Feasibility Study: Moving Non-Homogeneous Teams in Congested Video Game Environments[*]

Hang Ma, Jingxing Yang, Liron Cohen, T. K. Satish Kumar and **Sven Koenig**
Department of Computer Science, University of Southern California
{hangma,jingxiny,lironcoh,skoenig}@usc.edu, tkskwork@gmail.com

Abstract

Multi-agent path finding (MAPF) is a well-studied problem in artificial intelligence, where one needs to find collision-free paths for agents with given start and goal locations. In video games, agents of different types often form teams. In this paper, we demonstrate the usefulness of MAPF algorithms from artificial intelligence for moving such non-homogeneous teams in congested video game environments.

Introduction

Path finding is a component of many video games. For example, agents in turn-based or real-time strategy games need to plan collision-free paths from their current locations to their goal locations, often in dynamic and congested environments. Moving the agents in a team rather than individually makes it easier for players to control hundreds of agents. Furthermore, the agents often have types and thus form a non-homogeneous team. In Dragon Age: Origins, for example, a player moves teams of agents, where a team might consist of mages, warriors, and rogues. Agents of the same type form a group within the team because they can interchange their goal locations. In our example, the mages thus form the first group, the warriors form the second group, and the rogues form the third group. Each goal location reserved for a mage, warrior, or rogue can be occupied only by a mage, warrior, or rogue, respectively, but it does not matter by which one. A similar situation arises in Age of Empires for archers, spearmen, and knights.

We therefore study the problem where a player moves a non-homogeneous team by occasionally specifying goal locations for them, for example, after observing new parts of the environment. The player may specify new goal locations even before all agents have reached the previously specified goal locations. We demonstrate the usefulness of multi-agent path finding (MAPF) algorithms from artificial intelligence for finding collision-free paths for all agents.

[*]Our demonstration paper resulted from the research of undergraduate student Jingxing Yang, primarily guided by graduate students Hang Ma and Liron Cohen. It was supported by NSF under grant numbers 1409987 and 1319966.

Multi-Agent Path Finding

MAPF is NP-hard to solve optimally (Yu and LaValle 2013b; Ma et al. 2016b). It can be solved via reductions to other well-studied combinatorial problems (Surynek 2015; Yu and LaValle 2013a; Erdem et al. 2013) or by dedicated optimal, bounded-suboptimal, or suboptimal MAPF algorithms (Standley and Korf 2011; Goldenberg et al. 2014; Sharon et al. 2013; Wagner 2015; Sharon et al. 2015; Boyarski et al. 2015; Cohen et al. 2016; Silver 2005; Sturtevant and Buro 2006; Luna and Bekris 2011; de Wilde, ter Mors, and Witteveen 2013; Wang and Botea 2011). Many MAPF algorithms have been used on maps from video games (Silver 2005). See (Ma et al. 2016a; Felner et al. 2017) for longer surveys on MAPF algorithms.

MAPF has recently been generalized in different directions (Hönig et al. 2016a; Ma et al. 2016a; Hönig et al. 2016b; Ma, Kumar, and Koenig 2017; Ma and Koenig 2016; Ma et al. 2017). Target Assignment and Path Finding (TAPF) is a variant of MAPF that allows agents in the same group to interchange their goal locations (Ma and Koenig 2016) and thus applies to non-homogeneous teams. During execution, one can maintain user-specified safety distances between agents and adhere to their kinematic constraints (Hönig et al. 2016a; Hönig et al. 2016b).

We use Conflict-Based Min-Cost-Flow (CBM) (Ma and Koenig 2016), a state-of-the-art optimal TAPF algorithm. CBM is a two-level algorithm that minimizes the makespan (that is, the earliest time when all agents reach their goal locations). On the upper level, CBM performs a best-first search on a collision tree and resolves collisions between agents in different groups. Each high-level node contains a set of constraints and a path for each agent that obeys these constraints. On the lower level, CBM uses a polynomial-time min-cost max-flow algorithm (Goldberg and Tarjan 1987) on a time-expanded network.

