Conflict Based Search

Sharon, Guni, et al. "Conflict-based search for optimal multi-agent pathfinding." 2015

Jiří Krejčí

Outline

- MAPF and search-based solvers
- the CBS algorithm
- theoretical analysis
- empirical evaluation
- Meta-agent CBS

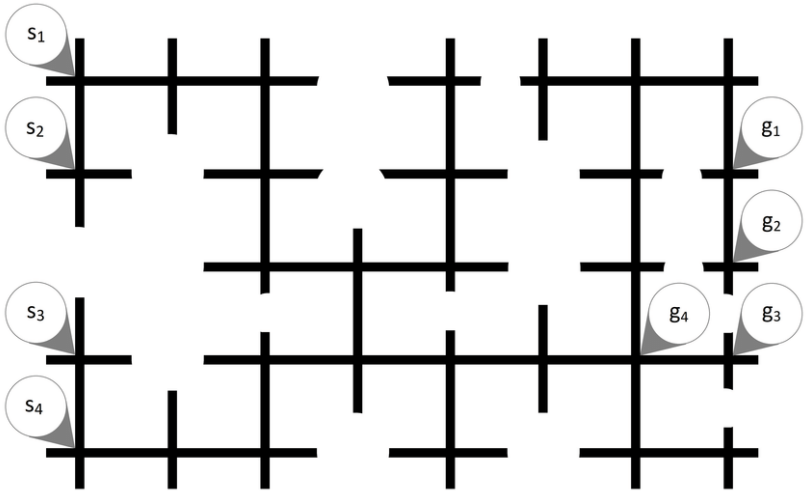
Multi-agent path finding

Input:

- a directed graph
- k agents, each agent has a start vertex and goal vertex

Task:

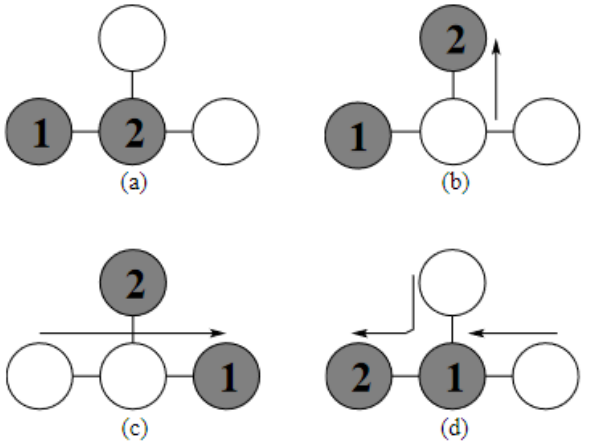
A solution to the MAPF problem is a set of non-conflicting paths, one for each agent, where each agent reaches its goal vertex, beginning at start vertex.



Source: Barták, Roman, Ivan Krasicenko, and Jiří Švancara. "Multi-Agent Path Finding on Ozobots."

Classification of MAPF solvers

- optimal
  - reduction-based solvers (SAT, ILP, ASP)
  - **search-based solvers (A*, CBS)**
- suboptimal
  - search-based (HCA*)
  - rule-based (Push and Swap)



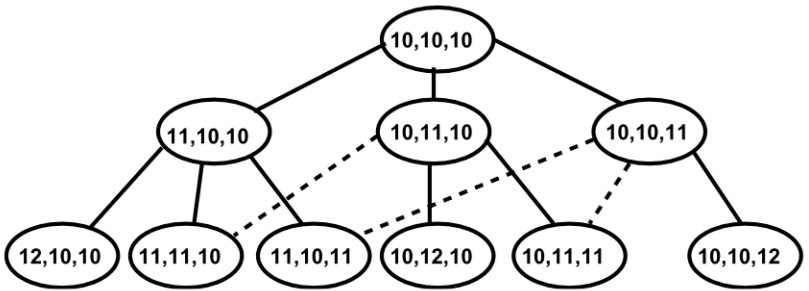
Source: Luna, Ryan J., and Kostas E. Bekris. "Push and swap: Fast cooperative path-finding with completeness guarantees." *Twenty-Second International Joint Conference on Artificial Intelligence*. 2011.

