Goal-Oriented Euclidean Heuristics with Manifold Learning

Wenlin Chen*, Yixin Chen*, Kilian Q. Weinberger*, Qiang Lu†, Xiaoping Chen†
*Dept. of Computer Science and Engineering, Washington University in St. Louis
†School of Computer Science and Technology, University of Science and Technology of China
wenlinchen@wustl.edu, chen@cse.wustl.edu, kilian@wustl.edu, {qianglu8, xpchen}@ustc.edu.cn

Abstract

Recently, a Euclidean heuristic (EH) has been proposed for A* search. EH exploits manifold learning methods to construct an embedding of the state space graph, and derives an admissible heuristic distance between two states from the Euclidean distance between their respective embedded points. EH has shown good performance and memory efficiency in comparison to other existing heuristics such as differential heuristics. However, its potential has not been fully explored. In this paper, we propose a number of techniques that can significantly improve the quality of EH. We propose a goal-oriented manifold learning scheme that optimizes the Euclidean distance to goals in the embedding while maintaining admissibility and consistency. We also propose a state heuristic enhancement technique to reduce the gap between heuristic and true distances. The enhanced heuristic is admissible but no longer consistent. We then employ a modified search algorithm, known as B$'$ algorithm, that achieves optimality with inconsistent heuristics using consistency check and propagation. We demonstrate the effectiveness of the above techniques and report un-matched reduction in search costs across several non-trivial benchmark search problems.

Introduction

A^* search is critical in many areas of real life. For example, GPS navigation systems need to find the shortest path between two locations *efficiently* and *repeatedly* (*e.g.* each time a new traffic update has been received, or when the driver makes a wrong turn) (Geisberger et al. 2008). As the processor capabilities of these devices and the patience of the users are both limited, the quality of the search heuristic is of great importance. This importance only increases as more and more *low powered* embedded devices (*e.g.* smartphones) are equipped with similar capabilities. Other applications include massive online multiplayer games (Sturtevant 2007), where artificial agents need to identify the shortest path along a map, which can change dynamically through actions by other users.

Designing heuristics for such problems is important but difficult. For an A^* search to find optimal paths, the heuristic must be *admissible*. Moreover, for a search to be fast, heuristic distances must be *consistent* and close to true distances. An accurate heuristic is crucial for time sensitive applications, *e.g.* GPS driver assistance systems, whose query time is currently in the low microsecond range. Moreover, embedded devices have limited memory, demanding a compact representation of the heuristic. For example, given n states in the search space, one can obtain a perfect heuristic by storing all pre-computed distances between any two states—however the $O(n^2)$ memory requirement renders this approach infeasible in practice.

Recently, Rayner, Bowling, and Sturtevant (2011) introduced a novel heuristic for A^* search, which they refer to as the *Euclidean heuristic (EH)*. It uses a manifold learning algorithm, Maximum Variance Unfolding (MVU) (Weinberger and Saul 2006), to embed the state space graph into a low dimensional Euclidean space. The Euclidean distance between two states in the embedding is then used as the heuristic distance for A^* search. The embedding is constrained to keep the heuristic admissible and consistent, by underestimating all distances. Further, the objective minimizes the differences between the heuristic estimates and their corresponding true distances. Moreover, it reduces the space requirement for storing heuristics from $O(n^2)$ to $O(dn)$, where d is the dimensionality of the embedding. EH achieves impressively better performance than other leading heuristics (Rayner, Bowling, and Sturtevant 2011).

However, MVU with its original semi-definite relaxation can only embed state spaces with up to a few thousand states—severely limiting its use in practice. Although there have been efforts to increase the scalability of MVU (Weinberger, Packer, and Saul 2005; Weinberger et al. 2007), these lead to approximate solutions and the resulting heuristics violate admissibility. A more recent variation of MVU, referred to as Maximum Variance Correction (MVC) (Chen, Weinberger, and Chen 2013), greatly improves its scalability (and accuracy) and can embed graphs with $200K$ states or more. MVC generates approximate MVU solutions but still guarantees admissibility and consistency of the heuristics. The EH heuristics, computed via MVC, lead to significant reduction in search time, even beating the competitive differential heuristic (Ng and Zhang 2002) by a large factor (Chen, Weinberger, and Chen 2013).

With the development of MVC, EH becomes an attractive and scalable choice for computing heuristics. However, its potential is yet to be fully explored. The objective of MVU/MVC minimizes the sum of the distance gaps between all pairs of states, while A^* search is guided only by the heuristic distance to the goal state. In many applications, possible (or likely) goal states form a small subset of the overall state space. For example, GPS driving assistants are mostly used to find directions to registered street addresses, which are a tiny fraction of all possible locations on a map. This suggests that the EH can be further improved by "tightening" the heuristic distances towards likely goal states.

Moreover, state graphs are typically not isometric to low dimensional Euclidean spaces, which leads to distance gaps between the EH heuristic and the true distances—in particular, for states that are far away from each other. We find that it is possible to compactly encode the information about distance gaps, using only a small amount of memory, and significantly improve the search performance by correcting heuristic estimates on-the-fly.

