

1 Implementation

The backbone of the Sokoban solver is the A* algorithm, which maintains a priority queue of up to *beam_width* states per iteration. It expands states in parallel, prune duplicates, and select a subset of states for the next iteration.

A *State* struct stores the player's position and the positions of boxes. Bitsets are employed to enable compact storage and efficient operations (e.g., XOR).

For the heuristic, it computes the minimum sum of Manhattan distance between boxes and targets using the Hungarian algorithm, and the results are cached per thread to avoid recomputation overhead.

Parallelism is implemented using Intel TBB. The *parallel_for* constructs expands nodes across threads, with *concurrent_vector* used for candidate collection and per-thread max-heaps to cap the number of candidates. Pruning and top-K selection are then performed sequentially to ensure correctness.

Deadlock detection is relatively simple. Checks for corner, blocked, and boxes stuck in aisles. These checks are applied using precomputed dead zones and dynamic conditions. BFS is used to compute player reachability and valid move paths.

The main program first parsing the map, initializing corresponding bitsets for walls, targets, fragile floors, boxes, and the player. Grid indices will be mapped to row-column pairs. It will precompute dead zones via flood-fill and static checks. The main beam A* loop will pop all states from priority queue, expand them in parallel using TBB to generate new states by pushing boxes. Then checks validity and deadlocks, computes heuristics with caching locally. Candidates are collected in a *concurrent_vector*, pruned sequentially to keep the best scores, and the top-K are selected and sorted for the next iteration. Each iteration checks for the goal. If all boxes are found on targets, reconstructing the moving sequence via backtracking. If not, tries with adaptively larger beam width.

2 Difficulties encountered

During this assignment, I realized that Sokoban is incredibly difficult to solve efficiently. The main challenges I encountered were not regarding parallel programming, but designing and implementing algorithms. Integrating BFS, A*, beam search while exploring state space was demanding. Carefully pruning deadlock pattern branches was also crucial. In addition, finding efficient state representations and selecting corresponding data structures were also troublesome tasks.

On the parallel programming side, I have hardly any experience with Pthread, OpenMP, or Intel TBB, so employ them to parallel the algorithm also consumed a significant amount of time. I frequently encountered synchronization issues and segmentation faults, and resolving them required substantial time and efforts.

3 Strengths and weaknesses of pthread and OpenMP

Pthread strengths:

Direct and fine-grained control over thread management, excellent for irregular parallelism. Standard low-level API, portable across POSIX systems.

Pthread weaknesses:

High implementation complexity, prone to synchronization pitfalls like race conditions, deadlocks, and livelocks.

OpenMP strengths:

Ease of use, one can rapidly parallelize existing loops or sections of code by adding a few OpenMP directives. The OpenMP directives are mostly non-intrusive, the parallel logic is separated from the core sequential algorithm. The runtime library and compiler automatically handle thread creation, synchronization, joining, and work distribution.

OpenMP weaknesses:

Less control over how threads are created and scheduled. It is best suited for parallelizing structured loops, but can be very less effective for complex, irregular task parallelism.

4 Any suggestions or feedback for the homework

I did learn a lot from this assignment, especially about exploring deadlock detection and state pruning. However, I feel the focus leaned more toward utilizing sophisticated algorithms and data structures rather than parallel programming itself. The resulting workload requires significantly more time than expected, and I ended up spending almost an entire week on this assignment. After all, still appreciate the efforts of the TAs and instructor in preparing this task and guiding us through it, and I sincerely hope the performance grading will not be based on directly comparing the ratio of the running time with the leader.