Search-based optimal solvers

- A*
  - drawbacks: exponential state space and branching factor in the number of agents
- A* with independence detection
  - reduces the effective number of agents
  - start with singleton groups
  - conflicting groups are merged
  - conflict avoidance tables - break ties in favor of states with the fewest conflicts
- M*
  - dynamically change the branching factor based on conflicts
  - expanded nodes generate only one child unless conflict occurs

Search-based optimal solvers II

- Operator decomposition
  - agents are assigned fixed order
  - A* node is expanded one agent at a time, creating intermediate nodes
  - number of regular surplus nodes is reduced
- Enhanced partial expansion (EPEA*)
  - when expanding a node, generates only children with current best f-value estimate
  - node is then re-inserted into the open list with f-cost of the next best child
- Increasing cost tree search (ICTS)
  - high level - increasing cost tree
  - low level - goal test



Source: Sharon, Guni, et al. "The increasing cost tree search for optimal multi-agent pathfinding." *Artificial intelligence* 195 (2013): 470-495.

CBS

- two-level algorithm
  - high-level search - Conflict Tree
  - low-level search - single-agent path finding algorithm
- CBS grows a set of constraints and finds paths consistent with these constraints
- conflicts are resolved by adding new constraints

Algorithm 2: High level of CBS (and MA-CBS).

Input: MAPF instance

1 $Root.constraints = \emptyset$
2 $Root.solution$ = find individual paths by the low level()
3 $Root.cost = SIC(Root.solution)$
4 insert $Root$ to OPEN
5 **while** OPEN *not empty* **do**
6 $P \leftarrow$ best node from OPEN // *lowest solution cost*
7 Validate the paths in P until a conflict occurs.
8 **if** P *has no conflict* **then**
9 **return** $P.solution$ // P *is goal*
10 C $\leftarrow$ first conflict (a_i, a_j, v, t) in P
19 **foreach** *agent* a_i *in* C **do**
20 $A \leftarrow$ new node
21 $A.constraints \leftarrow P.constraints + (a_i, v, t)$
22 $A.solution \leftarrow P.solution$
23 Update $A.solution$ by invoking low level(a_i)
24 $A.cost = SIC(A.solution)$
25 **if** $A.cost < \infty$ // *A solution was found* **then**
26 Insert A to OPEN

Source: Sharon, Guni, et al. "Conflict-based search for optimal multi-agent pathfinding."

CBS - low level

- input: agent and set of constraints
- two-dimensional state space - time and spatial
- any pathfinding algorithm can be used
  - only needs verification of constraints
  - if violated, state is discarded
- CAT tie-breaking policy possible for low-level A* states

Optimality of CBS

Definition 1: For a given node N in the constraint tree, let CV(N) be the set of all solutions that are: (1) consistent with the set of constraints of N and (2) are also valid (i.e., without conflicts).

Definition 2: We say that node N permits a solution p if it is an element of CV(N).

Lemma 1: The cost of a node N in the CT is a lower bound on minCost(CV(N)).

Lemma 2: For each valid solution p, there exists at least one node N in OPEN such that N permits p.

> Consequence: At all times at least one CT node in OPEN permits the optimal solution.

Theorem: CBS returns an optimal solution.