Our main contributions include, 1) a goal-oriented graph embedding framework that optimizes the heuristic distance to goals while preserving admissibility and consistency, and 2) an in-memory enhancement technique that can be used online to speed up search. Since the enhancement technique generates better heuristics that are admissible but no longer consistent, we employ the B$'$ search to make sure that the first solution found is an optimal one.

Background

The A^* search algorithm finds the shortest path between two nodes in a graph. Let $G = (V, E)$ denote such a graph with undirected edges E and nodes V, with $|V| = n$. Edges $(i, j) \in E$ are weighted by some cost, or length, $d_{ij} \geq 0$ (often $d_{ij} = 1$ for all i, j). In the worst case, a naive algorithm needs to exhaustively explore the whole graph which is expensive. But the search time can be reduced drastically with a good heuristic, which estimates the shortest distance δ_{ij} between two nodes $i, j \in V$ in the graph.

Euclidean heuristic (EH) and MVU

Rayner, Bowling, and Sturtevant (2011) propose the *Euclidean Heuristic*. They advocate to embed the graph with MVU (Weinberger and Saul 2006) into a Euclidean space as $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{R}^d$, such that $\|\mathbf{x}_i - \mathbf{x}_j\|_2 \leq \delta_{ij}$, where δ_{ij} denotes the graph distance between nodes i, j. The Euclidean distance $h(i,j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2$ can then be used as an efficient heuristic estimate of the graph distance δ_{ij}.

The authors prove that A^* with this heuristic is optimal, as the heuristic is admissible and consistent. More precisely, for all nodes i, j, k the following holds:

$$\text{Admissibility:} \quad \|\mathbf{x}_i - \mathbf{x}_k\|_2 \leq \delta_{ik} \tag{1}$$

$$\text{Consistency:} \quad \|\mathbf{x}_i - \mathbf{x}_j\|_2 \leq \delta_{ik} + \|\mathbf{x}_k - \mathbf{x}_j\|_2 \tag{2}$$

The proof is straightforward. Consistency follows from the triangular inequality in combination with (1) and admissibility is enforced by the MVU optimization problem implicitly through the stricter constraints $h(i,j) \leq d_{ij}$ for all $(i,j) \in E$ where d_{ij} is the edge length.

The closer the gap in the admissibility inequality (1), the better is the search heuristic. The perfect heuristic would be the actual shortest path, $h(i,j) = \delta_{ij}$ (with which A^* could find the exact solution in linear time with respect to the length of the shortest path). MVU narrows this gap by maximizing all pairwise distances as its objective, *i.e.* $\sum_{i,j} \|\mathbf{x}_i - \mathbf{x}_j\|^2$.

As the resulting heuristic is invariant to translation, a centering constraint, $\sum_i \mathbf{x}_i = \mathbf{0}$, is added for mathematical convenience. This additional constraint reduces the objective to $\sum_i \mathbf{x}_i^\top \mathbf{x}_i$. The MVU optimization then becomes:

$$\begin{aligned} \underset{\mathbf{x}_1,\ldots,\mathbf{x}_n \in \mathcal{R}^d}{\text{maximize}} \quad & \sum_{i=1}^{n} \mathbf{x}_i^\top \mathbf{x}_i \\ \text{subject to} \quad & ||\mathbf{x}_i - \mathbf{x}_j||_2 \leq d_{ij}, \quad \forall (i,j) \in E \\ & \sum_{i=1}^{n} \mathbf{x}_i = 0 \end{aligned} \tag{3}$$

Although (3) is non-convex, (Weinberger and Saul 2006) show that with a rank relaxation, this problem can be rephrased as a convex semi-definite program (SDP) by optimizing over the inner-product matrix $\mathbf{K}$, with $k_{ij} = \mathbf{x}_i^\top \mathbf{x}_j$:

$$\begin{aligned} \underset{\mathbf{K}}{\text{maximize}} \quad & \text{trace}(\mathbf{K}) \\ \text{subject to} \quad & k_{ii} - 2k_{ij} + k_{jj} \leq d_{ij}^2, \quad \forall (i,j) \in E \\ & \sum_{i,j} k_{ij} = 0 \\ & \mathbf{K} \succeq 0. \end{aligned} \tag{4}$$

The final constraint $\mathbf{K} \succeq 0$ ensures positive semi-definiteness and guarantees that $\mathbf{K}$ can be decomposed into Euclidean vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$ with a straight-forward eigenvector decomposition. To ensure strictly $d-$dimensional output, the final embedding is projected into $\mathcal{R}^d$ with principal component analysis (PCA). (This is identical to composing the vectors $\mathbf{x}_i$ out of the d leading eigenvectors of $\mathbf{K}$.) With c constraints, the time complexity of MVU becomes $O(n^3 + c^3)$, which makes it prohibitive for larger data sets.

