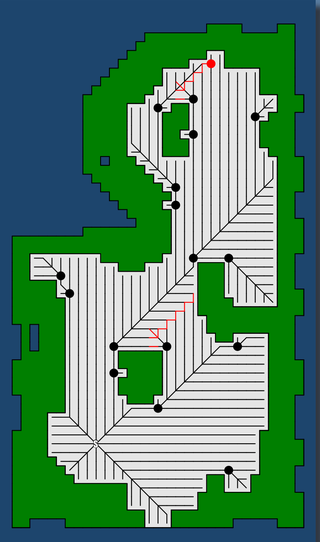
Canonical Orderings on Grids

Nathan R. Sturtevant, University of Denver
Steve Rabin, DigiPen Institute of Technology

Decomposing Jump Point Search

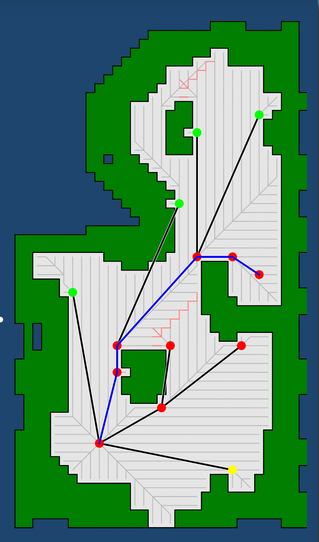
JPS uses a ***canonical ordering*** to break ties between paths of equal length. Diagonal moves are taken before cardinal moves. Special rules are used to wrap the canonical ordering around obstacles (from jump points) and guarantee that all states are reached.



JPS uses a ***jumping policy*** to avoid putting many states into the OPEN list.

JPS only puts the start, the goal, and jump points into the open list.

JPS continually expands states until one of these are generated.

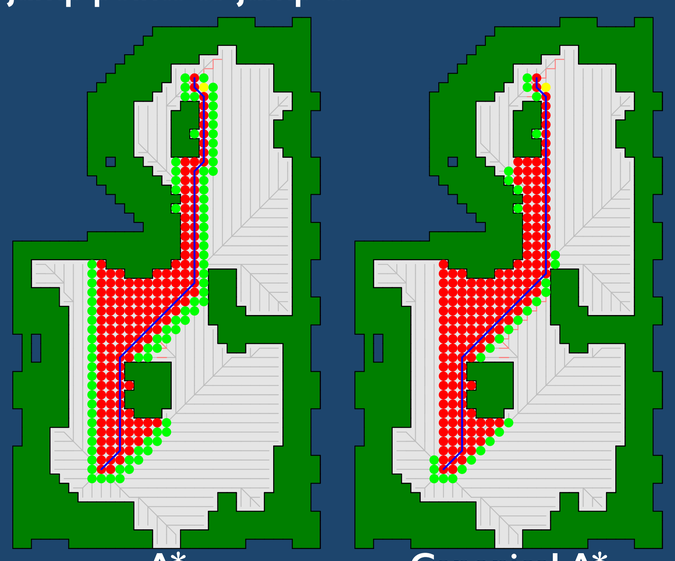


JPS performs ***best-first search*** over the states in the OPEN list. The search is identical to A*, just over a transformed search space from the original search graph.

Our novel decomposition of JPS allows us to create new algorithms that use these canonical ordering and other components in different ways or in different settings.

New Algorithms From JPS

Canonical A* just uses the canonical ordering from JPS. It expands far more states than JPS, because it doesn't jump. But, it generates far fewer states because it doesn't scan in the wrong direction for jump points to jump to.



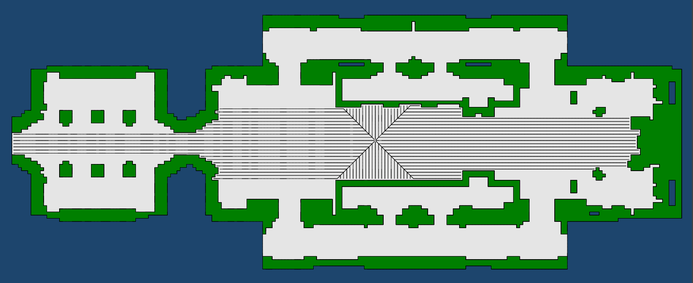
A* has mores states on OPEN (green). Canonical A* expands slightly more states because it is forced to follow the canonical ordering.

