

The Compressed Differential Heuristic*

Meir Goldenberg

ISE Department
Ben-Gurion University (Israel)
Mgoldenbe@yahoo.ca

Nathan Sturtevant

CS Department
Denver University (US)
Sturtevant@cs.du.edu

Ariel Felner

ISE Department
Ben-Gurion University (Israel)
Felner@bgu.ac.il

Jonathan Schaeffer

CS Department
University of Alberta (Canada)
Jonathan@cs.ualberta.ca

Abstract

The differential heuristic (DH) is an effective memory-based heuristic for explicit state spaces. In this paper we aim to improve its performance and memory usage. We introduce a compression method for DHs which stores only a portion of the original uncompressed DH, while preserving enough information to enable efficient search. Compressed DHs (CDH) can be tuned to fit any size of memory, even smaller than the size of the state space. Experimental results across different domains show that, for a given amount of memory, a CDH significantly outperforms an uncompressed DH.¹

1 Introduction and overview

True distance heuristics (TDHs) are a class of memory-based heuristics developed for explicit domains (Sturtevant et al. 2009; Felner et al. 2009; Goldenberg et al. 2010). A simple and effective TDH is the *differential heuristic* (DH) which was independently developed and used by a number of researchers (Goldberg and Harrelson 2005)(G&H), (Ng and Zhang 2002). A set of *pivot* states P is determined. A DH stores the exact distance from every state x to *all* states $p \in P$. We introduce the *compressed* DH (CDH) which stores a part of the DH's look-up table (hereafter, we refer to DH as the *regular* DH to distinguish from CDH). In CDH, a subset of the $|P|$ distances are stored for each state. When a distance from a state x to a pivot p is missing from the table, it is substituted by distances that were preserved by the compression. Any A^* -based search algorithm can use CDH.

Experimental results across multiple domains show that, for a given amount of memory, a CDH outperforms a regular DH of the same size. Furthermore, CDHs can be effectively used in applications, such as video games, where even the minimal amount of memory needed by a DH (i.e. the size of the domain) is not afforded. CDHs perform well because they are based on a larger DH containing a richer diversity of heuristic values. CDHs can be built without the need to create and store the entire uncompressed DH. While this paper focuses on DHs, the ideas are applicable to other types of TDHs.

*The full version will appear in AAAI-2011
Copyright © 2011, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

¹This research was supported by the Israeli Science Foundation (ISF) under grant no. 305/09 to Ariel Felner. Financial support of iCORE and NSERC to Jonathan Schaeffer is greatly appreciated.

2 DH and its compressed version

Given a weighted graph $G = (V, E, W)$, consider the problem of finding the shortest path between an arbitrary pair of states, $s, g \in V$ (a problem instance). Denote the cost of the shortest path between any two states $a, b \in V$ by $d(a, b)$. We assume that a base heuristic, $h_{base}(a, b)$, is defined for the domain (e.g., Manhattan distance for grid-based searches). Furthermore, we assume that $m|V|$ memory is available for storing the heuristic look-up table for some positive real number m . This table is computed once for a given input graph and is used across multiple problem instances.

Definition: Differential heuristic (DH). Choose $P \subset V$ ($|P| \ll |V|$) states called *pivots*, such that $|P| = m$. For each state $a \in V$, the look-up table of DH stores the distances from a to all pivots $p \in P$, i.e. exactly $m|V|$ distances $d(a, p)$ are stored in the table. For any given state a and pivot p , the following heuristic estimate of $d(a, g)$ with respect to p is admissible:

$$dh_p(a, g) = |d(a, p) - d(g, p)| \quad (1)$$

and the *differential heuristic* (DH) is:

$$dh(a, g) = \max \left\{ h_{base}(a, g), \max_{p \in P} dh_p(a, g) \right\}.$$

We refer to the DH just defined as the *regular* DH and illustrate it in Figure 1(left). Assume the search has reached state a and that a heuristic estimate to g is required. Let p_1 and p_2 be pivots, so that the DH stores $d(a, p_1) = 4$, $d(g, p_1) = 9$, $d(a, p_2) = 3$ and $d(g, p_2) = 14$. The resulting heuristic estimates with respect to the two pivots are $dh_{p_1}(a, g) = |4 - 9| = 5$ and $dh_{p_2}(a, g) = |3 - 14| = 11$. Therefore, $dh(a, g) = \max\{5, 11\} = 11$. One can easily verify that the regular DH is admissible and consistent.

2.1 Compressed DH

A CDH stores part of the information contained in the regular DH. For each state $a \in V$, only distances to pivots from a subset $P_a \subset P$ are stored. In Figure 1(right), $P_a = \{p_1, p_2\}$ but $P_g = \{p_1\}$. Solid edges represent distances that are stored in the CDH.

Assume that the A^* search reached state a and that an admissible estimate to the goal, $h(a, g)$, is needed. Let p be a pivot to which the distance from a , $d(a, p)$, is stored in the CDH (i.e. $p \in P_a$). There are now two cases.

Case 1: $d(g, p)$ is also stored. In this case, the CDH with respect to p is calculated as in the regular DH: $cdh_p(a, g) =$

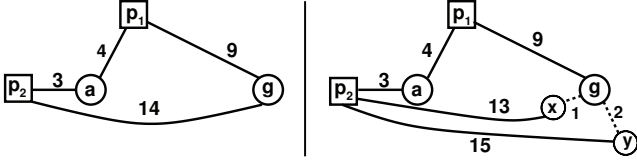


Figure 1: Regular DH (left) and compressed DH (right).