Maximum Variance Correction (MVC)

Recently, Chen, Weinberger, and Chen (2013) introduced MVC to scale up MVU by several orders of magnitude. It exploits the specific property of the MVU optimization that all distance constraints are strictly *local*. As a first step, MVC uses fast but inaccurate approximation methods such as isomap (Tenenbaum, Silva, and Langford 2000) or MVU approximations (Weinberger, Packer, and Saul 2005; Weinberger et al. 2007) and re-scaling to obtain an initial feasible solution $\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n$ to (3). This initial embedding is feasible but far from optimal. The second step of MVC is to refine the embedding to maximize its variance, the objective in (3), while maintaining feasibility.

MVC partitions the graph $G = (V, E)$ into r connected sub-graphs called *patches*: $G_p = (V_p, E_p)$, where $V = V_1 \cup \cdots \cup V_r$ and $V_p \cap V_q = \{\}$ for all p, q. There are two types of nodes within a partition V_p. A point $i \in V_p$ is an *anchor* *point* of V_p if there exists an edge $(i,j)\in E$ connecting i to some node $j\notin V_p$ outside the patch; Otherwise, $\mathbf{x}_i\in V_p$ is an *inner point* of V_p. Let V_p^x denote the set of all inner points and V_p^a the set of all anchor points in V_p.

Based on such partitioning, MVU is decomposed into r independent optimization problems of the following type:

$$\begin{aligned} \underset{\mathbf{x}_i\in V_p^x}{\text{maximize}} \quad & \sum_{i\in V_p}\mathbf{x}_i^\top\mathbf{x}_i \\ \underset{\mathbf{a}_k\in V_p^a}{\text{subject to}} \quad & ||\mathbf{x}_i-\mathbf{x}_j||_2\le d_{ij},\ \forall(i,j)\in E_p^{xx} \\ & ||\mathbf{x}_i-\mathbf{a}_k||_2\le d_{ik},\ \forall(i,k)\in E_p^{ax} \end{aligned} \tag{5}$$

where E_p^{xx} contains only edges between inner points of V_p, and E_p^{ax} contains only edges between anchor and inner points. The optimization (5) can be further reduced to an SDP and the solutions of the r sub-problems can be combined into a globally feasible solution to (3).

The anchor points are fixed in place, which ensures that no constraint in (5) involves two nodes in different patches. Consequently, the r sub-problems are completely independent and can be solved in parallel with off-the-shelf SDP solvers (Borchers 1999). Each iteration, different patches are selected and re-optimized. The algorithm terminates once the embedding converges. Chen, Weinberger, and Chen (2013) show that MVC can solve MVU problems of unprecedented scale and demonstrate that it can even obtain higher variance solutions than the convex MVU relaxation (4), as it requires no rank relaxation.

Goal-oriented Euclidean heuristic

We make several assumptions about our problem domain: The state-space is represented as an undirected graph with n nodes, which fits into memory. We are asked to solve search problems repeatedly, with different starting and goal states. We would like to minimize the time required for these search problems through better heuristics. The heuristics can be optimized offline, where computation time is of no particular importance, but the storage required to encode the heuristic must be small (as it might need to be communicated to the consumers and loaded into memory).

In many A^* search applications, some states are much more likely to be considered goal states than others. For example, in video games, certain locations (*e.g.* treasure targets) are typical goals that a players might want to move to, whereas most arbitrary positions in the game space are moved to very infrequently. In the subsequent section we describe a robot planning domain, where robots can pick up objects from pre-specified source locations and move them to pre-specified target locations. Here, there exists a strict subset of states that could potentially become goal states—important information that ideally should be incorporated into the heuristic design. As a final example, GPS navigation assistants are typically exclusively used to direct users to registered home addresses—a tiny subset of all possible locations on the map. Further, a particular user might only traverse between no more than 100 locations on the world map. It is fair to assume that she might prefer accelerating the search for these frequent goals, even at the cost that the search for infrequent goals takes a little longer.

Let $G=(V,E)$ be the state-space graph. For convenience, we assume that there is a subset of possible goal states $V_G\subseteq V$. When we know a set of possible goals, we can improve EH by maximizing the distances to the goals. Note that during an A^* search, only those distances to the particular goal state will be used as the heuristic function (h), and heuristic distances between two non-goal states are never used during search. This motivates us to modify the objective function in the EH/MVU optimization so that it exclusively maximizes distances to goals. Note that we can still guarantee admissibility and consistency of EH, since we keep all local constraints.

The proposed *goal-oriented Euclidean heuristic* (GOEH) solves the following:

$$\begin{aligned} \underset{\mathbf{x}_1,\ldots,\mathbf{x}_n\in\mathcal{R}^d}{\text{maximize}} \quad & \sum_{i=1..n,g\in V_G}\|\mathbf{x}_i-\mathbf{x}_g\|_2^2 \\ \text{subject to} \quad & ||\mathbf{x}_i-\mathbf{x}_j||_2\le d_{ij},\quad \forall(i,j)\in E \\ & \sum_{i=1}^n\mathbf{x}_i=0 \end{aligned} \tag{6}$$

Due to the centering constraint $\sum_{i=1}^n\mathbf{x}_i=0$, the objective in (6) can be further reduced to

$$\underset{\mathbf{x}_1,\ldots,\mathbf{x}_n\in\mathcal{R}^d}{\text{maximize}}\ n_g\sum_{i=1..n}\|\mathbf{x}_i\|_2^2+n\sum_{g\in V_G}\|\mathbf{x}_j\|_2^2, \tag{7}$$

where $n_g=|V_G|$ is the number of goals.

