Jan. 20, 2017

Solving the 15-tile puzzle using MM search

Submitted by: Yuval Lieberman
To: Prof. A. Felner
In the course: Search Methods in Artificial Intelligence 2017 - 1
Department of Software and Information Systems Engineering
Ben-Gurion University of the Negev

MM search is a novel bidirectional search algorithm that always "meets in the middle" (Holte, Felner, Sharon, & Sturtevant, 2016). This paper will examine how the MM bi-directional search can be applied to the 15-tile puzzle problem. The first section will provide a brief background on MM search and the 15-tile puzzle. The second section will describe the experimental design and some implementation details. The final section will present and discuss the experiment results.
The source code written for this experiment was made public.

Background

The motivation behind MM search was to create a bidirectional search algorithm that is guaranteed to meet at the middle. This guarantee is based on never expanding a node with $g(n)>C^*/2$ – a node with a distance from the start or goal, larger than half the optimal solution. This is based on a novel priority function for retrieving nodes from the open list: $pr_F(n) = \max(f_F(n), 2g_F(n))$, always expanding the node with minimum priority among the forward and backward open lists. The optimality of the solution relies on the stop condition of a cost that is lower than several lower bounds (Holte et al., 2016).
MMε is a variation of MM that may improve the number of expanded nodes under certain circumstances, such as a weaker heuristic. It is similar to MM, except for the priority function, which is now $pr\varepsilon_F(n) = \max(f_F(n), 2g_F(n)+ \varepsilon F(n))$, adding the minimum operation cost to the 2g(n) expression (Sharon, Holte, Felner, & Sturtevant, 2016).
This paper will examine these algorithms on the 15-tile puzzle. This puzzle is one of the classic problems in AI search. It is an exponential problem with a low branching factor, and a problem space of 10^13 states. The 15-tile puzzle was solved by (Korf, 1985) using IDA* with Manhattan distance. In this experiment, I will two heuristics: Manhattan distance, which is a simple and basic heuristic, and an improved version of Manhattan distance accounting for linear conflicts, which is more accurate. Both heuristics are admissible and consistent.

Experimental design

The experiment was done on 6 instances of the 15-tile puzzle, in an increasing difficulty level, from a puzzle instance that requires 13 steps, to an instance that requires 65 steps.
For each layout instance, a benchmark was set using an IDA* search. This benchmark was used to define the optimal solution cost.
After stipulating the optimal path cost, three bidirectional search algorithms were tested on each puzzle instance: bidirectional A*, MM and MMε.
To examine how heuristic accuracy affects the algorithm efficiency - each algorithm was deployed twice: once using Manhattan distance with linear conflict, and another time using Manhattan distance only, which is a weaker heuristic.
The total number of runs was: 6 benchmarks + 6 instances x 3 algorithms x 2 heuristics.
Additional exploratory runs were done using bidirectional weighted A* with a range of weight values from 1.2 to 4.

The software used for running the experiments was written in Java. The source code was made publically available at: https://github.com/yuvallb/15-puzzle-solver.
Original source code for bidirectional A* was taken from (Senecal, 2014).
The benchmark IDA* was done using Java code based on (Borowski, 2013).

Results

The first comparison, presented in table 1, is between bidirectional A* to MM and MMε, all using the same heuristic - Manhattan distance with linear conflicts.
The first column is the optimal cost, calculated as the solution of the puzzle instance using IDA*.
In the next columns, the upper cell contains the solution path that was found, in the form of: forward + backward = total. The lower cell in these columns show the number of expanded nodes, which is the total number of nodes in the open and close lists when returning the solution.
It can be seen that the MM algorithms always finds an optimal path, and – as expected - the optimal path always meets in the middle of the forward and backward search. In some cases, MM finds more than one optimal path.
A* bidirectional search does not find the optimal solution, except in the easiest problem instance. In terms of execution time and space complexity, MM search performs worse than A* bidirectional search, in the worst case by a factor of 10. It seems that the relative performance is getting worse as the problem size increases. When testing the search on a puzzle instance with an optimal path of 65, I was not able to get any results for MM due to its high run time.

