Euclidean Heuristic Optimization

Chris Rayner and **Michael Bowling**
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2E8
{rayner,bowling}@cs.ualberta.ca

Nathan Sturtevant
Computer Science Department
University of Denver
Denver, CO 80208
sturtevant@cs.du.edu

Abstract

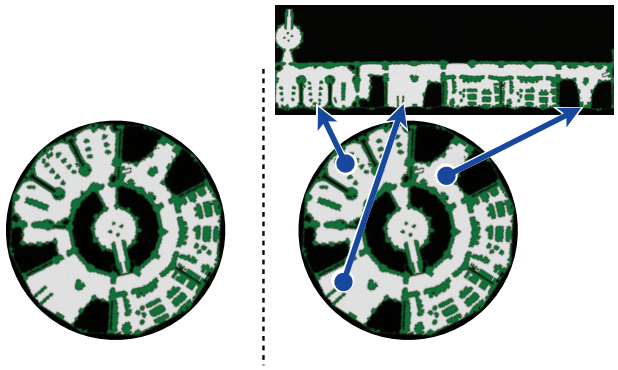
We pose the problem of constructing good search heuristics as an optimization problem: minimizing the loss between the true distances and the heuristic estimates subject to admissibility and consistency constraints. For a well-motivated choice of loss function, we show performing this optimization is tractable. In fact, it corresponds to a recently proposed method for dimensionality reduction. We prove this optimization is guaranteed to produce admissible and consistent heuristics, generalizes and gives insight into differential heuristics, and show experimentally that it produces strong heuristics on problems from three distinct search domains.

Introduction

An important problem in heuristic search is the selection of a strong heuristic function. This is especially true of many end-user applications of heuristic search, where high performance must be achieved under tight constraints on memory. Examples include virtual worlds and road networks on mobile GPS devices. Search domains such as these often specify an underlying geometry which can define a default heuristic. However, this underlying geometry–while small enough to keep in memory–may poorly represent the actual distances that must be traversed to move between two points.

We illustrate this with a map of one floor of a tower from the game *Dragon Age: Origins* in Figure 1. The original map, shown in Figure 1(a), has a wall at the top which prevents the walkable portions of the map from forming a complete loop. The default heuristic given by the underlying geometry is very poor, as two points at the top of the map that look close are actually quite far apart. Figure 1(b) demonstrates what would happen if the map were 'unrolled'. This unrolled map contains the same rooms and obstacles as the original, but rearranged in such a way that the default heuristic given by the underlying geometry is far more accurate. If this layout could be built automatically, search queries to this domain could be made faster and more efficient.

This motivates the following question: how can a graph be arranged in multidimensional Euclidean space so that the underlying Euclidean distance heuristic is more accurate, while still maintaining consistency and admissibility?



(a) (b)

Figure 1: This map from Dragon Age: Origins (a) is topologically closer to an open plane (b).

We formulate this as an optimization problem, which is a fundamentally different approach than traditional heuristic-building techniques. Furthermore, analysis of this approach reveals that a popular existing technique, differential heuristics, is equivalent to answering the same question when confined to a single dimension. In this paper we show how this optimization's general form can be solved efficiently, which allows a map or graph to be laid out in multiple dimensions.

This paper makes several layers of contributions to heuristic search. First, we introduce a new approach for building a heuristic in Euclidean space by solving a constrained optimization problem. We show that this optimization corresponds exactly to a recently proposed method for manifold learning in the field of dimensionality reduction (Weinberger and Saul 2006), drawing a previously unobserved but fundamental connection between these two subfields. Second, as the optimization provides information about the fundamental dimensionality of a search space, we can use our approach to make insights into the nature of heuristic construction. In particular, after showing that differential heuristics can be viewed as constructing a one dimensional Euclidean embedding, we observe that maps with a low fundamental dimensionality (e.g., the map in Figure 1(a) is nearly 1-dimensional) are particularly suited to differential heuristics. The inverse is also true – maps with higher intrinsic

dimensionality (such as 2, 3, or more dimensions) cannot be covered with a low number of differential heuristics, but we are able to find good multidimensional heuristics with our approach. Third, we demonstrate these insights empirically on graphs with low and high intrinsic dimensionality, showing competitive to significantly-improved search performance compared to the default and differential heuristics.