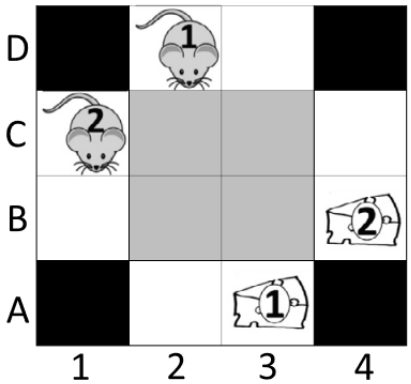
Completeness of CBS

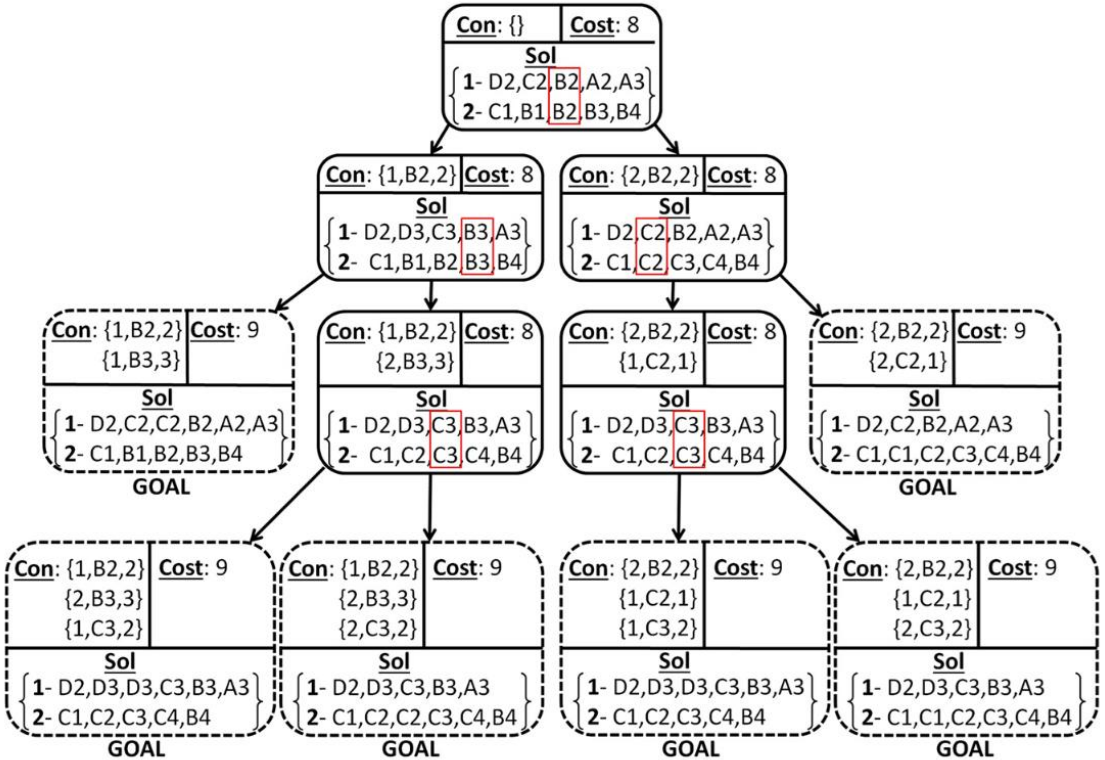
Theorem 2: For every cost C, there is a finite number of CT nodes with cost C.
Theorem 3: CBS will return a solution if one exists.

Claim: CBS will not identify an unsolvable problem.

A* vs CBS - bottlenecks

A* vs CBS - open spaces





Source: Sharon, Guni, et al. "Conflict-based search for optimal multi-agent pathfinding."

Empirical evaluation

- algorithms
  - A*
  - ICTS
  - EPEA*
  - CBS
- sum-of-costs function
- SIC heuristic on low-level
- maps
  - 8x8 4-connected grid
  - Dragon Age: Origin