Following the same rank relaxation as in MVU (Weinberger and Saul 2006) and considering (7), (6) can be rephrased as a convex SDP by optimizing over the inner-product matrix $\mathbf{K}$, with $k_{ij}=\mathbf{x}_i^\top\mathbf{x}_j$:

$$\begin{aligned} \underset{\mathbf{K}}{\text{maximize}} \quad & n_g\text{trace}(\mathbf{K})+n\sum_{g\in V_G}k_{gg} \\ \text{subject to} \quad & k_{ii}-2k_{ij}+k_{jj}\le d_{ij}^2,\quad\forall(i,j)\in E \\ & \sum_{i,j}k_{ij}=0 \\ & \mathbf{K}\succeq 0. \end{aligned} \tag{8}$$

The final embedding is projected onto $\mathcal{R}^d$ by composing the vectors $\mathbf{x}_i$ out of the d leading eigenvectors of $\mathbf{K}$.

Note that a general weighted EH model is already discussed in (Rayner, Bowling, and Sturtevant 2011). However, GOEH makes the goal-oriented cases explicit.

Figure 1 illustrates the effects of GOEH. We can see that GOEH "stretches" the embedding so that the goal states are further away from other states, while the local distance constraints still ensure the admissibility and consistency.

The time complexity for solving the SDP in (8) is the same as the original MVU formulation, which makes it prohibitive for larger data sets. To improve its scalability, we propose a modified version of MVC for solving (6). We follow the same steps in MVC to generate the initial embedding and r patches. However, each subproblem in (5) for patch p

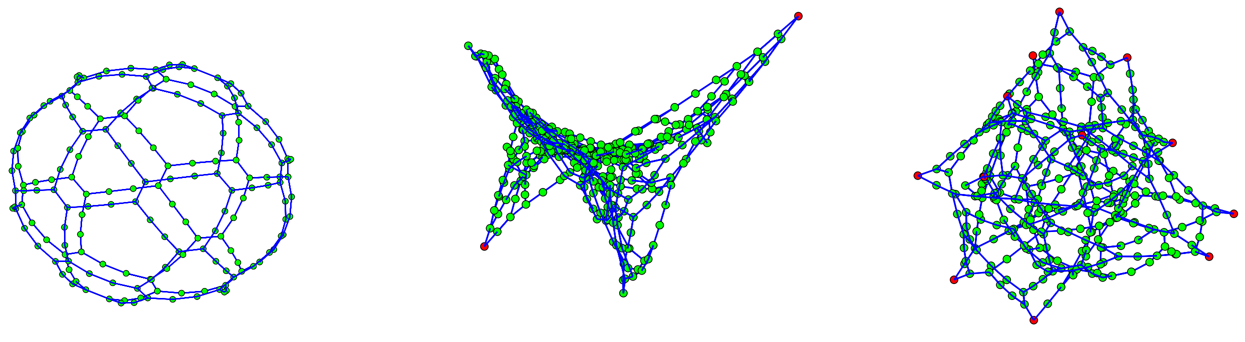

(a) Original EH embedding (b) GOEH with 2 goals (c) GOEH with 12 goals

Figure 1: EH and GOEH embeddings ($d = 3$) illustrated on a 5-puzzle problem. Goal states are colored in red, others in green. In this spherical embedding in (a), the Euclidean distance is a bad approximation for distant states. GOEH deforms the embedding to better reflect the spherical distance from far away states to the goal states—at the cost of shrinkage between non-goal states.

is changed to:

$$\begin{aligned} \underset{\mathbf{x}_i \in V_p^x}{\text{maximize}} \quad & \sum_{i \in V_p, g \in V_G} \|\mathbf{x}_i - \mathbf{x}_g\|_2^2 \\ \underset{\mathbf{a}_k \in V_p^a}{\text{subject to}} \quad & \|\mathbf{x}_i - \mathbf{x}_j\|_2 \le d_{ij}, \;\; \forall (i,j) \in E_p^{xx} \\ & \|\mathbf{x}_i - \mathbf{a}_k\|_2 \le d_{ik}, \;\; \forall (i,k) \in E_p^{ax}, \end{aligned} \tag{9}$$

Intuitively, for the subproblem of each patch, we maximize the distances of its inner nodes to the goals, while still enforcing the local distance constraints involving inner nodes and anchor nodes.