Related Work in Building Heuristics

One major area of research in heuristic search over the last fifteen years has been the use of memory to improve search. The two primary uses of memory have been to build heuristics for depth-first search algorithms (Culberson and Schaeffer 1998) and to enable large breadth-first searches (Zhou and Hansen 2004). Much of this work focuses on combinatorial puzzles, like the sliding tile puzzle, but researchers have also begun to use memory-based heuristics for planning. A common property of these domains is that they do not fit into main memory, so abstractions are used.

Only recently has it become important to quickly find routes through graphs that fit in memory. The two major applications of this research are computer games (Sturtevant 2007) and road networks (Geisberger et al. 2008). Heuristics for these domains fall into a broad class of true-distance heuristics (Sturtevant et al. 2009), where a subset of the shortest path distances in the graph are stored as the basis of heuristics. A variety of these true-distance methods have been developed, including differential heuristics (Ng and Zhang 2002; Goldberg and Harrelson 2005; Sturtevant et al. 2009) and portal heuristics (Goldenberg et al. 2010). Meanwhile, abstraction techniques such as contraction hierarchies (Geisberger et al. 2008) have also proved effective in these domains. The best techniques on road networks combine heuristics and contraction hierarchies (Bauer et al. 2008). In games, abstraction and refinement is most often used, but heuristics and contraction hierarchies have also been explored together (Sturtevant and Geisberger 2010).

One highly successful true-distance method is that of differential heuristics, which uses the triangle inequality to provide an admissible bound on distances in the search space. Suppose there is a pivot state s from which the distances to all other states are known. Then the distance between any two states a and b must be at least $|\delta(a, s) - \delta(b, s)|$, where $\delta(i, j)$ is the shortest path distance between states i and j. When multiple pivot states are available, the greatest distance over all available pivots is used. Later, we will show that a differential heuristic is equivalent to a one dimensional embedding, and is a special case of our approach to finding (possibly) multidimensional Euclidean heuristics.

Euclidean Heuristic Optimization

Many of the heuristics described in the previous section are computed by simply performing a breadth-first search through a domain (or an abstract domain), and caching the results for use as a heuristic. More complicated approaches exist, but heuristic building is generally formulated as a search problem. In this work, heuristic generation is uniquely formulated as an optimization problem.

Euclidean heuristics are heuristic values that can be computed as distances in some Euclidean space of d dimensions. That is, each state i in a search graph of size n is represented as a vector $y_i \in \mathbb{R}^d$, and the heuristic value between two states is $h(i, j) = \|y_i - y_j\|$, the Euclidean distance between the vectors. Let Y be an n by d matrix whose rows are these vectors; Y, then, implicitly encodes the heuristic function.

General Approach

By manipulating the location of these points, we propose to construct a consistent and admissible Euclidean heuristic as the solution to an optimization problem. In its general form:

$$\underset{Y}{\textbf{minimize}} \quad \mathcal{L}(Y) \tag{1}$$

$$\textbf{subject to} \quad Y \text{ is admissible and consistent}$$

where $\mathcal{L}$ is a loss function which measures the error in the heuristic values in Y compared to the true shortest path distances. This optimization problem can be thought of as looking among all admissible and consistent Euclidean heuristics, as encoded by Y, for the one with lowest error. We call such a Y an *optimal Euclidean heuristic*, and will examine the constraints and objective of this optimization in turn.

Constraints. The constraints on the admissibility and the consistency of the heuristic encoded by Y can be respectively formalized as the following:

$$\forall i, j \quad \|y_i - y_j\| \leq \delta(i, j) \tag{2}$$

$$\forall i, j, k \quad \|y_i - y_j\| \leq \delta(i, k) + \|y_j - y_k\| \tag{3}$$

where $\delta(i, j)$ is the true shortest path distance between i and j. These constraints apply to all combinations of states in the search space. However, we can drastically simplify them by observing that a subset of them satisfies all of them. We show this here, but also note it has also been proved in earlier work on building heuristics (Passino and Antsaklis 1994):

Theorem 1 *A Euclidean heuristic is admissible and consistent if and only if the heuristic is* locally admissible*, i.e., $\forall (i, j) \in E \; \|y_i - y_j\| \leq \delta(i, j)$ where E is the set of edges.*