Empirical evaluation - 8x8 grid

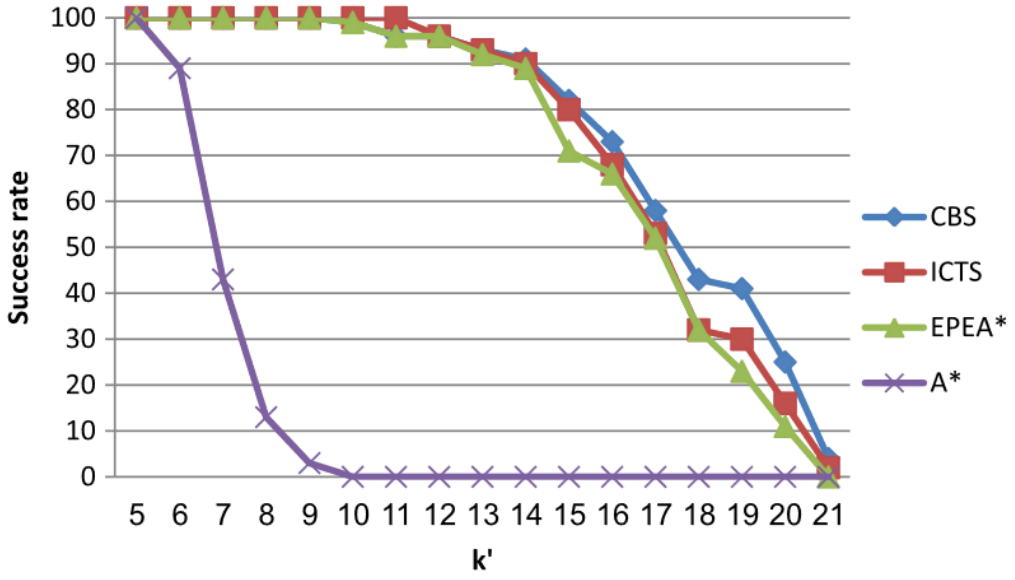


Fig. 8. Success rate vs. number of agents 8×8 grid.

Source: Sharon, Guni, et al. "Conflict-based search for optimal multi-agent pathfinding."

Empirical evaluation - 8x8 grid

Table 1
Nodes generated and running time on 8×8 grid.

k'	#Generated nodes					Run-time (ms)				
	Count	A*	EPEA*	CBS(hl)	CBS(ll)	A*	EPEA*	ICTS	CBS	p-val
3	100	640	15	10	490	8	0	1	7	0.01
4	100	3965	25	24	1048	207	1	1	14	0.02
5	100	21,851	35	51	2385	3950	3	1	32	0.01
6	89	92,321	39	45	1354	37,398	**4**	8	20	0.09
7	100	NA	88	117	3994	NA	**15**	20	60	0.00
8	100	NA	293	266	8644	NA	**75**	100	148	0.00
9	100	NA	1053	1362	45,585	NA	**444**	757	879	0.01
10	99	NA	2372	3225	111,571	NA	**1340**	3152	2429	0.02
11	94	NA	7923	8789	321,704	NA	8157	**7318**	7712	0.44
12	92	NA	13,178	12,980	451,770	NA	13,787	19,002	**12,363**	0.36
13	86	NA	14,989	15,803	552,939	NA	18,676	28,381	**16,481**	0.50
14	83	NA	13,872	21,068	736,278	NA	**15,407**	35,801	24,441	0.14
15	71	NA	22,967	24,871	826,725	NA	33,569	54,818	**30,509**	0.45
16	64	NA	26,805	24,602	822,771	NA	41,360	65,578	**34,230**	0.32
17	49	NA	25,615	17,775	562,575	NA	42,382	75,040	**25,653**	0.22

Source: Sharon, Guni, et al. "Conflict-based search for optimal multi-agent pathfinding."