Each time the player specifies new goal locations, CBM is called to solve a new TAPF instance from the current locations of the agents to their newly specified goal locations. We use cost one for move actions and cost zero for wait actions to avoid unnecessary move actions, such as moving forward and then immediately backward. We also use a large cost for actions that result in collisions between agents in different groups to make CBM more efficient, as discussed in (Ma and Koenig 2016).

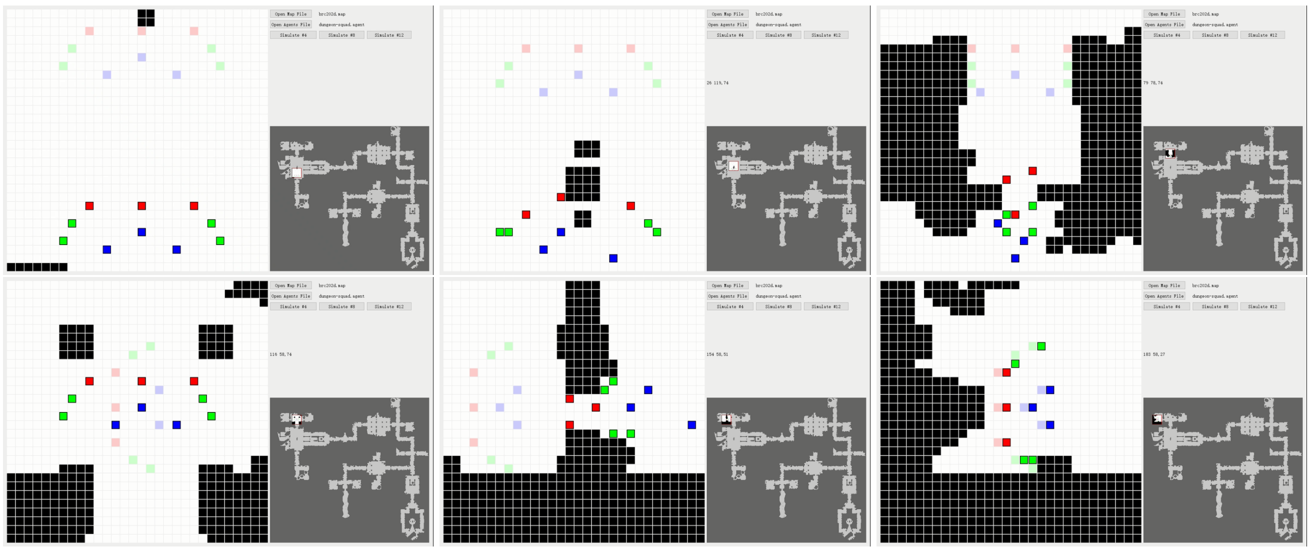

Figure 1: Screenshots of moving ten agents in three groups with a wide goal pattern. New goal locations are specified every twelve time steps. From top-left to bottom-right: (1) agents start; (2) agents avoid obstacles; (3) agents pass through a narrow passageway; (4) agents turn (since the goal pattern was rotated by ninety degrees); (5) agents pass through another narrow passageway; and (6) agents reach their final goal locations.

Table 1: Experimental results.

instance	makespan	CBM calls	average running time (s)
wide-4	193	50	0.146
wide-8	190	25	0.202
wide-12	184	17	0.153
narrow-4	158	40	0.139
narrow-8	157	20	0.139
narrow-12	156	14	0.151

Demonstration

We use CBM with ten agents in three groups in the video game environment `brc202d` (Sturtevant 2012) from Dragon Age: Origins on a 2.5GHz Intel Core i5-2450M with 4GB of RAM. CBM plans in a window of size 30 cells by 30 cells around the agents. The window typically contains many rows in the direction of the next goal locations and few rows in the opposite direction. The goal locations are specified manually. They are typically about twenty cells away from the current locations of the agents. We use two goal patterns (that is, layouts of the goal locations), namely (nineteen cells) wide and (seven cells) narrow, and three update frequencies of the goal locations, namely every four, eight, and twelve time steps. Our videos are available at `http://idm-lab.org/bib/abstracts/Koen17k.html`.

