-Agent Pathfinding. In *ICAPS*, pages 380–387, 2008.
- [Wang and Botea, 2009] K.-H. C. Wang and A. Botea. Tractable Multi-Agent Path Planning on Grid Maps. In *IJCAI*, 2009.
- [Yahja *et al.*, 1998] A. Yahja, A. Stentz, S. Singh, and B. Brummit. Framed-Quadtree Path Planning for Mobile Robots Operating in Sparse Environments. In *ICRA*, pages 650–655, 1998.
- [Yap, 2002] P. Yap. Grid-Based Path-Finding. In *Proceedings of the 15th Conference of the Canadian Society for Computational Studies of Intelligence*, pages 44–55, 2002.