Memory-Bounded Bidirectional Search

Eshed Shaham
School of Engineering and CS
Hebrew University of Jerusalem
Jerusalem, Israel
eshed.shaham@mail.huji.ac.il

Ariel Felner
ISE Department
Ben-Gurion University
Be'er-Sheva, Israel
felner@bgu.ac.il

Jeffrey S. Rosenschein
School of Engineering and CS
Hebrew University of Jerusalem
Jerusalem, Israel
jeff@cs.ac.il

1 Introduction and Overview

Bidirectional heuristic search (Bi-HS) is a topic that has seen great progress recently. New theoretical results have been established regarding states that must be expanded (Eckerle et al. 2017), and several novel Bi-HS algorithms have been introduced, such as MM (Holte et al. 2016), NBS (Chen et al. 2017), and Fractional MM (Shaham et al. 2017). Nevertheless, their main limitation is that, like all previous Bi-HS algorithms, they are constrained to store at least one of the frontiers in memory.

It is well known that unidirectional heuristic search (Uni-HS) can be replaced with memory-bounded variants such as IDA* (Korf 1985), SMA* (Russell 1992), and RBFS (Korf 1993). It is, however, far more difficult to do so with Bi-HS. This is the topic of our ongoing work.

We use f_F, g_F and h_F to indicate costs of the forward search, and f_B, g_B and h_B to indicate costs of the backward search. Likewise, $Open_F$ and $Open_B$ store states generated in the forward and backward directions, respectively.

2 Iterative Deepening Bidirectional Search

IDA*, a memory-bounded variant of A*, reduces memory requirements by using iterative deepening. Every iteration tries to find solutions of a fixed-cost threshold th. If no solution is found, th is increased.

How can we migrate this principle into Bi-HS? In Uni-HS, a solution of cost th is found when a goal state is generated with $g(goal) = th$. In Bi-HS, a solution of cost th is found when a state n is generated by both the forward and backward searches, and $g_F(n) + g_B(n) = th$. We call n the *meeting point*. Note that the Uni-HS case is identical to the Bi-HS case when the meeting point is the goal. Assuming unit costs, there are $th + 1$ possible meeting points on every solution, but only one of them has to be found. Thus, we specify th_F and th_B such that $th_F + th_B = th$, and then search for states n such that $g_F(n) = th_F$ and $g_B(n) = th_B$. If no such state is found, either th_F or th_B is increased. We thus introduce Iterative Deepening Bidirectional Search (IDBS), a general framework that acts in this way. It is flexible as to which threshold (th_F or th_B) to increase next; IDA* is a special case, where increasing th_F is mandatory.

3 Using Type Systems

IDBS is a framework and not an algorithm. It does not specify how the search phase of each iteration is implemented, and provides no guarantees on its memory requirements. How can we bound the memory used by the search phase of the different iterations?

Consider the following search phase: two separate searches, forward and backward, generate a list of potential meeting points, $Open_F$ and $Open_B$ respectively. A solution is found if $Open_F \cap Open_B$ is nonempty. The trivial implementation requires storing $Open_F$ in memory and searching for a match when generating $Open_B$.

Type systems (Lelis et al. 2012) partition the entire state-space into k disjoint types, labeled $T_1, T_2, \ldots, T_k$. Now, we incorporate a type system into IDBS by performing k different searches in each iteration. For each type T_i we perform the search with the same threshold th_F, but only store $Open_F \cap T_i$. We then perform the backward search for type T_i, and match against the stored $Open_F$. This is repeated until a solution is found or all types have been searched, in which case the next iteration is activated. Naturally, if the available memory can contain M states, we should partition the graph uniformly into $k = |Open_F|/M$ types. This comes with the tradeoff that the full search is executed $|Open_F|/M$ times for each iteration.

4 The Reference Search Improvement

A very effective pruning rule can be applied to the search of each type by using a type system that partitions the states based on their h-value towards an arbitrary *reference* state r. That is, we define $T_i = \{n \mid h(n, r) = i\}$. Assuming a consistent heuristic, for every two states u, v it must hold that $|h(u, r) - h(v, r)| \leq d(u, v)$. Let n be a state generated during the forward search with th_F. We know that for every state $u \in Open_F \cap T_i$ it holds that $g_F(u) = th_F$ and $h(u, r) = i$. Therefore, if $|h(n, r) - i| > th_F - g_F(n)$, n can be safely pruned from the search.

In addition, we are exploring ways to use Bloom Filters (Bloom 1970) to store the frontiers. Experiments applying IDBS with reference search and Bloom Filters, managed to optimally solve the 86-pancake puzzle in a few hours with a memory bound of approx. 8×10^5 states, generating more than 4×10^{10}. This is the largest pancake puzzle ever solved. We intend to continue this line of research in the future.

References

Bloom, B. H. 1970. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM* 13(7):422–426.

Chen, J.; Holte, R. C.; Zilles, S.; and Sturtevant, N. R. 2017. Front-to-end bidirectional heuristic search with near-optimal node expansions. In *IJCAI*.

Eckerle, J.; Chen, J.; Sturtevant, N. R.; Zilles, S.; and Holte, R. C. 2017. Sufficient conditions for node expansion in bidirectional heuristic search. In *ICAPS*.

Holte, R. C.; Felner, A.; Sharon, G.; and Sturtevant, N. R. 2016. Bidirectional search that is guaranteed to meet in the middle. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Korf, R. E. 1985. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial intelligence* 27(1):97–109.

Korf, R. E. 1993. Linear-space best-first search. *Artificial Intelligence* 62(1):41–78.

Lelis, L.; Stern, R.; Felner, A.; Zilles, S.; and Holte, R. C. 2012. Predicting optimal solution cost with bidirectional stratified sampling. In *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling, ICAPS 2012, Atibaia, São Paulo, Brazil, June 25-19, 2012*.

Russell, S. J. 1992. Efficient memory-bounded search methods. In *ECAI*, volume 92, 1–5.

Shaham, E.; Felner, A.; Chen, J.; and Sturtevant, N. R. 2017. The minimal set of states that must be expanded in a front-to-end bidirectional search. In *SoCS*.