$dh_p(a, g)$. The pivot p_1 in Figure 1(right) demonstrates this case, since distance to p_1 is stored both from a and from g . Therefore, $cdh_{p_1}(a, g) = dh_{p_1}(a, g) = |4 - 9| = 5$.

Case 2: $p \notin P_g$ (p_2 in our example). In this case, $cdh_p(a, g)$ is defined as follows. Equation 1 is based on the inequalities $d(a, g) \geq d(a, p) - d(g, p)$ and $d(a, g) \geq d(g, p) - d(a, p)$. Therefore, equation 1 can be re-written as

$$dh_p(a, g) = \max \{d(a, p) - d(g, p), d(g, p) - d(a, p)\}. \quad (2)$$

Let $\bar{d}(g, p)$ and $\underline{d}(g, p)$ denote an upper and lower bound on the missing distance $d(g, p)$. Using them with equation 2 gives the following admissible heuristic with respect to p :

$cdh_p(a, g) = \max \{d(a, p) - \bar{d}(g, p), \underline{d}(g, p) - d(a, p)\}$. Maximizing over all pivots yields the *compressed differential heuristic* (CDH):

$$cdh(a, g) = \max \left\{ h_{base}(a, g), \max_{p \in P_a} cdh_p(a, g) \right\}.$$

2.2 Computing the bounds

For a given problem instance, the goal state remains unchanged for the duration of the search. Therefore, for each pivot p , the bounds $\bar{d}(g, p)$ and $\underline{d}(g, p)$ can be precomputed before the main search begins. These bounds are stored in a table called the *bounds table*, to be used throughout the search for this particular problem instance.

Let $p \in P$ be a pivot to which the distance from the goal state g , $d(g, p)$, is not stored as in Case 2) above. Let x be an arbitrary state such that the distance from x to p is stored. More formally, $p \notin P_g$ and $p \in P_x$. The triangle inequality for the vertices $\{g, x, p\}$ gives the following bounds on $d(g, p)$ with respect to x :

$$\bar{d}_x(g, p) = d(g, x) + d(x, p), \text{ and}$$

$$\underline{d}_x(g, p) = |d(g, x) - d(x, p)| \quad (3)$$

The *Bounding* procedure calculates the bounds by applying equation 3 to states that are in close proximity to g as follows. Initially, for each pivot $p \notin P_g$, we set $\bar{d}(g, p) = \infty$ and $\underline{d}(g, p) = h_{base}(g, p)$. Perform a Dijkstra search from g . For each expanded state x and for each $p \in P_x$, we use the distance $d(x, p)$ to compute the bounds $\bar{d}_x(g, p)$ and $\underline{d}_x(g, p)$. These bounds are then used to improve the global upper and lower bounds on $d(g, p)$. This process is continued until at least r distances $d(x, p)$ have been seen for each pivot $p \notin P_g$, where r is a parameter.

3 Experimental results

We performed experiments on the 3 real-world domains (shown in Figure 2): **(1) Dragon Age maps** taken from Bioware's *Dragon Age: Origins*



Figure 2: The environments.

(DAO)(<http://movingai.com/benchmarks>). **(2) Road maps** of North American cities and states. **(3) Robotic Arm:** A segmented arm moves in two-dimensional space and can have any number of segments. Here, in Figure 3, we only provide results for the road and the game maps. However, we also confirmed the trends on the Robotic Arm domain and on two synthetically generated environments: rooms maps and Delaunay graphs.

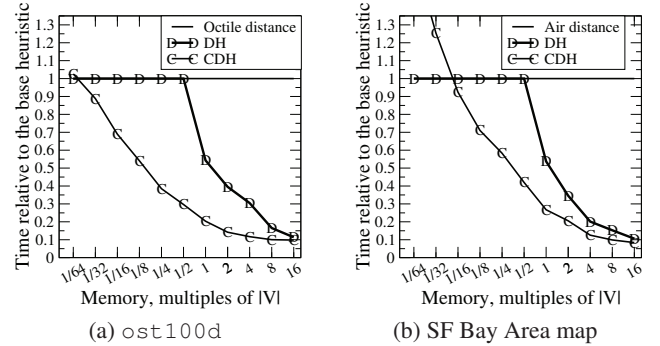


Figure 3: Pathfinding experiments.

All times that we report are measured in milliseconds. Charts (a) and (b) show the time performance of both the regular and the compressed DH for the *oct100d* DAO map and the SF Bay Area road map, respectively. For the memory amounts between $1|V|$ and $8|V|$, CDH significantly outperforms the regular DH. When larger amounts of memory are available, regular DHs achieve good coverage of the state space, leaving little space for improvement.

References

- Felner, A.; Berrer, M.; Sturtevant, N.; and Schaeffer, J. 2009. Abstraction-based heuristics with true distance computations. In *Proceedings of SARA*.
- Goldberg, A., and Harrelson, C. 2005. Computing the shortest path: A* search meets graph theory. In *SODA*, 156–165.
- Goldenberg, M.; Felner, A.; Sturtevant, N.; and Schaeffer, J. 2010. Portal-based true-distance heuristics for path finding. In *Proceedings of SoCS*.
- Ng, T., and Zhang, H. 2002. Predicting internet network distance with coordinates-based approaches. In *INFOCOM*, 170–179.
- Sturtevant, N.; Felner, A.; Barer, M.; Schaeffer, J.; and Burch, N. 2009. Memory-based heuristics for explicit state spaces. *IJCAI* 609–614.