	A*	CA*	JPS
Time (ms)	5.6000	2.566	1.982
Expan.	13,295	13,302	229
Gener.	99,483	13,654	61,282

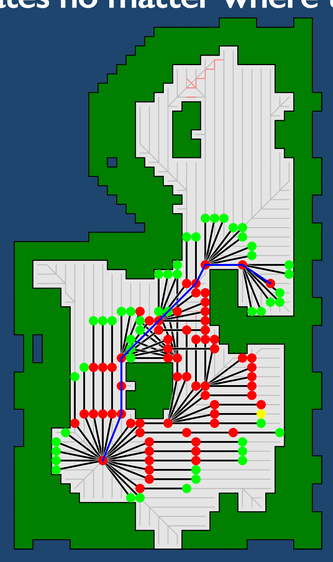
These results are from Dragon Age maps.

Canonical A* should be used when expansions are cheaper than node generations. (e.g. in a robotic domain where testing for legal successors is expensive)

Bounded JPS modifies the jumping rule from JPS. Instead of jumping to the next jump point, it bounds the length of a jump to some constant distance. This prevents JPS from jumping too far on large maps.



On this map, JPS will generate all the marked states no matter where the goal is.



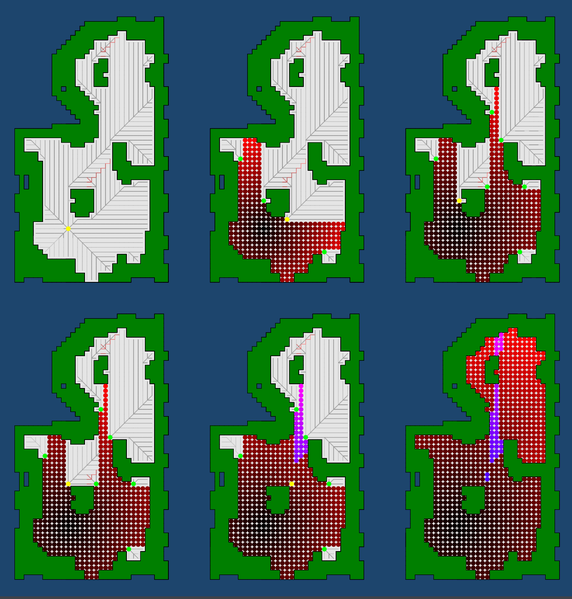
BJPS provides performance that interpolates between JPS and Canonical A*. Exact performance depends on the map set.

	CA*	BJPS(4)	JPS
Time (ms)	2.59	1.50	1.98
Expan.	13,301	4,091	229
Gener.	13,654	22,363	61,281

These results are from Dragon Age maps.

Canonical Dijkstra takes advantage of the fact that in a single-source shortest-path search every state in the state space will be visited. Therefore, the jumping policy does not hurt performance.

Canonical Dijkstra writes g-costs into the closed list while jumping, but must update g-costs if lower costs are found later (purple). It is 2.5x to 4.4x faster than Dijkstra, depending on the maps.



Weighted JPS uses a different best-first search strategy, replacing A* with weighted A*. Weighted JPS trades bounded suboptimality for faster performance.

	Time		Nodes Exp.	
Weight	CA*	JPS	CA*	JPS
1	2.566	1.982	13,302	229
2	1.471	1.210	8,002	146
5	1.052	0.921	6,003	110
10	0.966	0.862	5,574	103

	Path Quality		Nodes Gen.	
Weight	CA*	JPS	CA*	JPS
1	1.00	1.00	13,654	61,282
2	1.01	1.02	8,300	37,575
5	1.03	1.05	6,284	28,888
10	1.04	1.07	5,855	27,714