Proof. The backward implication is trivially true as local admissibility conditions are a subset of global admissibility conditions. The forward implication involves repeated application of the triangle inequality. For states i, j, k, let $p_1, p_2, ..., p_\ell$ be the vertices on a shortest path from i to j with $(p_l, p_{l+1}) \in E$:

$$\|y_i - y_j\| = \|p_1 - p_\ell\| \leq \textstyle\sum_{l=1}^{\ell-1} \|p_l - p_{l+1}\| \tag{4}$$

$$\leq \textstyle\sum_{l=1}^{\ell-1} \delta(l, l+1) = \delta(i, j) \tag{5}$$

and

$$\|y_i - y_j\| \leq \|y_i - y_k\| + \|y_j - y_k\| \tag{6}$$

$$\leq \delta(i, k) + \|y_j - y_k\|, \tag{7}$$

where line 4 (repeatedly) and 6 are applications of the triangle inequality; line 5 is the local admissibility condition; and line 7 comes from admissibility which was just proved. Thus proving admissibility and consistency. □

Therefore, when solving the optimization problem on line 1, it is only necessary to require constraints on the local admissibility of Y. This requires just one constraint per edge in the search graph, which greatly simplifies the optimization.

Objective. The loss function $\mathcal{L}$ combines the errors between the heuristic distances and the true distances into a single scalar value. It specifies the relative importance of the heuristic between each pair of states, as well as a trade-off between many small errors versus a single large error. We propose the following loss on Y:

$$\mathcal{L}(Y) = \sum_{i,j} W_{ij} \left|\delta(i,j)^2 - \|y_i - y_j\|^2\right| \quad (8)$$

Each entry in the weight matrix W specifies the importance of the heuristic between two states. For example, if it is important that the heuristic value between two states be accurate, the corresponding entries in W can be set to large values. The squaring of terms emphasizes heuristic errors on longer paths over the same magnitude errors on shorter paths,[1] and will prove to be computationally convenient since it induces a convex objective function.

In the context of the optimization described on line 1, this loss can be greatly simplified. Since Y must be admissible, the square of the optimal distance must be at least as great as the squared heuristic. It is therefore unnecessary to take the absolute value, and the loss can be broken up as follows:

$$\mathcal{L}(Y) = \sum_{i,j} W_{ij} \left(\delta(i,j)^2 - \|y_i - y_j\|^2\right) \quad (9)$$

$$= \sum_{i,j} W_{ij}\, \delta(i,j)^2 - \sum_{i,j} W_{ij} \|y_i - y_j\|^2 \quad (10)$$

Since the first term of Equation 10 does not depend on Y, minimizing $\mathcal{L}(Y)$ is equivalent to *maximizing* the weighted sum of squared distances between the points in Y. Altogether, we arrive at a specific optimization problem:

$$\underset{Y}{\textbf{maximize}} \quad \sum_{i,j} W_{ij} \|y_i - y_j\|^2 \quad (11)$$

$$\textbf{subject to} \quad \forall (i,j) \in E \quad \|y_i - y_j\| \leq \delta(i,j)$$

Unfortunately, the objective function on line 11 is not convex and so is not easy to solve efficiently. However, a change of variables allows us to rewrite the optimization in terms of a kernel matrix, which results in a convex objective. The next section goes into the technical detail of how to accomplish this, however the reformulated optimization is identical to the one above, and although necessary to make the problem tractable, it adds little to the understanding of the problem.

Semidefinite Formulation

Let $K = YY^T$ be the matrix of inner products between the vector representations of the states, so $K_{ij} = y_i \cdot y_j = y_i^T y_j$. We note that squared distances can be directly expressed in terms of the entries of this kernel matrix:

$$\|y_i - y_j\|^2 = (y_i - y_j)^T (y_i - y_j) \quad (12)$$

$$= y_i^T y_i - 2 y_i^T y_j + y_j^T y_j \quad (13)$$

$$= K_{ii} - 2K_{ij} + K_{jj} \quad (14)$$

Since the optimization on line 11 only refers to Y through distances $\|y_i - y_j\|$, we can then rewrite it in terms of K:

$$\underset{K}{\textbf{maximize}} \quad \sum_{i,j} W_{ij}(K_{ii} - 2K_{ij} + K_{jj}) \quad (15)$$

$$\textbf{subject to} \quad \forall (i,j) \in E \quad K_{ii} - 2K_{ij} + K_{jj} \leq \delta(i,j)^2$$
$$K \succeq 0$$

[1] For example, a heuristic distance of 9 when the true distance is 10 will be penalized by a significantly larger amount than a heuristic distance of 1 when the true distance is 2.

