State of the Art on: Multi Agent Path Finding

Matteo Bellusci, matteo.bellusci@mail.polimi.it

1. Introduction to the research topic

The *Multi-Agent Path Finding* (MAPF) topic lies at the intersection between *Artificial Intelligence* (AI), *Autonomous Agents and Multi-Agent Systems* (AAMAS), and *Autonomous Robotics* areas. In this region, we deal with intelligent agents, namely entities able to perform, by taking movement actions in a physical environment, tasks that normally require human intelligence. In general, researchers in this area study how autonomous cooperative agents can accomplish different kinds of tasks in a efficient way. The robotics contribution is due to the fact that agents are physical robots interacting with the world in several domains, like in warehouse logistics.

The main AI conferences are the *Association for the Advancement of Artificial Intelligence conference* (AAAI) and the *International Joint Conference on Artificial Intelligence* (IJCAI). The *International Conference on Autonomous Agents and Multiagent Systems* (AAMAS) covers the multi-agent area. Different works relative to MAPF have been discussed in the *Artificial Intelligence and Interactive Digital Entertainment* (AIIDE), a conference sponsored by the AAAI, and in the *International Symposium on Combinatorial Search* (SoCS). Similar topics are also discussed in the *International Conference on Robotics and Automation* (ICRA), in the *International Conference on Intelligent Robots and Systems* (IROS), and, secondly, in the *International Conference on Interactive Collaborative Robotics* (ICR).

The main AI journal is the *Artificial Intelligence Journal* (AIJ). The topic is also covered in different main Robotics journals: *IEEE Transactions on Robotics* (T-RO), *IEEE Robotics and Automation Letters* (RA-L), *IEEE Transactions on Computational Intelligence and AI in Games* (T-CIAIG), *Autonomous Robots* (AURO), and *Robotics and Autonomous Systems* (RAS).

1.1. Preliminaries

To understand the topic and the functioning of the main algorithms shown in next section, some basic rudiments of Graph Theory and AI techniques are needed.

Definition 1.1. A graph $G = (V, E)$ is a mathematical structure that consists of a set of vertices, or nodes, V and a set of edges $E \subseteq V^2$, which are unordered pairs of vertices.

Definition 1.2. A vertex v_i is said to be neighbor, or adjacent, of a vertex v_j in some graph if v_i is connected to v_j through an edge.

We introduce now a well-known algorithm in AI named A* (pronounced "A-star") [8]. A* is a widely used informed heuristic search algorithm in path planning and graph traversals. In its general form, A* is formulated in terms of weighted graphs: given a start node and a goal node, it aims to find the cheapest path between them. The idea is to rank all nodes in the frontier, i.e., the nodes that are already reached but not yet expanded, according to a function $f(n)$ and perform a best-first search. In A*, $f(n) = g(n) + h(n)$, where $g(n)$ is the cost from the start node to n, and $h(n)$ represents the *heuristic* estimated cost from node n to the goal node. The search quality of A* is strictly correlated to the accuracy of the heuristic estimate $h(n)$.

Definition 1.3. The heuristic function $h(n)$ is said to be admissible if $h(n)$ is always less than or equal to true cheapest cost from n to the goal.

Definition 1.4. The heuristic function $h(n)$ is said to be consistent if $h(n)$ is always less than or equal to the sum of the estimated distance from any neighboring node to the goal node and of the cost of reaching that neighbor.

Lemma 1.5. If the heuristic function $h(n)$ is consistent and $h(goal) = 0$, then it is also admissible.

Hart et al. [8] provided proofs of completeness and optimality. When performing tree search or graph search possibly re-exploring repeated nodes, if $h(n)$ is admissible then A* finds an optimal solution. Moreover, when performing graph search without re-exploring repeated nodes, if $h(n)$ is consistent then A* finds an optimal solution. For a complete reference about A* and path planning, see [21, 2].