Following a similar approach as in MVC (Chen, Weinberger, and Chen 2013), we reformulate (9) as a SDP. Given a patch $G_p = (V_p, E_P)$ with $n_p = |V_p|$, we define a design matrix $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathcal{R}^{d \times n_p}$, where each column corresponds to one embedding coordinate of V_p^x. Define the matrix $K \in \mathcal{R}^{(d+n_p)\times(d+n_p)}$ as:

$$\mathbf{K} = \begin{pmatrix} \mathbf{I} & \mathbf{X} \\ \mathbf{X}^\top & \mathbf{H} \end{pmatrix} \quad \text{where } \mathbf{H} = \mathbf{X}^\top \mathbf{X}. \tag{10}$$

Let the vector $\mathbf{e}_{i,j} \in \mathcal{R}^{n_p}$ be all-zero except the i^{th} element is 1 and the j^{th} element is -1. Let the vector $\mathbf{e}_i$ be all-zero except the i^{th} element is -1. With this notation, and the Schur Complement Lemma (Boyd and Vandenberghe 2004), we can relax (9) into a SDP:

$$\begin{aligned} \max_{\mathbf{X},\mathbf{H}} \quad & \sum_{i \in V_p^x} \left[n_g \mathbf{H}_{ii} - \sum_{g \in V_p^1} 2\bar{\mathbf{x}}_g^\top \mathbf{x}_i + \sum_{g \in V_p^2} (\mathbf{H}_{gg} - \mathbf{H}_{ig}) \right] \\ \text{s.t.} \quad & (\mathbf{0}; \mathbf{e}_{ij})^\top \mathbf{K} (\mathbf{0}; \mathbf{e}_{ij}) \le d_{ij}^2 \;\; \forall (i,j) \in E_p^{xx} \\ & (\mathbf{a}_k; \mathbf{e}_i)^\top \mathbf{K} (\mathbf{a}_k; \mathbf{e}_i) \le d_{ik}^2 \;\; \forall (i,k) \in E_p^{ax} \\ & \mathbf{K} = \begin{pmatrix} \mathbf{I} & \mathbf{X} \\ \mathbf{X}^\top & \mathbf{H} \end{pmatrix} \succeq 0 \end{aligned} \tag{11}$$

where $V_p^1 = V_G \backslash V_p^x$, $V_p^2 = V_G \cap V_p^x$, $\mathbf{H}_{ij}$ are elements in $\mathbf{H}$, $\mathbf{x}_i$ is the i^{th} column of $\mathbf{X}$, and $\bar{\mathbf{x}}_g$ is the coordinate for node g. The optimization (11) is a convex SDP and can be solved very efficiently for medium sized n_p. The r sub-problems are completely independent and can be solved *in parallel*, leading to almost perfect parallel speed-up on multi-core computers or clusters. Like MVC, we reiterate solving the r subproblems until convergence.

Both the centralized solution in (8) and the partitioned solution in (11) maintain the admissibility and consistency of EH, because the constraints are kept the same as in (3), and the proof for admissibility and consistency of EH only relies on these constraints (Rayner, Bowling, and Sturtevant 2011).

Proposition 1. *Any feasible solution to (8) or (11) gives admissible and consistent Euclidean heuristics.*

The memory requirement for storing the embedding results $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{R}^d$ is still $O(dn)$, which is reasonable since d is often a small constant such as 3 or 4.

State heuristic enhancement

We propose a *state heuristic enhancement* (SHE) technique to further improve the quality of GOEH. Since GOEH gives a lower bound of the true distance, we can calculate their gaps in a preprocessing phase and store the information in the memory to aid the search.

Suppose we are given a state-space graph $G = (V, E)$ and a goal set $V_G \subseteq V$. After we use GOEH to generate a d-dimensional embedding $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{R}^d$, for any $i = 1, \cdots, n$, we store a real number η_i defined as:

$$\eta_i = \min_{j \in V_G} \left\{ d_{ij} - \|\mathbf{x}_i - \mathbf{x}_j\|_2 \right\}. \tag{12}$$

During the search towards any goal g, for any state i, its enhanced heuristic value will be

$$h(i, g) = \|\mathbf{x}_i - \mathbf{x}_g\|_2 + \eta_i. \tag{13}$$

Intuitively, η_i stores the minimum gap between the EH and true distance from state i to any goal state in the goal set. During a search, we can add η_i to the Euclidean distance from i to g in the embedded space. Clearly, we have:

Proposition 2. *The heuristic function in (13) is* admissible.

However, the heuristic in (13) is no longer guaranteed to be consistent. A parent node i may receive a much larger η_i than the η_j received by a child node j, so that $h(i,g) > d_{ij} + h(j,g)$, leading to inconsistency.

As found in (Zahavi et al. 2007), inconsistent but admissible heuristics can often be preferable to consistent heuristics. To ensure optimality of the first solution found, we employ the B$'$ algorithm proposed in (Mero 1984).