Since both sides of the inequality in each local admissibility constraint are positive, an equivalent constraint has instead been imposed on the squared distances (heuristic and true). Also, note a positive semidefinite constraint $K \succeq 0$ was added. This importantly ensures that there exists some Y such that $K = YY^T$, or in other words that K represents inner products in some Euclidean space.

Since the optimization as stated is indifferent to translations of the points in Y, we force a unique solution by adding a "centering" constraint. We define this as $\sum_{i,j} W_{ij} K_{ij} = 0$ which also further simplifies the objective. Let Δ_W be a diagonal matrix with the sums of the rows of W on the diagonal. We arrive at the following optimization:

$$\underset{K}{\textbf{maximize}} \quad \text{Tr}(K\Delta_W) \quad (16)$$

$$\textbf{subject to} \quad \forall (i,j) \in E \quad K_{ii} - 2K_{ij} + K_{jj} \leq \delta(i,j)^2$$
$$\textstyle\sum_{i,j} W_{ij} K_{ij} = 0 \qquad K \succeq 0$$

This rewritten optimization involves a linear objective and linear constraints, plus a non-linear, but *convex*, semidefinite constraint. Such semidefinite programs can be solved using a standard toolbox solver such as SDPT3 (Toh, Todd, and Tutuncu 1999).

Extracting Y. The optimization on line 16 produces a kernel matrix K, but we want the Euclidean vectors in Y. This entails kernel principal component analysis, which involves centering the kernel to get $K' = (I - \frac{1}{n}\mathbf{1}\mathbf{1}^T)K(I - \frac{1}{n}\mathbf{1}\mathbf{1}^T)$ and then doing an eigendecomposition of K'. The resulting eigenvectors scaled by their eigenvalues correspond to the columns of Y. The number of non-zero columns may be prohibitively large (n in the worst case), but the optimization tends to keep this number small.

Under precise memory constraints, a reasonable way to create a representation with d values per state is to choose the d eigenvectors with the largest associated eigenvalues (the top *principal components*). This is because eigenvectors with small associated eigenvalues have small effects on the resulting heuristic estimates, so less is lost by ignoring them.

Maximum Variance Unfolding

This exact optimization problem is not new. It is a weighted generalization of maximum variance unfolding (MVU) (Weinberger and Saul 2006) which was originally introduced for non-linear dimensionality reduction. True to its name, the optimization has a nice visual interpretation. Imagine neighboring states in the search graph being connected by rods. A rod can pass through other rods, it can be compressed, or it can grow to a length no greater than the edge cost. Once these rods are all in place, the structure is pulled apart as far as the rods allow. The result is an embedding of points in Euclidean space whose squared interpoint distances are maximized, thus maximizing the variance. The observation that maximizing variance often results in low-dimensional embeddings (i.e., small d) corresponds to its proposed use in dimensionality reduction. As noted, this is a fortunate side-effect for our application.

While MVU has been applied in a number of dimensionality reduction applications, such as visualization (Weinberger and Saul 2006), sensor networks (Weinberger et

al. 2006), and mapping (Bowling, Wilkinson, and Ghodsi 2006), its connection to identifying admissible and consistent heuristics in search has not been observed previously.

Analysis

Heuristic admissibility and consistency are vital to the guaranteed solution optimality of many search algorithms. Here we prove these hold for optimal Euclidean heuristics. We then show that differential heuristics are a special case of our framework under a specific choice of weight matrix.

Admissibility and Consistency

While the optimization is constrained to produce admissible and consistent heuristics, the complete procedure took a subset of d principal components We show this reduction in dimensionality preserves admissibility and consistency.