1.2. Research topic

The task in MAPF problems is to find non-interfering paths for multiple agents, each one with a unique start and goal position. An instance of the MAPF problem is formally a tuple $\langle G, A, \{s_i\}, \{g_i\}\rangle$ where $G = (V, E)$ is a graph and $A = \{a_1, ..., a_k\}$ is a set of k labeled agents and each agent a_i has an associate start position $s_i \in V$ and goal position $g_i \in V$. Time is discretized into timesteps. At initial timestep t_0 each agent is located in its start position. At each timestep, every agent can either change location moving to an adjacent vertex or wait at its current position. Each action has an associate unit cost, except for the wait action if the agent has already reached its goal.

A *path* π_i for an agent a_i can be modeled as a sequence of vertices $(p_{t_0}, ..., p_{t_l})$ which brings a_i from $p_{t_0} = s_i$ to $p_{t_l} = g_i$. Each element of the sequence corresponds to a timestamp, starting from t_0. A *solution* is just a set of k paths. The task is to find a *valid solution*, i.e. a set of k non-conflicting paths, while minimizing a cumulative cost function. Common cost functions are the *flowtime*, defined as the sum of the individual path costs (SIC), and the *makespan*, defined as the maximum of the individual path costs.

In general two types of conflicts are considered. A *vertex conflict* is a tuple $\langle a_i, a_j, v, t\rangle$ meaning that agents a_i and a_j are occupying the same vertex v at timestamp t. An *edge conflict* is a tuple $\langle a_i, a_j, v, v', t\rangle$ meaning that from timestamp t to $t+1$ agent a_i is traveling from v to v' while a_j is traveling from v' to v, i.e., they are traveling along an edge in opposite directions. In this sense a valid solution is a conflict-free solution.

MAPF is a challenging problem with several practical applications. For instance, MAPF is at the core of the Kiva (Amazon Robotics) system [28], which has been successful in the marketplace and has brought innovation in the warehouse logistics field. The Kiva system involves small autonomous robots to lift movable storage shelves. Furthermore, MAPF can model numerous real-world problems in office robots [26], computer video games [13], aircraft-towing vehicles [14] and in many other domains.

2. Main related works

2.1. Classification of the main related works

Many different types of algorithms are available for solving MAPF problems. In this section we show the two main dimensions to classify MAPF solvers. The first dimension is related to solution quality provided by the algorithm.

- *Incomplete* solvers are generally really fast, but don't guarantee to find a solution, even if it exists.
- *Complete* solvers guarantee to find always a solution if it exists, but not necessarily the best one (according to flowtime, makespan).
- *Optimal* solvers are usually the slowest, but they guarantee to find the best (according to flowtime, makespan) solution if the problem is solvable.

The second dimension is related to the general idea behind the algorithm. In fact, the are three major approaches for solving MAPF:

- *Search-based* solvers search the solution in a specific search space. Broadly speaking, they are designed to deal with a flowtime solution measure.

- *Procedure-based* or *rule-based* solvers select actions for the agents following specific movement rules. In general they are complete, very fast and can solve very large problems, but they provide solutions that are often far from optimal.
- *Reduction-based* solvers translate the problem to another formalism exploiting knowledge of other problems. They are typically designed to deal with a makespan solution measure.

The following is a summary table in which we have classified the main related works.

Classification of the main related works

	Search-based	Procedure-based	Reduction-based
Incomplete	[21][18]		
Complete		[10][23][11][3]	
Optimal	[20][27][7][15][16]		[24][29][5]

2.2. Brief description of the main related works

Many algorithms have been proposed in recent years. We briefly describe some of the main relevant approaches and techniques, illustrating their strengths and limits. Complete surveys are also available [12, 6].

