Thesis Summary: Optimal Multi-Agent Pathfinding Algorithms

Guni Sharon

Abstract

The *multi-agent path finding* (MAPF) problem is a generalization of the single-agent path finding problem for $k > 1$ agents. It consists of a graph and a number of agents. For each agent, a unique start state and a unique goal state are given, and the task is to find paths for all agents from their start states to their goal states, under the constraint that agents cannot collide during their movements. In many cases there is an additional goal of minimizing a cumulative cost function such as the sum of the time steps required for every agent to reach its goal. The goal of my research is providing new methods to solve MAPF optimally and provide theoretical understandings that will help choose the best solver given a problem instance.

Introduction and Research Question

In the *multi-agent path finding* (MAPF) problem, we are given a graph, $G(V, E)$, and a set of $k > 1$ agents labeled $a_1 \ldots a_k$. Each agent a_i has a start position $s_i \in V$ and goal position $g_i \in V$. At each time step an agent can either *move* to a neighboring location or can *wait* in its current location. The task is to return the least-cost set of actions for all agents that will move each of the agents to its goal without *conflicting* with other agents (i.e., without being in the same location at the same time or crossing the same edge simultaneously in opposite directions). In the online variant of this problem agents may appear at any time step and disappear once they reach their goal. MAPF has practical applications in video games, traffic control (Silver 2005; Dresner and Stone 2008), robotics (Bennewitz *et al.* 2002) and aviation (Pallottino *et al.* 2007). Up until now I presented two algorithms for solving MAPF optimally: the Increasing Cost Tree Search (Sharon *et al.* 2013b) (ICTS) and the Conflict Based Search (Sharon *et al.* 2012a) (CBS).

ICTS is based on the understanding that a *complete solution* for the entire problem is built from *individual paths* (one for each agent). ICTS divides the MAPF problem into two levels. ***High level:*** At its *high level*, ICTS searches the *increasing cost tree* (ICT). Every node in the ICT consists of a k-ary vector $[C_1, \ldots C_k]$ which represents *all* possible solutions in which the individual path cost of agent a_i is exactly C_i. The root of the ICT is $[opt_1, ..., opt_k]$, where opt_i is the optimal individual path cost for agent a_i, i.e. the shortest path length from s_i to g_i while ignoring other agents. A child in the ICT is generated by increasing the cost limit for one of the agents by one (or some unit cost). An ICT node $[C_1, .., C_k]$ is a *goal* if there is a complete non-conflicting solution such that the cost of the individual path for a_i is exactly C_i.

Low level: The low level acts as a goal test for the high level. For each ICT node $[C_1, .., C_k]$ visited by the high level, the *low level* is invoked. The task of the low level is to find a non-conflicting complete solution such that the cost of the individual path of agent a_i is exactly C_i. For each agent a_i, ICTS stores *all* single-agent paths of cost C_i in a special compact data-structure called a *multi-value decision diagram* (MDD) (Srinivasan *et al.* 1990). The low level searches the cross product of the MDDs in order to find a set of k non-conflicting paths for the different agents. If such a non-conflicting set of paths exists, the low level returns *true* and the search halts. Otherwise, *false* is returned and the high level continues to the next high-level node (of a different cost combination).

1. **A set of constraints** ($N.constraints$). Each constraint prohibits a given agent from being at a given coordinate (location, time step). The root of the CT contains an empty set of constraints. The child of a node in the CT inherits the constraints of the parent and adds one new constraint for one agent.
2. **A solution** ($N.solution$). A set of k paths, one path for each agent. The path for agent a_i must be consistent with the constraints of a_i. Such paths are found by the low-level search algorithm.
3. **The total cost** ($N.cost$). The cost of the current solution (summation over all the single-agent path costs).

Node N in the CT is a goal node when $N.solution$ is valid, i.e., the set of paths for all agents have no conflicts. The high-level phase performs a best-first search on the CT where nodes are ordered by their costs. Given the list of constraints for a node N of the CT, the low-level search is invoked. This search returns one shortest path for each agent, a_i, that is consistent with all the constraints associated with a_i in node N. If the chosen paths for two agents, a_1 and a_2, are found to conflict at coordinate x, two new CT nodes

(N_1, N_2) are generated. At N_1, a_1 is constrained from coordinate x and at N_2, a_2 is constrained from x.

Many open question and research directions still exist regarding ICTS and CBS, namely:

1. Given a problem instance, which algorithm should be used? - A theoretical understandings on how each of the algorithms perform in different settings, followed by an empirical comparison between these algorithms and other optimal MAPF algorithms.
2. How to adapt these algorithms to be suboptimal?
3. How to adapt these algorithms to solve online MAPF?
4. How to generalize these algorithms to other problems?

