ICBS: The Improved Conflict-Based Search Algorithm for Multi-Agent Pathfinding: Extended Abstract*

Eli Boyarski
CS Department
Bar-Ilan Univ. (Israel)
eli.boyarski@gmail.com

Ariel Felner, Roni Stern, Guni Sharon
ISE Department
Ben-Gurion Univ (Israel)
felner@bgu.ac.il

Oded Betzalel, David Tolpin, Eyal Shimony
CS Department
Ben-Gurion Univ (Israel)
shimony@cs.bgu.ac.il

Introduction and Overview

A *Multi-Agent Path Finding* (MAPF) problem is defined by a graph, $G = (V, E)$ and a set of k agents labeled $a_1 \dots a_k$, where each agent a_i has a start position $s_i \in V$ and a goal position $g_i \in V$. At each time step an agent can either *move* to an adjacent location or *wait* in its current location. The task is to plan a sequence of move/wait actions for each agent a_i, moving it from s_i to g_i such that agents do not *conflict*, i.e., occupy the same location at the same time.

Conflict-Based Search (CBS) (Sharon et al. 2012), is an effective optimal MAPF solver. CBS has two levels. The low-level finds optimal paths for the individual agents. If the paths conflict, the high level, via a *split* action, imposes constraints on the conflicting agents to avoid these conflicts. Meta-agent CBS (MA-CBS) (Sharon et al. 2015) generalizes CBS by merging groups of agents into meta-agents when beneficial.

In this paper we introduce *Improved-CBS* (ICBS) which further adds three new improvements to MA-CBS.

The Conflict Based Search Algorithm (CBS)

A sequence of individual agent move/wait actions leading an agent from s_i to g_i is referred to as a *path*, and the term *solution* refers to a set of k paths, one for each agent. A *conflict* between two paths is a tuple $\langle a_i, a_j, v, t\rangle$ where agent a_i and agent a_j are planned to occupy vertex v at time point t. A solution is *valid* if it is conflict-free. The *cost* of a path is the number of actions in it (including wait), and the cost of a solution is the sum of the costs of its constituent paths. In CBS, a *constraint* for agent a_i is a tuple $\langle a_i, v, t\rangle$ where agent a_i is prohibited from occupying vertex v at time step t. A *consistent path* for agent a_i is a path that satisfies *all* of a_i's constraints, and a *consistent solution* is a solution composed of only consistent paths.

The high-level of CBS searches the *constraint tree* (CT). The CT is a binary tree, in which each node N contains: **(1)** A set of constraints imposed on the agents ($N.constraints$), **(2)** A single solution ($N.solution$) consistent with these constraints, **(3)** The cost of $N.solution$ ($N.cost$).

The root of the CT contains an empty set of constraints. A successor of a node in the CT inherits the constraints of the parent and adds a single new constraint for a single agent. $N.solution$ is found by the low-level search. A CT node N is a goal node when $N.solution$ is valid, i.e., the set of paths for all agents has no conflicts. The high-level of CBS performs a best-first search on the CT where nodes are ordered by their costs ($N.cost$). Given a CT node N, the low-level search is invoked for individual agents to return an optimal path that is consistent with their individual constraints in N.

Resolving a conflict - the *split* action: Given a non-goal CT node, N, whose solution, $N.solution$, includes a *conflict*, $\langle a_i, a_j, v, t\rangle$, we know that in any valid solution at most one of the conflicting agents, a_i or a_j, may occupy vertex v at time t. Therefore, at least one of the constraints, $\langle a_i, v, t\rangle$ or $\langle a_j, v, t\rangle$, must be satisfied. Consequently, CBS *splits* N and generates two new CT nodes as children of N, each adding one of these constraints to the previous set of constraints, $N.constraints$.

Improving the Merge Operation

Previous work: MA-CBS(B) (Sharon et al. 2015) generalizes CBS by adding the option to *merge* the conflicting agents a_i and a_j into a *meta-agent* instead of the *split* action. A meta-agent is logically treated as a single composite agent, whose state consists of a vector of locations, one for each individual agent. A meta-agent is never split in the subtree of the CT below the current node; it may, however, be merged with other (meta)agents into new meta-agents. The low-level search for a meta-agent is solved with any coupled MAPF solver.

Drawback: Merging agents into meta agents reduces the size of the CT at the cost of higher computational effort by the low-level solver. In MA-CBS this tradeoff is handled locally for each node. As a result, MA-CBS might duplicate much search effort.

ICBS Improvement 1 - Merge and Restart (MR): Had we known that a pair of agents are due to be merged, significant computational effort would have been saved by performing the merge at the root node of the CT. Once a merge decision has been reached inside a CT node N, we suggest to discard the current CT and *restart* the search from a new root node, where these agents are merged into a meta agent at the beginning of the search.

*Extended Abstract, full version has been accepted to ICAPS-2015, IJCAI-2015

Copyright © 2015, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

MR is simple to implement, and saves much of the effort duplicated in the multiple CT nodes. In most cases the gains outweigh the additional overhead. As it is very hard to model these quantities theoretically, we tested MR empirically, and indeed it speeds up the search by a significant factor.

Improving the Split Operation

ICBS Improvement 2 - Prioritize Conflicts (PC): CBS arbitrarily picks a conflict for splitting, and its performance is very sensitive to the conflicts chosen, as these can significantly influence the number of CT nodes.

