Scalable Multi-Agent Pathfinding on Grid Maps with Tractability and Completeness Guarantees

Ko-Hsin Cindy Wang and **Adi Botea**[1]

1 Introduction

Navigating multiple mobile units on grid maps is an NP-complete problem [4, 1] with many real-life applications. Centralized search in the combined state space of all units scales very poorly. Previous approaches that decompose the initial problem into a series of smaller searches, such as FAR [5] and WHCA* [2], can significantly improve scalability and speed. However, such methods are incomplete. They provide no guarantees with respect to the total running time, and are unable to apriori tell whether they would succeed in finding a solution to a given instance.

More recent algorithms, such as MAPP [6] and BIBOX [3], are complete on well-specified subclasses of problems. They also provide low-polynomial upper bounds for the running time, the solution length and the memory requirements. However, their empirical speed and scalability, compared to incomplete methods, has been an open question.

In this paper we take steps towards bridging the gap between the two categories of algorithms, combining strengths specific to each of them. We extended MAPP to improve its completeness range, solution quality, and total runtime, addressing main bottlenecks of the original algorithm. We performed the first empirical analysis of MAPP, showing that the enhanced MAPP has better success ratio and scalability than state-of-the-art incomplete algorithms, and is competitive in running times, with at most 92% longer solutions, while maintaining the theoretical properties of the original MAPP.

2 Original and Extended MAPP

Following previous work [6, 5, 2] we use grid maps with fixed obstacles. A tile of the grid is either traversable and can be accessed by up to one unit at a time, or permanently blocked off to units. In our 4-connect grids, only straight cardinal moves are allowed. We provide an intuitive description of MAPP and our extensions, skipping important details due to lack of space; a full description will be provided in a longer report. For more formal details of the basic MAPP algorithm, we guide the reader to the original publication [6].

For each mobile unit in a problem instance, MAPP attempts to compute a path π from the start to the target satisfying three properties. First, *blank availability* requires that, in the initial state, all units have a blank (i.e., empty location) at the first step of their precomputed π paths, ensuring there is room for the first moves. Second, every three consecutive locations along a path π have an alternate path, called Ω, connecting the two end locations without passing through the middle location. This property is called *alternate connectivity*, allowing a blocked unit to try to slide the blocking unit(s) along Ω, to bring a blank to its front again (similarly to how the blank travels in sliding tile puzzles). Lastly, *target isolation* requires that a target belongs to no other unit's π or Ω paths; so that once a unit gets on its target, it is out of the way, reducing the problem by one.

If all units are SLIDEABLE (satisfying the three stated conditions), the instance belongs to the SLIDEABLE class, which MAPP is guaranteed to solve. Otherwise, our implementation of MAPP computes a partial solution for solving the SLIDEABLE units, while non-SLIDEABLE units are still kept on the map.

An initial experimental evaluation indicated that the last two conditions are the most restrictive on our test data, limiting MAPP's ability to handle tunnels and to scale up to more units. To extend the completeness of MAPP beyond the SLIDEABLE class, we refined the suggestion in [6] regarding target isolation, and designed a new extension for alternate connectivity. We also improved MAPP's performance in terms of plan length and runtime. Our contributions are:

1. When targets are close together, *target isolation*, forbidding π and Ω paths to pass through targets, can create a "virtual wall", disconnecting the map. Our extension allows a unit to plan a path through other targets, when it can still be guaranteed that all units involved will reach their targets.
2. Single-width tunnels present another bottleneck. When a tunnel bridges two disjoint regions, basic MAPP fails to find a SLIDEABLE path since *alternate connectivity* is unsatisfiable for triples inside the tunnel. We introduce a technique that allows paths to cross tunnels. Using the blank positions ahead, along the remaining pre-computed path, a tunnel-crossing unit pushes its blocking units forward. A buffer area ensures it has enough room to clear each tunnel along its path.
3. In MAPP, to avoid replanning, units pushed off-track by other units undo some of their moves to get back on their π-paths. After observing that the original undo strategy leads to many useless undo moves, we designed a new strategy that reduces the number of moves and maintains the guarantee of converging to a solution.
4. A detailed empirical analysis of MAPP is presented in Section 3.

Each added enhancement maintains MAPP's solid theoretical properties of guaranteeing to solve every unit within its completeness range in low-polynomial time, unlike heuristic approaches which provide no such guarantees.

3 Experimental Results

We compared our enhanced MAPP with basic MAPP and with existing state-of-the-art decoupled, potentially fast but incomplete methods, FAR [5] and WHCA* [2]. We implemented MAPP on top of

[1] NICTA & The Australian National University, email: {cindy.wang, adi.botea}@nicta.com.au; NICTA is funded by the Australian Government's *Backing Australia's Ability* initiative.

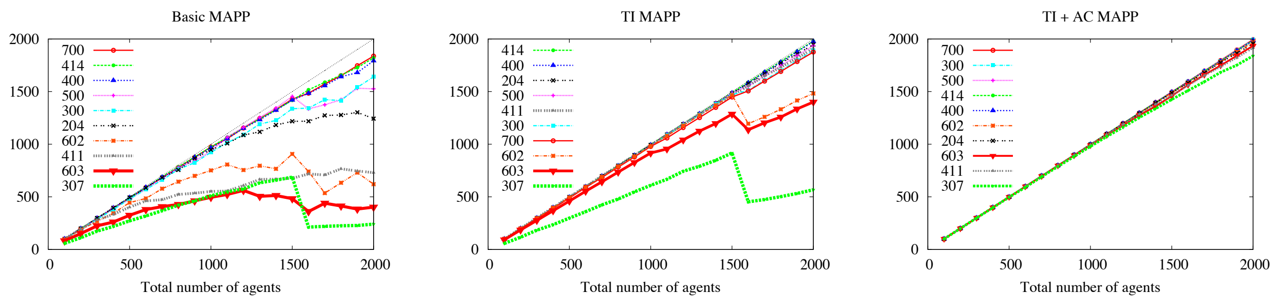

Figure 1. The number of SLIDEABLE units for each instance of each map is plotted on the left (the top straight diagonal line being the total number of units), followed by improvements after each extension. For brevity, only the 3-digit unique ID for each map is listed in the legend.