Theorem 2 *If Y is the first d principal components of K, as per line 16, then Y is admissible and consistent.*

Proof. According to Theorem 1, we only need to show that Y is locally admissible. Let $\tilde{Y}$ contain all n principal components of K, so $K = \tilde{Y}\tilde{Y}^T$. Since Y is the first d principal components, we can represent $\tilde{Y}$ as $[Y\ Y']$ for some Y' which contains the remaining principal components. Then:

$$h(i,j)^2 = \|y_i - y_j\|^2 = (y_i - y_j)^T(y_i - y_j) \quad (17)$$
$$\leq (y_i - y_j)^T(y_i - y_j) + (y'_i - y'_j)^T(y'_i - y'_j) \quad (18)$$
$$= (\tilde{y}_i - \tilde{y}_j)^T(\tilde{y}_i - \tilde{y}_j) = K_{ii} - 2K_{ij} + K_{jj} \quad (19)$$

Line 18 holds since the dot-product of real-valued vectors is always non-negative. The constraints on K on line 16 guarantee for $(i,j) \in E$ that $K_{ii} - 2K_{ij} + K_{jj} \leq \delta(i,j)^2$. Thus for $(i,j) \in E$ we have that $h(i,j)^2 \leq \delta(i,j)^2$, so $h(i,j) \leq \delta(i,j)$ since both are non-negative. □

Generalizing Differential Heuristics

Recall that differential heuristics are constructed by measuring the optimal path length from a pivot state to all other states, then applying the triangle inequality on these measurements to get a heuristic for the distance between any two states. This heuristic is well thought of as a one dimensional embedding: the pivot is embedded at the origin, and every other state is embedded at its shortest path distance from the pivot. In fact, differential heuristics are optimal Euclidean heuristics for a carefully chosen weight matrix W_{diff} and thus are a special case of our general framework.

Without loss of generality assume the index 1 refers to the pivot. Let W_{diff} be the weight matrix with zeros on the diagonal, ones on the non-diagonal entries of the first row and column, and $\frac{-1}{n-1}$ elsewhere. For example, for $n = 4$:

$$W_{\text{diff}} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & -1/3 & -1/3 \\ 1 & -1/3 & 0 & -1/3 \\ 1 & -1/3 & -1/3 & 0 \end{bmatrix}$$

Theorem 3 *The optimization on line 16 with weight matrix W_{diff} results in a single principal component (K will be rank one); the resulting heuristics are $h(i,j) = |\,\delta(i,1) - \delta(j,1)|$.*

Proof. (sketch) The proof is rather involved, but shows that if K is not rank one we can find an alternative embedding that strictly improves the objective. Further, for any rank one K, unless $y_i = \delta(i,1)$ the objective can be improved. □

Thus, differential heuristics are one example of optimal Euclidean heuristics for a particular choice of weight matrix.

The design of W_{diff} shows that differential heuristics optimize a strange objective. The negative weights between non-pivot states drive these states closer together. That is, differential heuristics are in some sense designed to give poor heuristic values between the non-pivot states (albeit with small weight). Intuitively, good heuristics should use non-negative weights so all distances decrease the loss. This suggests the differential heuristic notion might be improved upon, in some domains, by solving for the optimal Euclidean heuristic using W^+_{diff} where the negative entries are replaced with small positive values. We explore this issue empirically.

Complexity of Optimization

The semidefinite program scales linearly with the number of points, but has a time complexity in $O(N^3)$ where N is the number of constraints and space complexity in $O(n^2)$ to store instantiations of the kernel matrix $K \in \mathbb{R}^{n \times n}$, where n is the number of states. Recall that in our formulation one constraint is required per edge in the search graph. In practical terms, performing the optimization on a search graph with thousands of states with octile connectivity requires several hours of computation. Techniques for scaling it to significantly larger problems have been proposed in the application to dimensionality reduction (Weinberger et al. 2006, for example), but will not be examined in this paper.

Experiments

We consider three experimental domains exhibiting different dimensionality. In each, we measure the performance of the A* search algorithm (Hart, Nilsson, and Raphael 1968) with different generated heuristics. Our performance metric is an average count of the number of nodes A* expands, bucketed by solution length. Note that drastically more computation is needed for longer paths than shorter ones, and so performance on long paths is a key detail in these results.

The inset bar plots accompanying each figure illustrate the notion of intrinsic dimensionality for each domain (or a sample average for the game maps). They show how much statistical variance each of the first five dimensions of the embedding contributes to the heuristic values when the objective is posed with uniform weights.