2.2.1 A*-based Algorithms

A-based Algorithms* involve the use of A* [8]. Basically they perform a state-space search in which the states are the locations of the agents and transitions are related to the joint actions of the agents. A common and simple admissible heuristic is the *Euclidean distance*. The main issue of this approach is related to the branching factor that may be exponential in the number of agents. However, several improvements are possible. For instance, M* [27] dynamically mutates the branching factor based on conflicts, while *Enhanced Partial Expansion A** (EPEA*) [7] tries to avoid the generation of *surplus* nodes. In general, A*-based algorithms outperform other approaches in areas dense with agents.

2.2.2 Operator Decomposition

The *Operator Decomposition* (OD) [20] aims to overcome the high branching factor and the generation of surplus nodes. The idea is to decompose the standard operator for obtaining the next state into a sequence of operators for individual agents. Hence, in the state-space search the transitions are related to the actions of a single agent. This leads in considering *intermediate* states where not all the agents have selected their move. A *standard* node is reached when all the agents have selected their move. Once the solution is found, non-expanded intermediate nodes are not developed into standard nodes, reducing the number of surplus nodes.

2.2.3 Independence Detection

The *Independence Detection* (ID) [20] framework pursues the idea to split the problem into a series of (smaller) sub-problems detecting independent groups of agents. Two groups of agents are *independent* if there is an optimal solution for each group such that no conflicts exists between the two solutions. Thus, the ID framework acts as a reducer for the number of agents. The *Simple Independence Detection* (SID) algorithm is the simplest approach, it basically merges two group of agents when a conflict arise between their solutions.

2.2.4 Conflict-Based Search

The *Conflict-Based Search* (CBS) [15] algorithm is a two-level complete and optimal MAPF solver. The low-level searches for an optimal path for each agent individually, generally using an A*-based procedure, taking into account constraints imposed by the high-level. When a conflict is found between individual paths the high-level expands a constraint tree via a split action, keeping all information about the conflict, and imposes new constraints to the agents in order to avoid the conflict. The idea is to try to solve the conflict in all the possible ways. A constraint essentially prohibits an agent from being in a certain position at a given time. The algorithm is exponential in the number of conflicts. An improved version of CBS is available with the name of *Improved Conflict-Based Search* (ICBS) [1]. In general, CBS-based solvers generally outperform other approaches in environments with many bottlenecks.

2.2.5 Increasing Cost Tree Search

The *Increasing Cost Tree Search* (ICTS) [16] algorithm is another example of two-level solver. Unlike CBS, which plans for single agents under constraints, ICTS works in a k-agent search space, like A*. The high level, searching the *increasing cost tree* (ICT), asks the low level to find a set of k non-interfering paths, each with a specific length. A compact data-structure called *multi-value decision diagram* (MDD) [19] is involved to store all single-agent paths of a certain length, for each agent. At each request the high level requires different specific path lengths, starting from the best case in order to guarantee optimality. The algorithm halts when the low level finds such a valid solution. The solver is exponential in Δ, defined as the depth of the lowest cost ICT goal node. However, there exists different pruning techniques for the high-level [17]. For instance, ICTS+3E is an enhanced version of ICTS in which we use information about small groups of up to 3 agents and their internal conflicts. In general, ICTS-based algorithms outperform other approaches in open areas with few agents.

2.2.6 SAT-based Algorithms

SAT-based solvers [24] are reduction-based approaches. They exploit the properties of the Boolean satisfiability problem. The idea is to use layered (temporally extended) graphs to encode plans of a known length and use multi-valued state variables to encode the position of the agents in the layers. Generally, they are adopted for optimizing the makespan of a sub-optimal solution, and not for solving the entire problem instance. However, Surynek et al. [25] recently introduced a reduction-based SAT solver for the flowtime variant. An improved variant called MDD-SAT, employ the use of MDDs to reduce the number of propositional variables in SAT formulas. In general, search-based methods are faster for easier problems while SAT methods are faster for harder problems.

2.2.7 Other Works

There are also other relevant works. For instance, Sturtevant [22] has made available a standard test set of grid-based maps and problems on the maps for benchmark, making experimental results more comparable across papers.

