Jump Point Search with Temporal Obstacles

Shuli Hu, Daniel D. Harabor, Graeme Gange, Peter J. Stuckey, Nathan R. Sturtevant

NORTHEAST NORMAL UNIVERSITY · MONASH University · UNIVERSITY OF ALBERTA

Motivation

In 4-connected grid-based path planning one often needs to account for temporal and moving obstacles. However, finding a time-optimal path in such case is more challenging than 2D pathfinding. The main issues are as follows.

① The temporal aspect increases the size of the search space.
② The search space contains many symmetric paths. There are spatial symmetries (Figure 2) and temporal symmetries (Figure 1).
We develop jump point search with temporal obstacles (JPST) to break these symmetries.

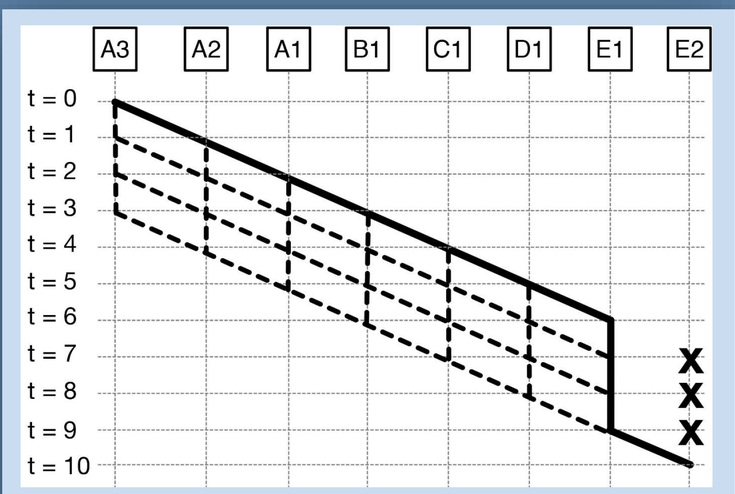

Figure 1: Temporal symmetries.

Main Idea

We introduce a new canonical ordering called Vertical-then-Horizontal-then-Wait (VHW). The VHW-paths has:

- Vertical first
- Wait last
- Backtrack free

In Figure 1 and 2, there are many equivalent paths. However, there is only one path (the black line) follows the new VHW-ordering.

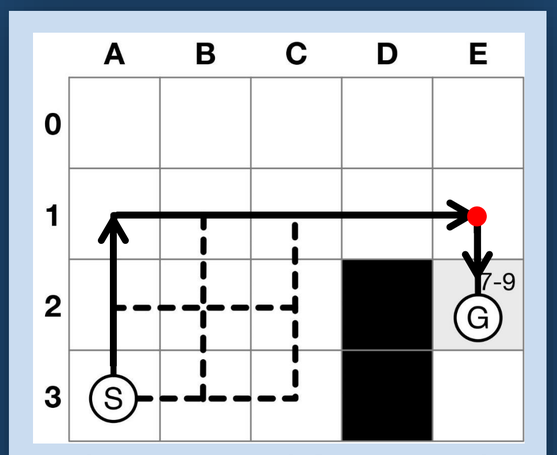

Figure 2: Spatial Symmetries.

Pruning and Jumping Rules

- Spatial pruning: The vertical direction is with horizontal direction. The horizontal direction restarts vertical direction only when there is a forced neighbor (Figure 3).
- Temporal pruning: B2 is a temporal obstacle (Figure 4). JPST takes wait action and prunes other neighbors. JPST is forced to generate B2 as the obstacle disappears.
- Jumping rules apply pruning rules recursively until the next move is invalid or there is a forced neighbor.
- Jumping procedure processes the canonical continuations in following order to guarantee the paths are VHW: first wait actions, then horizontal, finally vertical actions.