Empirical evaluation - DA:O

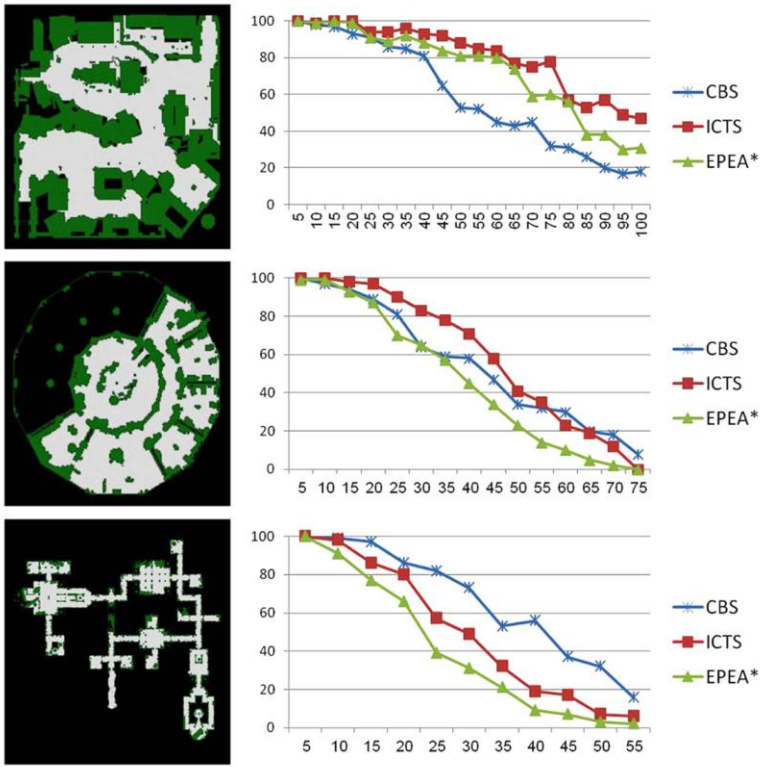


Fig. 9. The success rate of the different algorithms all running on top of ID for different DAO maps den520d (top), ost003d (middle), brc202d (bottom).

Source: Sharon, Guni, et al. "Conflict-based search for optimal multi-agent pathfinding."

Empirical evaluation - DA:O

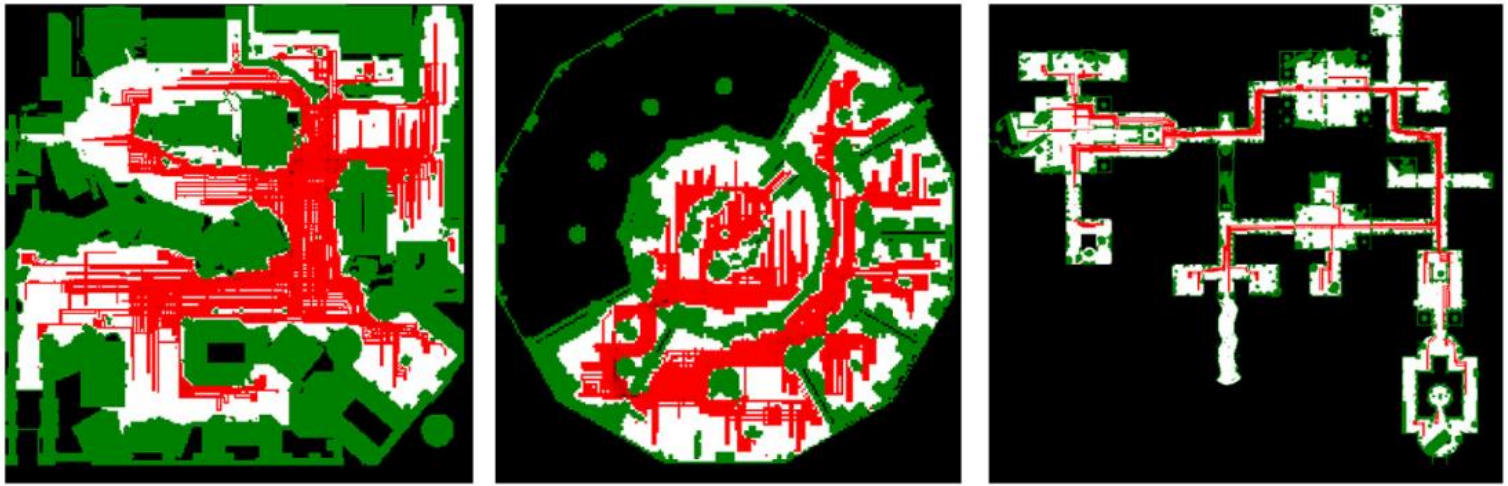


Fig. 10. DAO maps den520d (left), ost003d (middle), brc202d (right) and their conflicting locations.