Hierarchical Open Graph (HOG)[2]. The input data, a set of randomly generated instances, was also used in previously published work [5]. The input maps[3] are 10 of the largest from the game BALDUR'S GATE, 13765 to 51586 traversable tiles in size, each with different configurations of rooms, corridors, and narrow tunnels. We test each map with 100 to 2000 mobile units. All experiments were run on a 2.8 GHz Intel Core 2 Duo iMac with 2GB RAM.

We use the success ratio, i.e. the percentage of solved units, to evaluate the usefulness of MAPP's (partial) completeness in practice. The success ratio is a direct measure of MAPP's completeness range because MAPP solves a unit iff it can a priori guarantee to solve it. We compared 3 versions of MAPP: *Basic* MAPP; *TI* MAPP with the target isolation extension; and *TI+AC* MAPP with both *TI* and alternate connectivity extensions. Figure 1 shows the improved success ratio from often below 50% to at least 92%, and most of the times close to 100%. The success ratios of FAR and WHCA* suffer greatly on the harder instances, solving as low as only around 12.5% of units.

The third enhancement in Section 2 eliminates many unnecessary undo moves, reducing MAPP's travelled distance by half or even more. Evaluating the solution quality of enhanced MAPP, FAR and WHCA* on the subset of instances that FAR and WHCA* can complete, MAPP's distance can be at best 20% shorter than WHCA* without diagonal moves, and 24% worse on average. Compared to WHCA* with diagonals, MAPP's total distance is from comparable to at most 92% longer; on average 53% worse. Comparing with FAR's travel distance, MAPP has between 20–76% longer distance; on average 47% worse. A typical hard case, where MAPP's total distance increases at a faster rate than the rest, is a direct result of an increasingly larger number of undo moves. A closer inspection reveals that even with enhancement (3), MAPP still makes many unnecessary moves. Each useless undo counts double in the final solution length, as it has to be matched by a new forward move. Improving the solution length further is a promising direction for future work.

MAPP's total running time grows at a smaller rate than WHCA* (Figure 2). In most cases, MAPP's curve is below WHCA* with diagonal moves, and often below WHCA* no diagonals. With the overhead of computing the Ω paths, MAPP is slower than FAR. The Ω computation takes up the majority of MAPP's search, and stays constant to an increasing number of agents on each map. This strongly supports that MAPP's total runtime can be significantly reduced by taking the Ω computations offline into a map pre-processing step, re-using Ω paths between instances on the same map.

[2] `http://webdocs.cs.ualberta.ca/~nathanst/hog.html`
[3] `http://users.cecs.anu.edu.au/~cwang/`

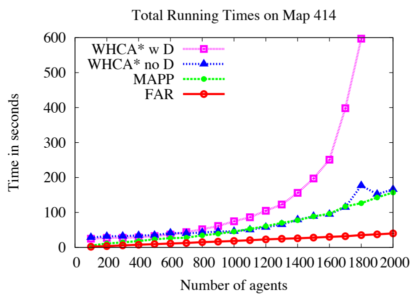

Figure 2. An average case of total running times.

4 Conclusion

In this work, we bridged a missing link between having formal completeness guarantees with scalable, efficient performance for multi-agent pathfinding in practice. We extended MAPP, a recent tractable algorithm that is complete on a subclass of problems, called SLIDEABLE, significantly improving its completeness range to 92–99.7% even in challenging scenarios with 2000 mobile units. Addressing MAPP's key bottleneck in distance and running time, our new undo strategy can reduce the travel distance by 50% and even more. We presented the first experimental evaluation of MAPP with these enhancements, showing its empirical performance has better success ratio and scalability than state-of-the-art decentralised algorithms, which are fast but incomplete. MAPP is also competitive in running time, with at most 92% longer solutions as reported in Section 3, while offering completeness guarantees.

REFERENCES

[1] D. Ratner and M. Warmuth, 'Finding a shortest solution for the $N \times N$ extension of the 15-puzzle is intractable', in *AAAI*, pp. 168–172, (1986).

[2] D. Silver, 'Cooperative pathfinding', *AI Programming Wisdom*, **3**, 99–111, (2006).

[3] P. Surynek, 'An Application of Pebble Motion on Graphs to Abstract Multi-robot Path Planning', in *ICTAI*, pp. 151–158, (2009).

[4] Pavel Surynek, 'An Optimization Variant of Multi-Robot Path Planning is Intractable', in *AAAI*, (2010).

[5] K.-H. C. Wang and A. Botea, 'Fast and Memory-Efficient Multi-Agent Pathfinding', in *ICAPS*, pp. 380–387, (2008).

[6] K.-H. C. Wang and A. Botea, 'Tractable Multi-Agent Path Planning on Grid Maps', in *IJCAI*, (2009).