In Figure 1 we show three of the most frequently used test maps that are taken from the *Dragon Age: Origins* (DAO) game. The den520d map (left) is used to test the performance of the algorithms in an environment with wide open spaces. On the contrary, the brc202d map (right) is used to test the performance of the algorithms in an environment with narrow corridors and bottlenecks. Finally, the ost003d map (middle) is a open space with almost isolated rooms and lies, intuitively, in the middle between the other two types of environments.

2.3. Discussion

Optimal single agent path finding is tractable (e.g., using the Dijkstra's algorithm [4]) while optimal (according to flowtime, makespan) MAPF is NP-hard [30]. The choice of the algorithm to apply depends on various factors related to the instance of the problem. Felner et al. [6] showed an interesting comparison between the main approaches. The results emphasize that there is no universal winner. However, as the authors confirm, a more systematic comparison between existing solvers is needed in order to understand which algorithm performs better

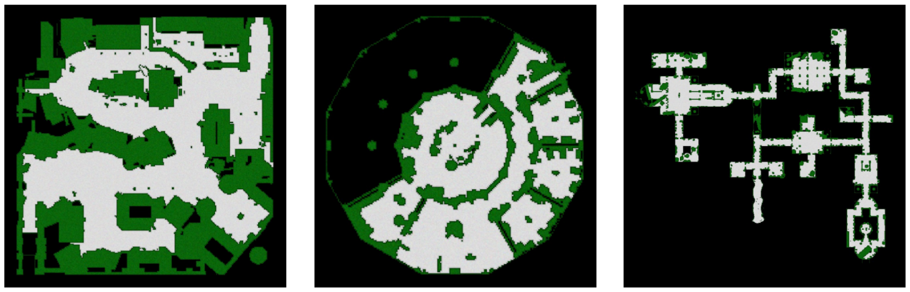

Figure 1: DAO maps den520d (left), ost003d (middle) and brc202d (right)
.

given the initial problem settings.

Moreover, when generalizing MAPF to real-world scenarios different kinds of issues arise. According to Felner et al. [6], more research should be done to adapt existing MAPF solvers to real-world domains. In fact, Ma et al. [12] have highlighted that improving existing solvers or developing new approaches for classical MAPF is insufficient because, in several real-world scenarios, new domain properties (e.g., uncertainty in actions) are required. For instance, classical MAPF formulation ignores the fact that robot movements are subject to kinematic constraints [9]. Furthermore, in classical MAPF problems, we make strong assumptions concerning the environment. Real-world environments may have different properties, while classical MAPF contemplates only a fixed discretized environment.

References

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

[2] Cui, X., and Shi, H. A*-based pathfinding in modern computer games. *International Journal of Computer Science and Network Security 11*, 1 (2011), 125–130.

[3] De Wilde, B., Ter Mors, A. W., and Witteveen, C. Push and rotate: Cooperative multi-agent path planning. In *Proceedings of the International Conference on Autonomous Agents and Multi-agent Systems* (2013), pp. 87–94.

[4] Dijkstra, E. W. A note on two problems in connexion with graphs. *Numerische mathematik 1*, 1 (1959), 269–271.

[5] Erdem, E., Kisa, D. G., Öztok, U., and Schüller, P. A general formal framework for pathfinding problems with multiple agents. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2013), pp. 290–296.

[6] Felner, A., Stern, R., Shimony, S. E., Boyarski, E., Goldenberg, M., Sharon, G., Sturtevant, N., Wagner, G., and Surynek, P. Search-based optimal solvers for the multi-agent pathfinding problem: Summary and challenges. In *Proceedings of the Symposium on Combinatorial Search* (2017).

[7] Goldenberg, M., Felner, A., Stern, R., Sharon, G., Sturtevant, N., Holte, R. C., and Schaeffer, J. Enhanced partial expansion A*. *Journal of Artificial Intelligence Research 50* (2014), 141–187.

[8] Hart, P. E., Nilsson, N. J., and Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics 4*, 2 (1968), 100–107.