Suppose the current expanded node is i, the rules of the B$'$ algorithm to handle inconsistency are:

a) For each successor j of i, if $h(j,g) < h(i,g) - d_{ij}$, set $h(j,g) = h(i,g) - d_{ij}$

b) Let j be the successor of i with minimum $h(j,g) + d_{ij}$. If $h(i,g) < h(j,g) + d_{ij}$, set $h(i,g) = h(j,g) + d_{ij}$

It is shown in (Mero 1984) that adding those rules to A^* search makes it optimal even when the heuristic is inconsistent. They can actually further improve the heuristic since it propagates improvements on the heuristic values based on consistency constraints. It is also shown that if the heuristic is consistent, then these rules will never be invoked and the B$'$ search is exactly the same as the A^* search (Mero 1984). Experimentally, we found that inconsistency does not occur often. For many problem instances we tested, the enhanced heuristics are consistent. For other instances, these B$'$ rules are invoked for some states to improve their GOEH values.

We note that there is a tradeoff between space complexity and heuristic accuracy. SHE uses $O(n)$ memory since it stores one real number per state, which adds only $1/d$ overhead to the $O(dn)$ space used by EH and GOEH. As we will see, this simple SHE technique can drastically improve the efficiency of optimal search on every domain we test. Note that EH cannot utilize dimensions that exceed the intrinsic dimensionality of the state space. This is in contrast to SHE, which does take advantage of such "extra" dimensions.

Experimental results

We evaluate our algorithms on two well-known benchmark AI problems and on a real world service robot application.

M-Puzzle Problem (Jones 2008) is a popular benchmark problem for search algorithms. It consists of a frame of M tiles and one tile missing. All tiles are numbered and a state constitutes any order of the tiles. An action is to move a cardinal neighbor tile of the empty space into the empty space. The task is to find a shortest action sequence from a predefined start to a goal state. We evaluate our algorithm on 7- and 8-puzzle problems (4×2 and 3×3 frames), which contain 20160 and 181440 states, respectively.

Blocks World (Gupta and Nau 1992) is a NP-hard problem with the goal to build several given stacks out of a set of blocks. Blocks can be placed on the top of others or on the ground. Any block that is currently under another block cannot be moved. We test blocks world problems with 6 blocks (4,051 states) and 7 blocks (37,633 states), respectively.

Home Service Robot is designed to handle daily indoor activities and can interact with human. In the Home Service Robot Competition at the RoboCup (http://robocup.rwth-aachen.de/athomewiki/index.php/Main_Page), there are a service robot, human, small objects that can be picked up and moved by the robot, and some big objects that cannot be moved. A typical task is to ask the robot to pick up a small object and bring it to the human. In this problem, each state describes different variables, such as the locations of the robot, human and small objects. There are a lot of intermediate states that the robot never takes as the goal. The robot can store the precomputed GOEH and SHE information in memory and use it to solve various daily tasks online.

Goal sets. We design three kinds of goal settings for our datasets. For M-puzzles, we let the goal set include all the states where the first three tiles are blank, tile 1 and tile 2, respectively. In this case, all goal states are distributed uniformly on the surface of the embedded sphere (see Figure 1c). For blocks world problems, we randomly pick a state and add more states in its vicinity to the goal set. For the home service robot application, the goal states are built-in goals for completing specific tasks from the competition. In this way, we evaluate our algorithms on the scenarios of distributed goals, clustered goals, and real-world goals, leading to a comprehensive assessment.

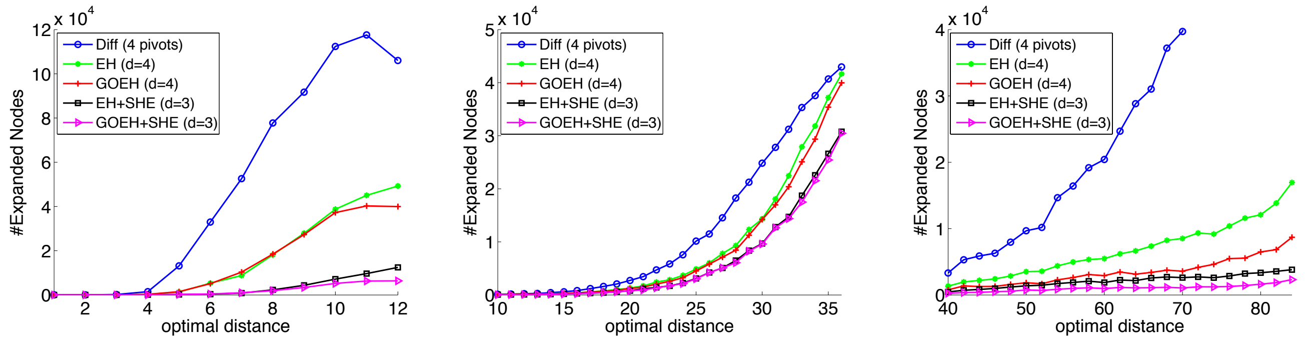
Experimental setup. We use MVC to learn GOEH and use Isomap (Tenenbaum, Silva, and Langford 2000) to initialize MVC. For datasets of a size greater than 8K, we set 8K landmarks for Isomap. For MVC we use a patch size of $m = 500$ throughout (for which problem (11) can be solved in less than $20s$). Following (Rayner, Bowling, and Sturtevant 2011; Chen, Weinberger, and Chen 2013), we choose a small embedding dimensionality for saving memory space. We set the number of embedding dimensions $d = 4$ for all experiments, meaning that each heuristic stores $4n$ real numbers. Since SHE requires another array of n numbers, the embedding dimension is set to $d = 3$ when SHE is used. This gives a fair comparison since all algorithms use $4n$ space. In our experiments, all MVC or goal-oriented MVC algorithms are stopped after a maximum of 50 iterations.

