Escaping Heuristic Depressions in Real-Time Heuristic Search

(Extended Abstract)

Carlos Hernández
Departamento de Ingeniería Informática
Universidad Católica de la Ssma. Concepción
Concepción, Chile

Jorge A. Baier
Depto. de Ciencia de la Computación
Pontificia Universidad Católica de Chile
Santiago, Chile

ABSTRACT

Heuristic depressions are local minima of heuristic functions. While visiting one them, real-time (RT) search algorithms like LRTA* will update the heuristic value for most of their states several times before escaping, resulting in costly solutions. Existing RT search algorithm tackle this problem by doing more search and/or lookahead but do not guide search towards leaving depressions. We present eLSS-LRTA*, a new RT search algorithm based on LSS-LRTA* that actively guides search towards exiting regions with heuristic depressions. We show that our algorithm produces better-quality solutions than LSS-LRTA* for equal values of lookahead in standard RT benchmarks.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Graph and tree search strategies, Heuristic methods*

General Terms

Algorithms, Experimentation

Keywords

Agent Reasoning::Planning (single and multi-agent), Robot Reasoning::Planning, Path Planning

1. INTRODUCTION

In many real-world applications, agents need to act quickly in dynamic, initially unknown domains. Example applications range from robot navigation, to agent navigation in games (e.g., Baldur's Gate, Starcraft, etc.). Real-time (RT) search (e.g., [8]) is a standard paradigm for solving search problems in those settings. RT algorithms run a computationally cheap observe-plan-act cycle, in which the environment is observed, an action is selected, and then executed. Their search is guided by a heuristic function h, like in standard A* search [3].

Early heuristic RT algorithms like LRTA* and RTA* [8] perform poorly in presence of *heuristic depressions* [5] – bounded regions of the search space in which the heuristic is unrealistic with respect to the heuristic value of the states in the border of the region. Before exiting a depression, they will visit most states in the depression, possibly many times. State-of-the-art RT search algorithms (e.g. [7, 2, 6, 1, 4]) escape depressions more quickly as a consequence of performing more lookahead or more learning. More lookahead involves selecting an action by looking farther away in the search space, whereas more learning involves updating the heuristic values of several states in a single iteration. There are many algorithms that use one or a combination of these techniques.

In this paper, we propose eLSS-LRTA*, an RT search algorithm that *actively* guides search towards escaping heuristically depressed regions. eLSS-LRTA* is based on LSS-LRTA*, a state-of-the-art RT search algorithm. eLSS-LRTA* defers from its ancestor in two main aspects: (1) it provides a mechanism for detecting states that belong in a heuristic depression, and (2) it prefers to expand nodes that are *not* in a heuristic depression during its lookahead search.

We perform an experimental evaluation that shows generally improved performance in standard benchmark domains. In most cases, for an equal amount of lookahead eLSS-LRTA* finds better solutions in less time. Additionally, we performed a theoretical analysis; desirable properties, such as termination, hold for eLSS-LRTA*.

2. ESCAPING HEURISTIC DEPRESSIONS

The heuristic value of a state s in the search space is an estimation of the optimal cost incurred to reach a goal state. As such, a good heuristic is one that assigns higher values to states that are farther from the goal. In RT search problems, however, heuristics usually contain *depressions* [5].

Brief Sketch of the Algorithm Our algorithm, eLSS-LRTA*, is a simple yet effective modification of LSS-LRTA*. The description that follows assumes familiarity with LSS-LRTA* (details in [7]).

To escape heuristic depressions eLSS-LRTA* seeks a quick way out by explicitly *avoiding* states in a depression. The main conceptual difference between eLSS-LRTA* and its ancestor is that the former carries out its lookahead search by *preferring* states that are not in a depression. To achieve this, we slightly modify LSS-LRTA*'s lookahead procedure (originally an A*) to use two priority queues instead of a single one. In the first priority queue, $Open_1$, it inserts all states that are still not proven to be in a depression, whereas in the second queue, $Open_2$, it puts states that are known

Cite as: Escaping Heuristic Depressions in Real-Time Heuristic Search (Extended Abstract), Carlos Hernandez and Jorge A. Baier, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 1267-1268.

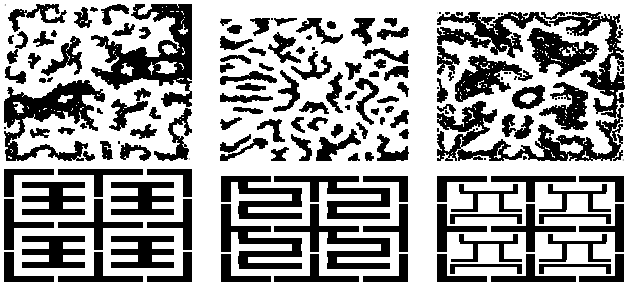

Figure 1: Game (top row) and office maps (bottom row). Areas of 2×2 rooms are shown for the office maps.