[9] Hönig, W., Kumar, T. S., Cohen, L., Ma, H., Xu, H., Ayanian, N., and Koenig, S. Multi-agent path finding with kinematic constraints. In *Proceedings of the International Conference on Automated Planning and Scheduling* (2016).

[10] Kornhauser, D., Miller, G., and Spirakis, P. Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. In *Proceedings of the Annual Symposium on Foundations of Computer Science* (1984), pp. 241–250.

[11] Luna, R., and Bekris, K. E. Push and swap: Fast cooperative path-finding with completeness guarantees. In *Proceedings of the International Joint Conference on Artificial Intelligence* (2011), pp. 294–300.

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

[13] Ma, H., Yang, J., Cohen, L., Kumar, T. K. S., and Koenig, S. Feasibility study: Moving non-homogeneous teams in congested video game environments. In *Proceedings of the Artificial Intelligence and Interactive Digital Entertainmen* (2017), pp. 270–272.

[14] Morris, R., Pasareanu, C. S., Luckow, K., Malik, W., Ma, H., Kumar, T. S., and Koenig, S. Planning, scheduling and monitoring for airport surface operations. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2016).

[15] Sharon, G., Stern, R., Felner, A., and Sturtevant, N. Conflict-based search for optimal multi-agent path finding. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2012), pp. 563–569.

[16] Sharon, G., Stern, R., Goldenberg, M., and Felner, A. The increasing cost tree search for optimal multi-agent pathfinding. In *Proceedings of the International Joint Conference on Artificial Intelligence* (2011), pp. 662–667.

[17] Sharon, G., Stern, R. T., Goldenberg, M., and Felner, A. Pruning techniques for the increasing cost tree search for optimal multi-agent pathfinding. In *Proceedings of the Annual Symposium on Combinatorial Search* (2011).

[18] Silver, D. Cooperative pathfinding. In *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment* (2005), pp. 117–122.

[19] Srinivasan, A., Ham, T., Malik, S., and Brayton, R. K. Algorithms for discrete function manipulation. In *Proceedings of the IEEE International Conference on Computer-aided Design. Digest of Technical Papers* (1990), pp. 92–95.

[20] Standley, T. Finding optimal solutions to cooperative pathfinding problems. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2010), pp. 173–178.

[21] Stout, B. Smart moves: Intelligent pathfinding. *Game Developer Magazine 10* (1996), 28–35.

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

[23] Surynek, P. A novel approach to path planning for multiple robots in bi-connected graphs. In *Proceedings of the International Conference on Robotics and Automation* (2009), pp. 3613–3619.

[24] Surynek, P. Towards optimal cooperative path planning in hard setups through satisfiability solving. In *Proceedings of the The Pacific Rim International Conference on Artificial Intelligence* (2012), pp. 564–576.

[25] Surynek, P., Felner, A., Stern, R., and Boyarski, E. Efficient sat approach to multi-agent path finding under the sum of costs objective. In *Proceedings of the European Conference on Artificial Intelligence* (2016), pp. 810–818.

[26] Veloso, M., Biswas, J., Coltin, B., and Rosenthal, S. Cobots: Robust symbiotic autonomous mobile service robots. In *Proceedings of the International Conference on Artificial Intelligence* (2015), pp. 4423–4429.

[27] Wagner, G., and Choset, H. M*: A complete multirobot path planning algorithm with performance bounds. In *Proceedings of the International Conference on Intelligent Robots and Systems* (2011), pp. 3260–3267.

[28] Wurman, P. R., D'Andrea, R., and Mountz, M. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2007), pp. 1752–1759.

[29] Yu, J., and LaValle, S. M. Planning optimal paths for multiple robots on graphs. In *Proceedings of the International Conference on Robotics and Automation* (2013), pp. 3612–3617.

[30] Yu, J., and LaValle, S. M. Structure and intractability of optimal multi-robot path planning on graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2013), pp. 1443–1449.