Timeline

Since 2011 (as a MSc. student) I have been researching the MAPF problem. I intend to continue this line of work and answer the questions presented above.

Progress so Far

During 2011 I have presented the Increasing Cost Tree Search (ICTS) (Sharon *et al.* 2011a). Later that year I presented special pruning techniques that dramatically reduce the size of the search space used by ICTS (Sharon *et al.* 2011b). A journal paper that summarize the research on ICTS was published in 2013 (Sharon *et al.* 2013b).

During 2012 I presented the Conflict Based Search (CBS) (Sharon *et al.* 2012a). CBS has a major drawback when solving MAPF instances that is dense with agents that conflict frequently. In order to mitigate the drawback of CBS I presented, later that year, a variant of CBS named Meta-Agent Conflict Based Search (MA-CBS) (Sharon *et al.* 2012b), for which I won the SOCS best paper award. Instead of planning a path for a single agent at a time, MA-CBS allows merging agents and solving them as a composite agent.

During 2013 I presented a generalization of CBS for solving CSPs (Sharon *et al.* 2013a). The main idea here is to view each variable and a value it may take as an agent and a path it may choose, than solve this problem using CBS. Preliminary results suggest this to be a very promising direction for future research. Besides being applicable to CSPs, CBS could be used to solve many other problems. Characterizing the set of problems that could be solved using CBS is one direction I aim to pursue.

During 2014 I was part of team researching suboptimal variants for the CBS algorithms (Barer *et al.* 2014). The main concept here was to use a greedy search on the CT while exploiting different heuristic evaluations. Using greedy search on the ICT for a suboptimal variant for ICTS is also on my agenda and is left for future work.

Future Research

Currently I am working towards generalizing the CBS algorithm to other problems, namely CSP, SAT and multi agent planning. Hopefully by AAAI-15 I will have preliminary results for these generalizations. During 2015 I intend to perform a comprehensive comparison between the different techniques for solving MAPF that will include new theoretical understandings for characterizing the advantages and drawbacks of each approach. Simultaneously, I intend to implement and experiment with a sub-optimal variant of the ICTS algorithm as well as adapting CBS to the online MAPF variant.

References

Max Barer, Guni Sharon, Roni Stern, and Ariel Felner. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, pages 961–962, 2014.

Maren Bennewitz, Wolfram Burgard, and Sebastian Thrun. Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots. *Robotics and autonomous systems*, 41(2):89–99, 2002.

Kurt M. Dresner and Peter Stone. A multiagent approach to autonomous intersection management. *Journal of Artificial Intelligence Research*, 31:591–656, 2008.

Lucia Pallottino, Vincenzo Giovanni Scordio, Antonio Bicchi, and Emilio Frazzoli. Decentralized cooperative policy for conflict resolution in multivehicle systems. *Robotics*, 23(6):1170–1183, 2007.

Guni Sharon, Roni Stern, Meir Goldenberg, and Ariel Felner. The increasing cost tree search for optimal multi-agent pathfinding. In *IJCAI*, pages 662–667, 2011.

Guni Sharon, Roni Stern, Meir Goldenberg, and Ariel Felner. Pruning techniques for the increasing cost tree search for optimal multi-agent pathfinding. In *Symposium on Combinatorial Search (SOCS)*, 2011.

Guni Sharon, Roni Stern, Ariel Felner, and Nathan R. Sturtevant. Conflict-based search for optimal multi-agent path finding. In *AAAI*, 2012.

Guni Sharon, Roni Stern, Ariel Felner, and Nathan R. Sturtevant. Meta-agent conflict-based search for optimal multi-agent path finding. In *Symposium on Combinatorial Search (SOCS)*, 2012.

Guni Sharon, Ariel Felner, Roni Stern, and Nathan R. Sturtevant. Partial domain search tree for constraint-satisfaction problems. In *Late-Breaking Developments in the Field of Artificial Intelligence, Bellevue, Washington, USA, July 14-18, 2013*, 2013.

Guni Sharon, Roni Stern, Meir Goldenberg, and Ariel Felner. The increasing cost tree search for optimal multi-agent pathfinding. *Artificial Intelligence*, 195:470–495, 2013.

David Silver. Cooperative pathfinding. In *Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 117–122, 2005.

Arvind Srinivasan, Timothy Ham, Sharad Malik, and Robert K. Brayton. Algorithms for discrete function manipulation. In *International Conference on Computer Aided Design (ICCAD)*, pages 92–95, 1990.