We also test the differential heuristic (Ng and Zhang 2002; Sturtevant et al. 2009) as a baseline. The differential heuristic pre-computes the exact distance from all states to a few pivot nodes in a set $S \subseteq V$. We implemented the algorithm to select good pivot nodes in (Sturtevant et al. 2009). The differential heuristic between two states a, b is $\max_{s\in S} |\delta(a,s) - \delta(b,s)| \leq \delta(a,b)$. In our experiments, we set the number of pivot nodes to 4 so that both differential heuristics and Euclidean heuristics have the same space complexity of $4n$.

Comprehensive evaluation. Figure 2 shows the total expanded nodes of three problems as a function of the optimal solution length, averaged over 500 start/goal pairs for each solution length. Each goal is randomly chosen from the goal set V_G. The figures compare the performance of the differential heuristic (Diff), EH, GOEH, EH combined with SHE (EH+SHE), and GOEH+SHE. To quantify the performance improvement, Table 1 shows the speedup of each method over the differential heuristic, defined as:

$$\text{Speedup(M)} = \frac{\sum_l \sum_{p=1}^{500} \text{NumExpand}(l,p,\text{Diff})}{\sum_l \sum_{p=1}^{500} \text{NumExpand}(l,p,\text{M})}$$

where NumExpand(l,p,M) is the number of states expanded



(a) 7-blocksworld (b) 7-puzzle (c) robot2

Figure 2: The number of expanded nodes in the optimal search as a function of the optimal solution length. For EH+SHE and GOEH+SHE, B′ search is used and the re-opened nodes are counted as new expansions.

Problem	Diff	EH	GOEH	EH+SHE	GOEH+SHE
6-block	1.00	3.83	5.77	5.02	**6.26**
7-block	1.00	3.02	3.46	15.06	**23.30**
7-puzzle	1.00	1.40	1.53	2.01	**2.08**
8-puzzle	1.00	1.25	1.36	**3.20**	3.18
robot1	1.00	5.09	14.14	17.12	**26.83**
robot2	1.00	4.63	9.19	14.68	**29.28**

Table 1: Speedup of various methods as compared to the differential heuristic.

Problem	n	n_g	EH	GOEH	EH+SHE	GOEH+SHE
6-block	$4K$	60	9m	10m	9m+1s	9m+1s
7-block	$37K$	100	56m	62m	51m+9s	60m+9s
7-puzzle	20K	120	40m	49m	40m+6s	45m+7s
8-puzzle	181K	720	7h	11h	6h+3m	10h+3m
robot1	50K	80	5h	6h	4h+15s	6h+15s
robot2	90K	100	7h	10h	7h+33s	9h+32s

Table 2: The total number of states (n), size of goal sets (n_g), and training time for various heuristics on different problems. For EH+SHE and GOEH+SHE, we also report the additional time for computing SHE.

by algorithm M to solve the p^{th} start/goal pair under solution length l. From Figure 2 and Table 1, we can see that each of GOEH and SHE dramatically reduces the number of expanded states. Combining them gives the most reduction.

Embedding time. Table 2 shows the training time for each embedding algorithm. Note that for real deployment, such as GPS systems, MVC only needs to be run once to obtain the embedding. This offline computing cost is of no importance – it is the online search speed that matters to end users. Once such an embedding is loaded into memory, the online calculation of Euclidean heuristics is very fast. Since we set $d = 4$ when SHE is not used and $d = 3$ when SHE is used, the training times for the MVC optimization in EH and EH+SHE are different. The same difference applies to GOEH and GOEH+SHE. All embeddings were computed on a desktop with two 8-core Intel(R) Xeon(R) processors at 2.67 GHz and 128GB of RAM. We implement MVC in MATLAB^{TM} and use CSDP (Borchers 1999) as the SDP solver. We parallelize each run of MVC on eight cores.

From the last two columns of Table 2, we can observe that the overhead for computing SHE is negligible compared to the training time for MVC optimization.

Conclusions and Future Work

In conclusion, we have introduced several substantial improvements to the Euclidean Heuristic (Rayner, Bowling, and Sturtevant 2011). We narrow the gap between heuristic estimates and true distances through two different means: 1) we optimize the Euclidean embedding to yield especially accurate distance estimates for the relevant goal states, and 2) we store the remaining approximation gaps in a compact fashion for online correction during search. The combination of these two enhancements yields drastic reductions in search space expansion.