Optimal cost	**Metric**	**Bidirectional A***	**MM**	**MMε**
13	Forward + Backward:	8 + 5 = 13	7 + 6 = 13	7 + 6 = 13
	Nodes Explored:	48	50	50
30	Forward + Backward:	14+18=32	15+15=30	15+15=30
	Nodes Explored:	1,867	4,808	4,808
35	Forward + Backward:	15+22=37	18+17=35 (2 paths)	18+17=35
	Nodes Explored:	2,675	9,595	19,192
43	Forward + Backward:	20+25=45	21+22=43	22+21=43
	Nodes Explored:	25,183	100,141	104,403
50	Forward + Backward:	18+34=52	25+25=50 (2 paths)	25+25=50 (2 paths)
	Nodes Explored:	44,910	482,778	482,778

Table 1: Comparing performace of bidirectional search algorithms on the 15 puzzle, using Manhattan distance + linear conflict heuristic

The comparison of MM to bidirectional A* was done in order to compare two bidirectional searches. However, this comparison lacks since bidirectional A* returns suboptimal solutions. Table 2 compares the number of nodes expanded in MM to unidirectional A* and IDA* with the same heuristic. The best result is highlighted in bold. Also displayed in the table is the MM_0 upper bound: $2*b^{C*/2}$ (b=3, the static branching factor).
As expected, IDA* always expands more nodes than A*, due to its iterative nature of reopening nodes. MM outperforms A* in 3 of out 5 instances.

C*	A*	IDA*	MM	MMε	$2x3^{C*/2}$
13	**37**	94	50	50	2,525
30	5,179	7,962	**4,808**	**4,808**	2.8e7
35	11,735	31,570	**9,595**	19,192	4.4e8
43	**45,029**	87,313	100,141	104,403	3.6e10
50	1,188,329	3,565,946	**482,778**	**482,778**	1.6e12

Table 2: Comparing nodes explored by A, IDA*, MM and MMε*

The second step in the experiment was to examine the effect of the heuristic strength. To do that, the experiment was repeated with the same heuristic: Manhattan distance, but without accounting for linear conflict.
Table 3 displays the effect of improving the heuristic function on the time and space complexity of the solution, as measured by the number of nodes explored. As the problem size increases, the effect of a stronger heuristic is more meaningful. Also, shown in the table is that the solution optimallity remains the same – MM will find the optimal cost path, even when the heuristic is weaker. MMε displayed similar results, so they are not presented in detail here. Full results are available online at: https://github.com/yuvallb/15-puzzle-solver/blob/master/output.txt

Optimal cost	Metric	MM using Manhattan distance only	MM using Manhattan distance and linear conflict	Effect of improving the heuristic (reduction in nodes explored)
13	Forward + Backward:	7 + 6 = 13	7 + 6 = 13	
	Nodes Explored:	75	50	67%
30	Forward + Backward:	15+15=30	15+15=30	
	Nodes Explored:	9,450	4,808	51%
35	Forward + Backward:	18+17=35 (2 paths)	18+17=35 (2 paths)	
	Nodes Explored:	18,604	9,595	52%
43	Forward + Backward:	21+22=43	21+22=43	
	Nodes Explored:	369,259	100,141	27%
50	Forward + Backward:	25+25=50 (3 paths)	25+25=50 (2 paths)	
	Nodes Explored:	2,398,242	482,778	20%

Table 3: Comparing performance of MM search on the 15 puzzle, with weak and strong heuristics

The third step was to examine if and how MMε reduces the number of expanded nodes. Table 1 shows that the number of explored nodes is the same or larger in MMε. However, this number includes nodes that were generate but not expanded. Table 4 compares the number of expanded nodes between MM and MMε. It would be expected that for weaker heuristics MMε will expand less nodes than MM. This was not the case in this specific experimental setup, where MM always expanded less or equal number of nodes than MMε.

	Manhattan distance only		**Manhattan distance and linear conflict**	
	MM	**MMε**	**MM**	**MMε**
13	33	33	21	21
30	4,701	4,701	2,409	2,409
35	9,317	18,005	4,809	9,681
43	195,413	208,421	51,871	54,206
50	1,278,238	1,278,238	260,544	260,544

Table 4: Comparing number of expanded nodes in MM and MMε under different heuristics and problem sizes

Conclusion

This paper presented an experiment on implementing the MM bidirectional search algorithm on the 15-tile puzzle.
When compared to A* bidirectional search, MM reached optimal solutions, but at higher run times. When compared to unidirectional A* and IDA*, MM was mostly superior.
The effect of the heuristic accuracy was shown as crucial for improving the run time of the solution.
MMε was not shown to reduce the number of expanded nodes.
The source code for this experiment was made public and can be used to reproduce and improve these results.

References

Borowski, B. (2013). Optimal 8/15-Puzzle Solver. Retrieved from http://brian-borowski.com/software/puzzle/

Holte, R., Felner, A., Sharon, G., & Sturtevant, N. (2016). Bidirectional Search That Is Guaranteed to Meet in the Middle. Retrieved from http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12320

Senecal, J. (2014). 15 Puzzle Solver. Retrieved from https://github.com/jeffsenecal9/15-puzzle-solver

Sharon, G., Holte, R. C., Felner, A., & Sturtevant, N. R. (2016). Extended Abstract: An Improved Priority Function for Bidirectional Heuristic Search. *Ninth Annual Symposium on Combinatorial Search*.

Source Code

Source code and full results are available at: https://github.com/yuvallb/15-puzzle-solver