Using Infeasibility to Improve Abstraction-Based Heuristics

Fan Yang, Joseph Culberson, and Robert Holte

Computing Science Department, University of Alberta
Edmonton, Alberta T6G 2E8 Canada
{fyang,joe,holte}@cs.ualberta.ca

The contribution of our research is to show that the accuracy of the heuristics generated by abstraction can be improved by checking for infeasibility. What do we mean by infeasible heuristics? For a state t, the heuristic value h is infeasible if it is proved that the cost of a solution for t cannot be h. Take the sliding puzzle for example, assuming that the manhattan heuristic for state t is md(t), if md(t) is even, any odd number is infeasible. To substantiate our approach, we begin with formal definitions and lemmas. Then empirical results show the effectiveness of the approach. For more details please refer to our longer work[5].

A *state space* is a weighted directed graph with a set of states, a set of directed edges (ordered pairs of states) and the edge cost function. For example, a set of states may be defined by the set of all possible assignments to a set of state variables and the edges, and the edge cost function will depend on the operations on the variable sets. An *abstraction system* includes a state space, a set of abstract state spaces and a set of mappings $\mathbf{\Psi} = \{\psi_1, \ldots, \psi_k\}$ from the initial state space to abstract spaces. Our definition is similar to the work of Prieditis[4]. The key difference is that here we split each edge cost into two costs: the *primary cost* C_i and a *residual cost* R_i. Given a path p from t to g in the initial state space, for each abstract space $\mathcal{A}_i$, $\boldsymbol{p}_i$ is the corresponding abstract path from t_i to g_i, where $t_i = \psi_i(t)$ and $g_i = \psi_i(g)$. To guarantee admissibility, we require that for any path p from t to g, $C(p) \geq C_i(\boldsymbol{p}_i) + R_i(\boldsymbol{p}_i)$. We say that abstractions are additive if the cost of each edge in the original space is larger than or equal to the sum of C_i of corresponding edges in all abstract state spaces. This definition generalizes those in [1–4]. $C_i^*(t_i, g_i)$ is the minimum primary cost of an abstract path from t_i to g_i. Define $R_i^*(t_i, g_i)$ to be the minimum residual cost among the paths whose primary cost is minimal. Given a goal state g, the heuristic of state t defined by k additive abstractions is $h(t) = \sum_{i=1}^{k} C_i^*(t_i, g_i)$. Lemma 1 gives a test for infeasibility of additive abstraction-based heuristics.

Lemma 1. *Given k additive abstractions, if for some $j, 1 \leq j \leq k$, we have $h(t) < C_j^*(t_j, g_j) + R_j^*(t_j, g_j)$, then $h(t)$ is infeasible.*

Figure 1 is an example test for infeasibility. Table 1 indicates that additive heuristics may be improved by checking for infeasibility. The performance of IDA* using additive heuristics with/without checking for infeasibility can be compared in the first two rows and the last two rows. The average running time of IDA* using the heuristics enhanced by checking for infeasible additive values is over 2 times faster than the running time required on average without checking for infeasibility on the same machine.

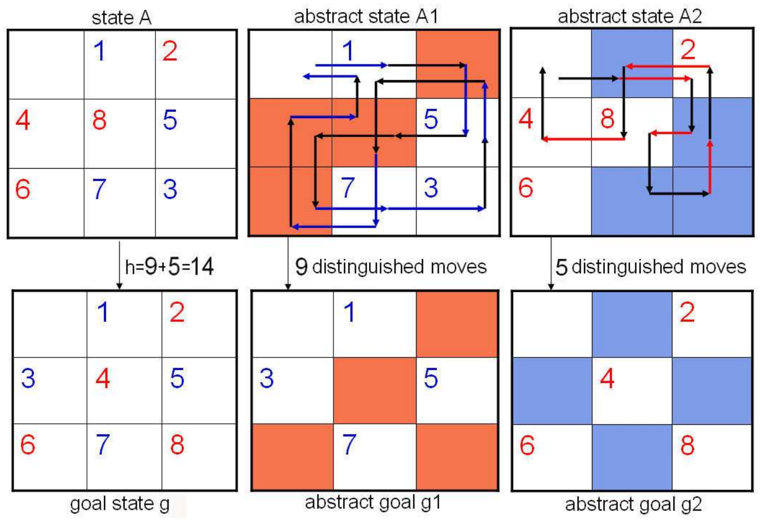

Fig. 1. The primary cost C^* is defined by the total moves of numbered tiles in the abstract state (i.e. distinguished moves) and the residual cost R^* is the number of moves of other tiles. $(C_1^*, R_1^*) = (9, 9)$, $(C_2^*, R_2^*) = (5, 7)$. $h = \sum_{i=1}^{2} C_i^* < (C_1^* + R_1^*)$. So h=14 is an infeasible heuristic value. h can be improved to be 16.

Tile Partition	Check Infeasibility	Average H	Average Nodes	Average Sec
	Yes	42.10	1,453,358	0.312
5-5-5	No	41.56	3,186,654	0.642
	Yes	42.78	784,145	0.171
6-6-3	No	42.13	1,858,899	0.379

Table 1. 15 sliding tile puzzle results

References

1. Edelkamp, S.: Planning with pattern databases. In: Proceedings of the 6th European Conference on Planning. (2001) 13–34
2. Felner, A., Korf, E., Hanan S.: Additive pattern database heuristics. Journal of Artificial Intelligence Research. **22** (2004) 279–318
3. Korf, E., Felner, A.: Disjoint pattern database heuristics. Artificial Intelligence. **134** (2002) 9–22
4. Prieditis, A.E.: Machine discovery of effective admissible heuristics. Machine Learning **12** (1993) 117–141
5. Yang, F., Culberson, J., Holte, R.: A general additive search abstraction. Technical Report TR07-06. Department of Computing Science, University of Alberta (2007)