Modelling Multi-Agent Pathfinding Problems by Integrating Connectivity and No-Collision Constraints (Extended Abstract)

Jiří Švancara[1], **Roman Barták**[1], **Ian Miguel**[2], **Joan Espasa**[2]
[1]Charles University, Czech Republic
[2]University of St Andrews, United Kingdom
{svancara,bartak}@ktiml.mff.cuni.cz, {ijm,jea20}@st-andrews.ac.uk

Extended version — https://ifaamas.org/Proceedings/aamas2026/pdfs/BHIZ5140.pdf (Švancara et al. 2026)
Code — https://github.com/svancaj/MAPF-encodings

Introduction

Multi-agent pathfinding (MAPF) is the task of navigating a set of mobile agents from their starting locations to their desired goal locations while avoiding collisions (Stern et al. 2019). It has many practical applications, most notably in warehousing. Finding an optimal solution is NP-hard under standard cost objectives (Surynek 2015). A popular approach is to reduce the problem to a well-established formalism such as SAT and invoke a high-performance solver. This is a strong approach, particularly in densely populated environments. However, encoding graph-level constraints – most critically, the requirement that each agent follow a *connected path* from start to goal – is notoriously difficult in propositional logic, since connectivity is an inherently transitive property that does not map naturally to CNF clauses.

This paper aims to replace the path-existence constraints in standard SAT encodings with native graph-theory propagators provided by MonoSAT, an SMT solver for monotonic theories (Bayless et al. 2015). Collision avoidance constraints remain expressed as a Boolean formula.

Pure SAT Encodings

To model MAPF via SAT, Boolean variables encoding the position and transition of each agent are created. The correct movement and conflict avoidance is enforced by CNF clauses. Two well-known variable families serve as the foundation:

Pass (P) (Barták and Švancara 2019) Boolean variables $At(a_i, v, t)$ and $Pass(a_i, u, v, t)$ track, per agent a_i, which vertices v are occupied and which edges (u, v) are traversed in each time-step t. Path existence is enforced by movement constraints linking At and $Pass$ variables across time-steps; conflict avoidance is encoded as binary clauses.

Shift (S) (Achá et al. 2022) A more compact alternative keeps the At variables and replaces $Pass$ variables with agent-independent variables $\mathit{Shift}(u, v, t)$ to represent that *something* traverses edge (u, v) at time t. Additional constraints couple these movements to specific agents. Because Shift variables are not per-agent, the encoding is smaller, but connectivity is still expressed via explicit movement clauses.

A limit on the number of time-steps is determined. In the case of a successful call to the solver, the found solution is optimal; otherwise, the limit is increased. This method guarantees finding the makespan optimal solution. For sum of costs optimization, additional $Late(a_i, t)$ variables are introduced to count the number of time-steps an agent has not yet permanently settled at its goal. An *at-most-k* cardinality constraint bounds the total extra cost. All encodings exploit MDD-based reachability preprocessing (Sharon et al. 2011) – variables and clauses are only created for positions that a given agent can reach within the cost bound.

Graph-Theory Encodings

The key insight is to model each agent's trajectory as a path in a *time-expanded graph* (TEG) — a layered copy of the input graph (environment) in which edges connect vertex v at layer t to vertex u at layer $t+1$ for each edge (v, u) in the original graph. MonoSAT allows the user to associate Boolean variables with edges in a graph and to post *s-t connectivity constraints* whose satisfaction is determined by an efficient graph propagator rather than by CNF clauses.

Pass+Graph (PG) A separate TEG is created per agent. The At and $Pass$ variables are associated with TEG vertices and edges, respectively. A connectivity constraint from the starting location in the first layer to the goal location in the last layer is posted for each agent and enforced by a unit clause. This single constraint replaces movement constraints, leaving only the binary conflict clauses in the Boolean part of the encoding. Phantom agents — disconnected variable assignments set to true — are harmless because the connectivity constraint guarantees at least one valid path, and post-processing via DFS extracts it.

Shift+Graph (SG) Inspired by the Shift encoding, a *single* TEG is shared by all agents, with Shift variables on its edges. One connectivity constraint per agent is posted over this shared graph. Non-overlapping paths are enforced by *at-most-one* constraints on incoming and outgoing Shift variables at each vertex, which also implicitly prevents vertex

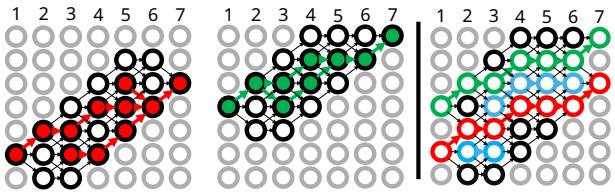

Figure 1: Solution example on a map of 7 vertices (vertical) in 7 time-steps (horizontal) using PG (2 TEGs on the left) and SG (1 TEG on the right). Greyed vertices are unreachable. In the first and last time-step, only the start and goal vertices are reachable, respectively. Red and green correspond to agents or connected phantom agents. Blue vertices are disjoint phantom agents.

conflicts. Swapping conflicts are forbidden by binary clauses as before. Phantom agents are able to appear, but can again be handled by post-processing via DFS, as they are guaranteed to be disconnected from the paths of the *real* agents.