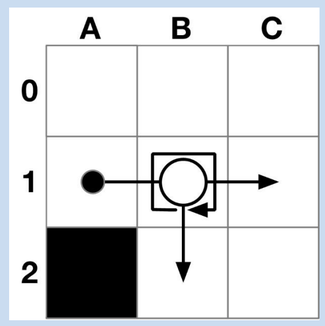

Figure 3: Spatial pruning rules

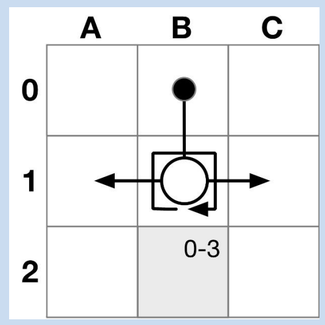

Figure 4: Temporal pruning rules

Practical Considerations

- Jump Limits: Helps to avoid recursively scanning large and uninteresting areas.
- Fast Scanning: Use bitfields to represent the positions of permanent and temporal obstacles, and scan map fast.

- Speculative Waiting and Safe Intervals: These are closely related to SIPP, but we take wait actions as late as possible. We also use an interval-based representation of the temporal dimension which further improves search performance.

Experimental Results

Experiment #1: Moving Obstacles

Use k trajectories of agents as the moving obstacles. In figure 5, the curves represent the time ratio of SIPP (Save Intervals Path Planning) over JPST. The higher ratio, the more efficient JPST. Ratio larger than one means JPST is faster than SIPP.

Observations:
① JPST improves on SIPP in most cases, when k=0 it gets maximum speedups.
② As k increases, the gap narrows: with temporal obstacles increasing, JPST and SIPP will converge to time-expanded A*.

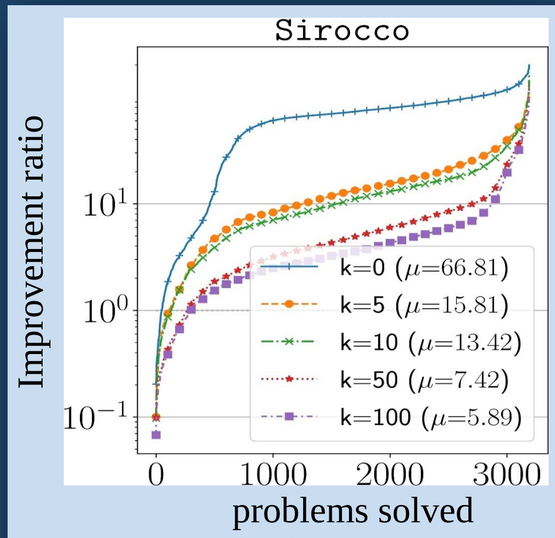

Figure 5: Plots of algorithms with Manhattan heuristic.

Experiment #2: Pathfinding with Constraints

Generate constraints by solving multi-agent pathfinding using conflict-based search, and solve pathfinding with constraints by JPST and SIPP.

From results, we can observe:
① In average runtime, JPST gets several factors speedup vs. SIPP, and several orders vs. Time Expanded A* (Figure 6).

② Main performance improving of JPST with perfect heuristic comes from strong symmetry breaking rather than strong heuristic (results can be seen in paper).

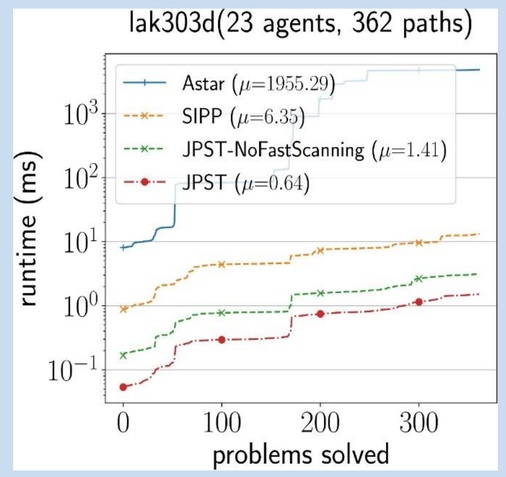

Figure 6: Runtime plots with Manhattan distance.