We distinguish between three types of conflicts: **Cardinal conflict.** A conflict $C = \langle a_1, a_2, v, t \rangle$ is *cardinal* for a CT node N if adding any of the two constraints derived from C ($\langle a_1, v, t \rangle$, $\langle a_2, v, t \rangle$) to N and invoking the low-level on the constrained agent, the cost of its path is increased when compared to its cost in N. A conflict C is **semi-cardinal** for a CT node N if adding one of the two constraints derived from C increases $N.cost$ but adding the other leaves $N.cost$ unchanged. A conflict is **non cardinal** for a CT node N if neither of the constraints derived from C increases $N.cost$.

When a node N with $N.cost = c$ is chosen for expansion by CBS, we examine all its conflicts. If a cardinal conflict is encountered, it is immediately chosen for the split action. This generates two children with cost $> c$. In this case if another node N' in OPEN has cost c, then N' can be immediately chosen for expansion next without further developing nodes below N. If no cardinal conflicts exist, the next preference is to choose semi-cardinal conflicts. Here, we know that at least the cost of one child will immediately increase. If only non-cardinal conflicts exist, one is chosen arbitrarily.

ICBS Improvement 3 - Bypassing Conflicts (BP): CBS is sensitive to the paths found by the low level. BP provides a way to bypass conflicts by modifying the chosen path of one of the agents, thereby preventing a split and reducing the CT.

For each CT node N we use $N.NC$ to denote the total *number of conflicts* of the form $\langle a_i, a_j, v, t \rangle$ between the paths in $N.solution$. A path P_i' is a **valid bypass** to path P_i for agent a_i with respect to a conflict $C = \langle a_i, a_j, v, t \rangle$ and a CT node N, if: (i) Unlike P_i, P_i' does not include conflict C, (ii) $cost(P_i') = cost(P_i)$ and (iii) P_i and P_i' are both consistent with $N.constraints$.

Adopting path P_i' by a CT node N means replacing P_i with a valid bypass P_i' for agent a_i. Adopting a valid bypass may introduce more conflicts compared to the original path and potentially lead to worse overall runtime. Thus, we only allow adopting **helpful bypasses** that reduce $N.NC$.

For a given CT node N, BP peeks at either of the immediate children of N in the CT. If the path of a child includes a *helpful bypass* for the conflict of N, this path is *adopted* by N without splitting N and adding new nodes to the CT. This can potentially save a significant amount of search due to a smaller size CT. Valid bypasses can only be found for semi- or non-cardinal conflicts and not for cardinal conflicts. Therefore if PC found a cardinal conflict BP is skipped.

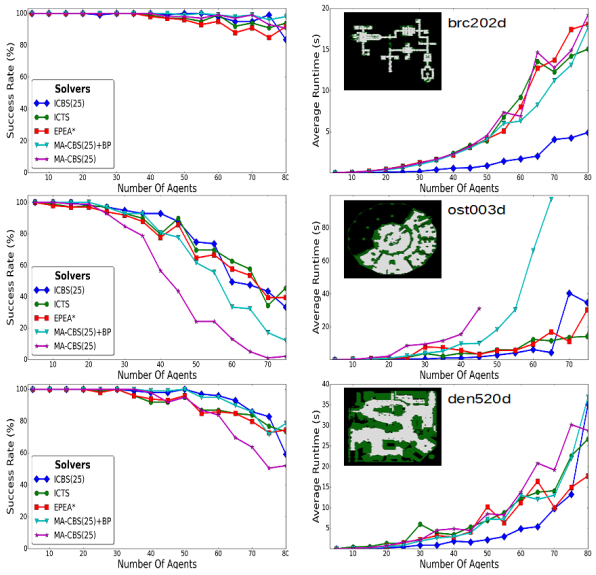

Figure 1: Comparison of all algorithms, DAO maps

Experimental Results

We fine-tuned a number of leading algorithms with the best parameters. We compare ICBS(25) (=MA-CBS(25)+BP+PC+MR), to the following other MAPF solvers: **(1)** MA-CBS(25)+BP, the former best CBS variant. **(2)** MA-CBS(25) **(3)** EPEA* (Goldenberg et al. 2014), an enhanced version of A* designed for cases with large branching factors, the best A* variant. **(4)** ICTS+p (Sharon et al. 2013), the best ICTS variant.

Figure 1 presents the success rate (left) and runtime (right) for the three standard benchmark maps (brc202d:top, ost003d:middle, den520d:bottom) of the game *Dragon Age: Origins* (DAO) (Sturtevant 2012) that were used by Sharon et al. (2013; 2015), 100 instances per map.

In all three maps, ICBS is clearly the best CBS variant. ICBS' runtime is also the best amongst the ones compared here. ICBS is also the best in success rate for brc202d and for den520d, while ICBS, EPEA* and ICTS are tied in the success rate in ost003d .

Conclusions and Future Work

ICBS is the best solver on standard benchmarks, or ties with other solvers. Future work will further study the relations between these enhancements, find a better merge policy than a fixed B parameter and study the pros and cons of the various MAPF algorithms under different circumstances.

References

Goldenberg, M.; Felner, A.; Stern, R.; Sharon, G.; Sturtevant, N. R.; Holte, R. C.; and Schaeffer, J. 2014. Enhanced partial expansion A. *J. Artif. Intell. Res. (JAIR)* 50:141–187.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2012. Conflict-based search for optimal multi-agent path finding. In *AAAI*.

Sharon, G.; Stern, R.; Goldenberg, M.; and Felner, A. 2013. The increasing cost tree search for optimal multi-agent pathfinding. *Artif. Intell.* 195:470–495.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2015. Conflict-based search for optimal multi-agent path finding. *Artificial Intelligence Journal (AIJ)* 218:40–66.

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