The optimization for makespan remains the same. Optimizing sum of costs with the presence of phantom agents requires care: in PG, lateness is detected by checking whether the agent occupies any non-goal vertex; in SG, lateness is inferred from whether movement into the goal occurs after the shortest-path distance. An example of a solution can be seen in Figure 1.

Empirical Evaluation

Experiments were run on maps inspired by the standard benchmark set (Stern et al. 2019) with sizes increasing from 20×20 to 100×100 and 3 game maps. Instances begin with 5 agents, 5 more are added until unsolvable within a 5-minute CPU limit. Five different start/goal placements were used per instance. We report the key findings below and in Table 1.

Pass+Graph is the top performer overall Compared with its pure-SAT counterpart *Pass*, it solves more than *double* the instances under makespan optimization and roughly *25% more* under sum of costs — consistently across map sizes.

Formula size is dramatically reduced The movement clauses delegated to the graph propagator account for the bulk of clause counts in the pure SAT encodings. In turn, the building time of the encoding is reduced.

Shift+Graph underperforms expectations While *Shift* outperforms *Pass* in pure SAT, adding graph propagators to the *Shift* encoding moderately improves makespan results but *degrades* sum of costs performance relative to pure *Shift*. We conjecture that enforcing disjoint paths over a single compact graph is harder for the propagator than handling separate per-agent TEGs; this is an important future work.

MonoSAT vs. Kissat Running the pure SAT encodings on the state-of-the-art SAT solver Kissat (Biere et al. 2024) yields more solved instances than MonoSAT on those same encodings, demonstrating the gap in solver maturity. Nevertheless, PG with the older MonoSAT still outperforms the best pure SAT encoding on Kissat, underscoring the strength of the graph-propagator approach independently of solver quality. Combining graph theories with a modern SAT backend is a clear direction for future gains.

		20	40	60	80	100	brc	den	ost	total
mks	*P*	380	77	16	4	1	0	0	0	478
	S	402	133	25	4	1	0	0	0	565
	PG	392	**353**	**227**	**131**	**91**	**1**	**4**	**6**	**1205**
	SG	207	227	172	113	71	0	3	5	798
soc	*P*	123	207	215	271	236	29	68	47	1196
	S	139	220	224	273	235	25	64	47	1227
	PG	132	**238**	**290**	**324**	**347**	**36**	**77**	**56**	**1500**
	SG	**140**	188	173	202	187	20	47	36	993

Table 1: The number of solved instances per map size.

Acknowledgments

Research supported by the Czech Science Foundation Grant No. 23-05104S, by the Czech-Israeli Cooperative Scientific Research Project LUAIZ24104, and by Charles University project UNCE 24/SCI/008.

References

Achá, R. A.; López, R.; Hagedorn, S.; and Baier, J. 2022. Multi-Agent Path Finding: A New Boolean Encoding. *JAIR*, 75: 323–350.

Barták, R.; and Švancara, J. 2019. On SAT-Based Approaches for Multi-Agent Path Finding with the Sum-of-Costs Objective. In *SOCS'19*, 10–17. AAAI Press.

Bayless, S.; Bayless, N.; Hoos, H.; and Hu, A. 2015. SAT Modulo Monotonic Theories. In *AAAI'15*, 3702–3709. AAAI Press.

Biere, A.; Faller, T.; Fazekas, K.; Fleury, M.; Froleyks, N.; and Pollitt, F. 2024. CaDiCaL, Gimsatul, IsaSAT and Kissat Entering the SAT Competition 2024. In *SAT Competition 2024 – Solver, Benchmark and Proof Checker Descriptions*, volume B-2024-1, 8–10. University of Helsinki.

Sharon, G.; Stern, R.; Goldenberg, M.; and Felner, A. 2011. The Increasing Cost Tree Search for Optimal Multi-Agent Pathfinding. In *IJCAI'11*, 662–667. IJCAI/AAAI Press.

Stern, R.; Sturtevant, N.; Felner, A.; Koenig, S.; Ma, H.; Walker, T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, T.; Barták, R.; and Boyarski, E. 2019. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. In *SOCS'19*, 151–159. AAAI Press.

Surynek, P. 2015. On the Complexity of Optimal Parallel Cooperative Path-Finding. *Fundamenta Informaticae*, 137(4): 517–548.

Švancara, J.; Barták, R.; Miguel, I.; and Espasa, J. 2026. Modelling Multi-Agent Pathfinding Problems by Integrating Connectivity and No-Collision Constraints. In *AAMAS '26*, 3686–3694. IFAAMAS.