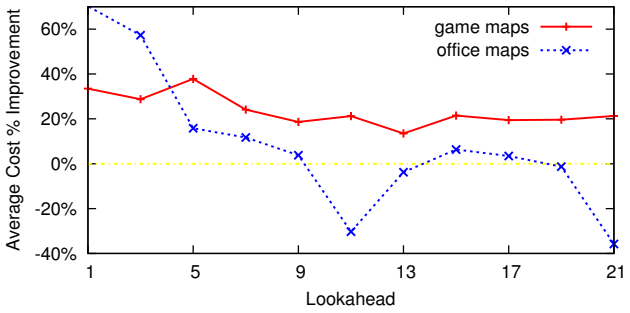

Figure 2: Average Percentage Cost Improvements

to be in a depression. While doing the lookahead search, the algorithm will always expand a state in $Open_1$ if possible, and will only expand a state from $Open_2$ if $Open_1$ is empty. Once the lookahead has been carried out, the strategy for updating the heuristic values is essentially the same as in LSS-LRTA*, but the update procedure is extended to *mark* the states that are inside a depression, so that in later iterations those marked states can be avoided. The learning phase of eLSS-LRTA* is a slight modification of LSS-LRTA*'s. It is a version of Dijkstra's algorithm that increases the heuristic of the states in the closed list of the A* search to the maximum value that maintains consistency. A state is marked if and only if its heuristic increased.

When LSS-LRTA* finishes the lookahead search, it decides what path to traverse by looking at the best state in the priority queue resulting from the A* search. eLSS-LRTA*, on the other hand, selects the best state in $Open_1$, if one exists, and else selects one from $Open_2$.

3. EVALUATION

We evaluated the algorithm theoretically and proved it has desirable properties. In particular if h is initially consistent, it remains consistent. eLSS-LRTA* is complete: if a solution exists, it is found. Experimentally, we compared eLSS-LRTA* with LSS-LRTA* in pathfinding tasks over initially unknown terrain. For fairness, we use comparable implementations with the same underlying code base.

The user-given h-values are the octile distances [9]. We use three computer game maps adapted from the game *World of Warcraft* (sizes: 169×169, 128×128, and 128×128) and three indoor office maps of 1000×1000 cells each (Fig. 1). For each test case in game maps, we choose the start and the goal cell randomly, ensuring they are sufficiently far apart.

Figure 2 shows average cost improvements averaged over 3000 test cases for the game maps (1000 test cases for each particular game map) and over 3000 test cases for the office maps (1000 test cases for each particular office map). Time per search episode is very similar for both algorithms, and thus total search time is proportional to solution cost. We used a Linux machine with a Pentium CoreQuad 2.33 GHz CPU and 8 GB RAM.

In game maps eLSS-LRTA* consistently outperforms LSS-LRTA*. Most significant improvements are produced for low values of the lookahead parameter. Improvements decrease as the lookahead parameter increases. In the office maps, we do observe significant improvements for small values of the lookahead parameter (1–5), however for higher values (>9) the quality may be degraded. We think this may be explained by the quality of the heuristic and the structure of the problem. The heuristic is more misleading in the office scenario than on the game scenario. In these problems the cell corresponding to the position of the agent usually lies in the interior of a big heuristic depression. When lookahead is carried out, most cells are marked as in a depression. Due to the structure of the problem, it is often the case that the agent finds an obstacle on its way. Thus, a new search is started from a states whose immediate neighbors are already marked. In that case eLSS-LRTA* behaves like LSS-LRTA*.

4. RELATED WORK

There exist algorithms that escape heuristic depressions by doing more lookahead or learning. Examples and RTAA* [6], LRTA*$_{\text{LS}}(k)$ [4]. LSS-LRTA* finds better-quality solutions than RTAA* for the same value of the lookahead parameter (though in more time) [6] . LSS-LRTA* is competitive with LRTA*$_{\text{LS}}(k)$ [4]. We are not aware of any algorithms that guide search towards escaping away of depressions.

Acknowledgements Carlos Hernandez was partly funded a by Fondecyt project #11080063. Jorge Baier was funded by the VRI-38-2010 grant from Universidad Católica de Chile.

5. REFERENCES

[1] Yngvi Björnsson, Vadim Bulitko, and Nathan R. Sturtevant. TBA*: Time-bounded A*. In *IJCAI*, pages 431–436, 2009.

[2] V. Bulitko and G. Lee. Learning in real time search: a unifying framework. *Journal of Artificial Intelligence Research*, 25:119–157, 2006.

[3] Peter E. Hart, Nils Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimal cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2), 1968.

[4] C. Hernandez and P. Meseguer. Improving LRTA*(k). In *IJCAI*, pages 2312–2317, 2007.

[5] Toru Ishida. Moving target search with intelligence. In *AAAI*, pages 525–532, 1992.

[6] S. Koenig and M. Likhachev. Real-time adaptive A*. In *AAMAS*, pages 281–288, 2006.

[7] Sven Koenig and Xiaoxun Sun. Comparing real-time and incremental heuristic search for real-time situated agents. *Autonomous Agents and Multi-Agent Systems*, 18(3):313–341, 2009.

[8] Richard E. Korf. Real-time heuristic search. *Artificial Intelligence*, 42(2-3):189–211, 1990.

[9] Nathan R. Sturtevant and Michael Buro. Partial pathfinding using map abstraction and refinement. In *AAAI*, pages 1392–1397, 2005.