Table 1 reports the makespan, the number of calls to CBM, and the average running time per call (in seconds) for each combination of width of the goal pattern and update frequency of the goal locations. Figure 1 shows screenshots for a wide goal pattern and an update frequency of twelve time steps (called wide-12 in the table). The first group, shown in red, consists of three warriors. The second group, shown in green, consists of four rogues. Finally, the third group, shown in blue, consists of three mages. The goal pattern is shown in a lighter shade of the same color as the agents of the same group. CBM is called every twelve time steps to solve a new TAPF instance from the current locations of the agents to their newly specified goal locations.

Conclusions and Future Work

Our experimental results show that CBM runs sufficiently fast to find collision-free paths for small numbers of agents over short distances in real-time, even in congested video game environments. We are currently working on a system that moves agents in formation over longer distances. Formations, for example, keep the agents safe by making the warriors move in front of the formation, the rogues on both sides, and the mages in the back. However, formations often have to be compromised temporarily in congested video game environments, for example, when agents move through passageways that are narrower than their formation. In our demonstration, the goal patterns correspond to the above formation, yet the agents do not always restore the formation during execution immediately when possible, which is an issue that our system will address. Our system will use swarm-based approaches in simple parts of the video game environment (because they run fast) and switch to CBM in congested parts (because swarm-based approaches fail in them), where it automatically determines appropriate intermediate goal locations that trade off between a small running time, a small makespan (that is, short paths) and a close adherence to the player-specified formation (where the tradeoff can be specified by the player as well).

References

[Boyarski et al. 2015] Boyarski, E.; Felner, A.; Stern, R.; Sharon, G.; Tolpin, D.; Betzalel, O.; and Shimony, S. E. 2015. ICBS: Improved conflict-based search algorithm for multi-agent pathfinding. In *International Joint Conference on Artificial Intelligence*, 740–746.

[Cohen et al. 2016] Cohen, L.; Uras, T.; Kumar, T. K. S.; Xu, H.; Ayanian, N.; and Koenig, S. 2016. Improved solvers for bounded-suboptimal multi-agent path finding. In *International Joint Conference on Artificial Intelligence*, 3067–3074.

[de Wilde, ter Mors, and Witteveen 2013] de Wilde, B.; ter Mors, A. W.; and Witteveen, C. 2013. Push and rotate: Cooperative multi-agent path planning. In *International Conference on Autonomous Agents and Multi-Agent Systems*, 87–94.

[Erdem et al. 2013] Erdem, E.; Kisa, D. G.; Oztok, U.; and Schueller, P. 2013. A general formal framework for pathfinding problems with multiple agents. In *AAAI Conference on Artificial Intelligence*, 290–296.

[Felner et al. 2017] Felner, A.; Stern, R.; Shimony, E.; Boyarski, E.; Goldenerg, M.; Sharon, G.; Sturtevant, N. R.; Wagner, G.; and Surynek, P. 2017. Search-based optimal solvers for the multi-agent pathfinding problem: Summary and challenges. In *Annual Symposium on Combinatorial Search*.

[Goldberg and Tarjan 1987] Goldberg, A. V., and Tarjan, R. E. 1987. Solving minimum-cost flow problems by successive approximation. In *Annual ACM Symposium on Theory of Computing*, 7–18.

[Goldenberg et al. 2014] Goldenberg, M.; Felner, A.; Stern, R.; Sharon, G.; Sturtevant, N. R.; Holte, R. C.; and Schaeffer, J. 2014. Enhanced Partial Expansion A*. *Journal of Artificial Intelligence Research* 50:141–187.

[Hönig et al. 2016a] Hönig, W.; Kumar, T. K. S.; Cohen, L.; Ma, H.; Xu, H.; Ayanian, N.; and Koenig, S. 2016a. Multi-agent path finding with kinematic constraints. In *International Conference on Automated Planning and Scheduling*, 477–485.