Source: Sharon, Guni, et al. "Conflict-based search for optimal multi-agent pathfinding."

MA-CBS

- mitigates the worst case - strongly coupled agents
- can be adapted to suit the topology
- bounded number of conflicts between any pair of agents
- low-level solver has to be a MAPF solver

MA-CBS - Merging

- if there is a conflict, MA-CBS has two options:
  - branch
  - merge two conflicting (meta-)agents into a single meta-agent
- merge policy - conflict bound oriented merging
  - conflict bound parameter B
  - conflict matrix CM

MA-CBS - Conflicts

- there are three groups of conflicts before merging
  - internal - solved by merging
  - external(i)
  - external(j)
- merging external constraints
  - each constraint must apply only to the original agent

Experimental results

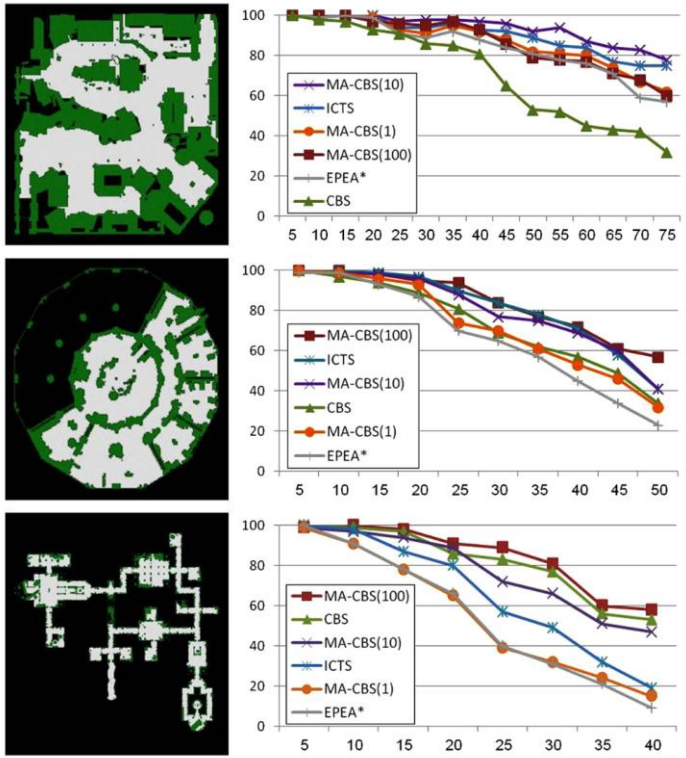


Fig. 11. Success rate of the MA-CBS on top of ID with EPEA* as the low-level solver.

Source: Sharon, Guni, et al. "Conflict-based search for optimal multi-agent pathfinding."

Conclusions from experiments

- in dense maps with many agents, low values of B are more efficient
- in maps with large open spaces and few bottlenecks, low values of B are more efficient
- if the MAPF solver is weak (plain A*), high values of B are preferred

“It is not yet fully understood how these different features are related to the performance of each algorithm, a point we intend to research in the future.”

Conclusions from experiments

- in dense maps with many agents, low values of B are more efficient
- in maps with large open spaces and few bottlenecks, low values of B are more efficient
- if the MAPF solver is weak (plain A*), high values of B are preferred

“It is not yet fully understood how these different features are related to the performance of each algorithm, a point we intend to research in the future.”