As future work, we would like to investigate automatic EH re-optimization of problem domains with frequent A^* applications. If the solver keeps statistics of how often states are chosen as goals, it can periodically re-optimize the heuristics to best serve the users' demands. We will also study the generalization where we know a prior distribution of the goal state in the state space and design a new optimization objective to incorporate such probability distributions. Since there are known limits of improving heuristic accuracy alone (Helmert and Röger 2008), we will also study the interaction between GOEH heuristics and other orthogonal search-space reduction techniques such as (Chen and Yao 2009; Chen, Xu, and Yao 2009; Wolfe and Russell 2011; Wehrle and Helmert 2012; Alkhazraj et al. 2012).

We believe that, because of its simplicity, elegance and unmatched accuracy, the goal-oriented Euclidean heuristic, based on the large-scale MVC solver (Chen, Weinberger, and Chen 2013), will become a powerful standard heuristic for optimal search problems in many areas of AI.

Acknowledgements. WC and YC are supported in part by NSF grants CNS-1017701 and CCF-1215302. KQW is supported by NIH grant U01 1U01NS073457-01 and NSF grants 1149882 and 1137211.

References

[Alkhazraj et al. 2012] Alkhazraj, Y.; Wehrle, M.; Mattmuller, R.; and Helmert, M. 2012. A stubborn set algorithm for optimal planning. In *Proc. ECAI*.

[Borchers 1999] Borchers, B. 1999. Csdp, ac library for semidefinite programming. *Optimization Methods and Software* 11(1-4):613–623.

[Boyd and Vandenberghe 2004] Boyd, S., and Vandenberghe, L. 2004. *Convex Optimization*. New York, NY, USA: Cambridge University Press.

[Chen and Yao 2009] Chen, Y., and Yao, G. 2009. Completeness and Optimality Preserving Reduction for Planning. In *Proc. IJCAI*.

[Chen, Weinberger, and Chen 2013] Chen, W.; Weinberger, K.; and Chen, Y. 2013. Maximum variance correction with application to A^* search. In *Proc. ICML*.

[Chen, Xu, and Yao 2009] Chen, Y.; Xu, Y.; and Yao, G. 2009. Stratified Planning. In *Proc. IJCAI*.

[Geisberger et al. 2008] Geisberger, R.; Sanders, P.; Schultes, D.; and Delling, D. 2008. Contraction hierarchies: faster and simpler hierarchical routing in road networks. In *Proc. ICEA*, 319–333.

[Gupta and Nau 1992] Gupta, N., and Nau, D. 1992. On the complexity of blocks-world planning. *Artificial Intelligence* 56(2):223–254.

[Helmert and Röger 2008] Helmert, M., and Röger, G. 2008. How good is almost perfect? In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 2*, AAAI'08, 944–949. AAAI Press.

[Jones 2008] Jones, M. 2008. *Artificial Intelligence: A Systems Approach: A Systems Approach*. Jones & Bartlett Learning.

[Mero 1984] Mero, L. 1984. A heuristic search algorithm with modifiable estimate. *JAIR* 23:13–27.

[Ng and Zhang 2002] Ng, T., and Zhang, H. 2002. Predicting internet network distance with coordinates-based approaches. In *Proc. INFOCOM*, 170–179.

[Rayner, Bowling, and Sturtevant 2011] Rayner, C.; Bowling, M.; and Sturtevant, N. 2011. Euclidean Heuristic Optimization. In *Proc. AAAI*, 81–86.

[Sturtevant et al. 2009] Sturtevant, N. R.; Felner, A.; Barrer, M.; Schaeffer, J.; and Burch, N. 2009. Memory-based heuristics for explicit state spaces. In *Proc. IJCAI*, 609–614.

[Sturtevant 2007] Sturtevant, N. R. 2007. Memory-efficient abstractions for pathfinding. In Schaeffer, J., and Mateas, M., eds., *Proc. AIIDE*, 31–36.

[Tenenbaum, Silva, and Langford 2000] Tenenbaum, J. B.; Silva, V.; and Langford, J. C. 2000. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science* 290(5500):2319–2323.

[Wehrle and Helmert 2012] Wehrle, M., and Helmert, M. 2012. About partial order reduction in planning and computer aided verification. In *Proc. ICAPS*.

[Weinberger and Saul 2006] Weinberger, K., and Saul, L. 2006. Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision* 70:77–90.

[Weinberger et al. 2007] Weinberger, K.; Sha, F.; Zhu, Q.; and Saul, L. 2007. Graph laplacian regularization for large-scale semidefinite programming. In *Proc. NIPS*.

[Weinberger, Packer, and Saul 2005] Weinberger, K.; Packer, B.; and Saul, L. 2005. Nonlinear dimensionality reduction by semidefinite programming and kernel matrix factorization. In *Proc. Int'l Workshop on Artificial Intelligence and Statistics*.

[Wolfe and Russell 2011] Wolfe, J., and Russell, S. 2011. Bounded intention planning. In *Proc. IJCAI*.

[Zahavi et al. 2007] Zahavi, U.; Felner, A.; Schaeffer, J.; and Sturtevant, N. 2007. Inconsistent heuristics. In *Proc. AAAI*, 1211–1216.