15 puzzle problem

We used Iterative deepening A\*(IDA\*) to find the minimum number of steps to reach the complete puzzle (there is a board 4x4 with numbers from 1 to 15 and an empty tile, the solution is reached when all the tiles are in numerical order, from left to right, top to bottom).

IDA\* uses a more informative search depth as cutoff for each iteration, f(n) = g(n) + h(n), where g(n) represents the cost of the path from the root to the node n, h(n) represents the problem-specific heuristic estimate of the cost of the path from node n to the goal.

We choose h(n) to be the Manhattan Distance of the current board state, g(n) represents the number of steps made so far (the number of times the empty tile was moved).

Implementation

We used ExecutorService with a thread pool of fixed size to parallelize the search. Futures are submitted recursively to the service and at each submission the current number of steps is increased by 1. Each submitted task receives nrThreads/nrUnexploredNextMoves threads, when nrThreads reaches 0 we execute sequential search. When the futures finish, the smallest length of the path is returned and if we reached a solution the algorithm finishes, otherwise a new minimum bound is set and the parallel search begins again.

The distributed algorithm uses the MPJ Express Java library which implements the MPI Interface. The number of nodes is passed as a command line argument, with one master node and several worker nodes.

Work is assigned to each worker by performing one or more generateMoves() from the root node recursively, until the total workload gets as close as possible to the number of nodes.

When each worker receives its state, it begins searching from that point on using the iterative A\* sequential implementation and sends each solution along with its number of steps to the master function. The master function checks whether a solution has been found, and if it has, all the workers are signaled to stop searching. If not, the min bound is reset and the search continues.