[Hönig et al. 2016b] Hönig, W.; Kumar, T. K. S.; Ma, H.; Ayanian, N.; and Koenig, S. 2016b. Formation change for robot groups in occluded environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 4836–4842.

[Luna and Bekris 2011] Luna, R., and Bekris, K. E. 2011. Push and Swap: Fast cooperative path-finding with completeness guarantees. In *International Joint Conference on Artificial Intelligence*, 294–300.

[Ma and Koenig 2016] Ma, H., and Koenig, S. 2016. Optimal target assignment and path finding for teams of agents. In *International Conference on Autonomous Agents and Multiagent Systems*, 1144–1152.

[Ma et al. 2016a] Ma, H.; Koenig, S.; Ayanian, N.; Cohen, L.; Hönig, W.; Kumar, T. K. S.; Uras, T.; Xu, H.; Tovey, C.; and Sharon, G. 2016a. Overview: Generalizations of multi-agent path finding to real-world scenarios. In *IJCAI-16 Workshop on Multi-Agent Path Finding*.

[Ma et al. 2016b] Ma, H.; Tovey, C.; Sharon, G.; Kumar, T. K. S.; and Koenig, S. 2016b. Multi-agent path finding with payload transfers and the package-exchange robot-routing problem. In *AAAI Conference on Artificial Intelligence*, 3166–3173.

[Ma et al. 2017] Ma, H.; Li, J.; Kumar, T. K. S.; and Koenig, S. 2017. Lifelong multi-agent path finding for online pickup and delivery tasks. In *International Conference on Autonomous Agents and Multiagent Systems*, 837–845.

[Ma, Kumar, and Koenig 2017] Ma, H.; Kumar, T. K. S.; and Koenig, S. 2017. Multi-agent path finding with delay probabilities. In *AAAI Conference on Artificial Intelligence*, 3605–3612.

[Sharon et al. 2013] Sharon, G.; Stern, R.; Goldenberg, M.; and Felner, A. 2013. The increasing cost tree search for optimal multi-agent pathfinding. *Artificial Intelligence* 195:470–495.

[Sharon et al. 2015] Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2015. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence* 219:40–66.

[Silver 2005] Silver, D. 2005. Cooperative pathfinding. In *Artificial Intelligence and Interactive Digital Entertainment*, 117–122.

[Standley and Korf 2011] Standley, T. S., and Korf, R. E. 2011. Complete algorithms for cooperative pathfinding problems. In *International Joint Conference on Artificial Intelligence*, 668–673.

[Sturtevant and Buro 2006] Sturtevant, N. R., and Buro, M. 2006. Improving collaborative pathfinding using map abstraction. In *Artificial Intelligence and Interactive Digital Entertainment*, 80–85.

[Sturtevant 2012] Sturtevant, N. R. 2012. Benchmarks for grid-based pathfinding. *Transactions on Computational Intelligence and AI in Games* 4(2):144–148.

[Surynek 2015] Surynek, P. 2015. Reduced time-expansion graphs and goal decomposition for solving cooperative path finding sub-optimally. In *International Joint Conference on Artificial Intelligence*, 1916–1922.

[Wagner 2015] Wagner, G. 2015. *Subdimensional Expansion: A Framework for Computationally Tractable Multi-robot Path Planning*. Ph.D. Dissertation, Carnegie Mellon University.

[Wang and Botea 2011] Wang, K., and Botea, A. 2011. MAPP: a scalable multi-agent path planning algorithm with tractability and completeness guarantees. *Journal of Artificial Intelligence Research* 42:55–90.

[Yu and LaValle 2013a] Yu, J., and LaValle, S. M. 2013a. Planning optimal paths for multiple robots on graphs. In *IEEE International Conference on Robotics and Automation*, 3612–3617.

[Yu and LaValle 2013b] Yu, J., and LaValle, S. M. 2013b. Structure and intractability of optimal multi-robot path planning on graphs. In *AAAI Conference on Artificial Intelligence*, 1444–1449.