Cube World. The cube world is a synthetic domain, generalizing the octile grid-world to three dimensions. A large cube whose sides measure 20 units each is divided into 8,000 unit cubes which may be transitioned between if they share a vertex. The edge costs between cubes are the distances between their centers. A perfect heuristic for this simple domain is obvious by its construction, but for the purpose of demonstration we examine building heuristics from scratch.

The differential heuristics have pivots in the four corner states on the bottom half of the large cube. Meanwhile the

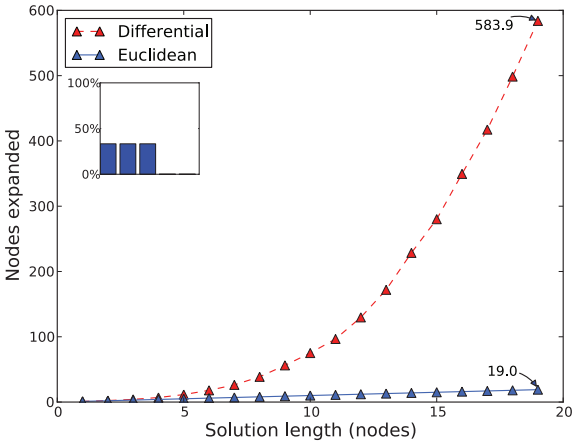


Figure 2: Cube world results, averaged over 10,000 random problems. Inset plot shows equally contributing dimensions. Euclidean heuristics excel, while also using less memory.

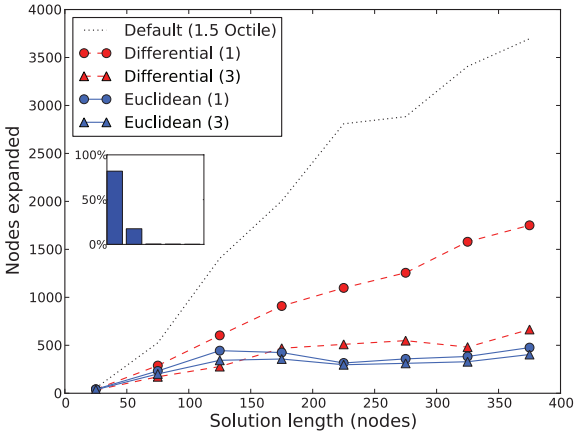


Figure 3: *Dragon Age: Origins* results on standard problem sets. Inset plot reveals low intrinsic dimensionality: the first dimension of each embedding supports most of the variance.

embedding from a uniformly weighted optimal Euclidean heuristic is simply the positions of the centers of the unit cubes. The performances of these heuristics are compared in Figure 2. This search space is intrinsically multidimensional, and a heuristic based on the three-dimensional embedding significantly, although not surprisingly, outperforms the combined heuristics from one dimensional embeddings (i.e., differential heuristics) while using less memory.

Game Maps. The next domain is a test suite of path planning problems from 61 computer game maps containing 168 to 6,240 states. The maps come from Bioware's *Dragon Age: Origins* and the path planning problems are typical to many computer games. Cardinal moves have unit cost, and diagonal moves cost 1.5, enabling heuristic values to be rounded up to the nearest half. Some transitions are blocked by obstacles, and cutting corners (moving diagonally between states which are not in a 4-clique) is disallowed to mimic spatial restrictions on an agent with volume.

In contrast to the cube world, these game maps have low intrinsic dimensionality – they often feature long 'corridors' such as the one shown in Figure 1. As a result, most paths only require good heuristics on states in corridors, for which a one dimensional differential heuristic combined with the default heuristic suffices. Similarly, the multidimensional embeddings found by applying our method tend to have only one descriptive dimension corresponding to the longest corridor. Storage of the less descriptive dimensions is wasteful compared to embedding another (distinct) corridor.

With these considerations in mind, we turn to the weight matrix W. The best way to define W is an open question, but we take a first step by showing that it can be designed to augment the differential heuristics defined by an effective pivot layout. The first pivot is placed in a state farthest from a randomly chosen seed state. Each subsequent pivot is placed in a state most distant from the previous pivots. The nth Euclidean heuristic uses weights W^+_{diff} based on the nth pivot, where non-pivot entries in W^+_{diff} are set to 10^{-3}. From each embedding only the most descriptive dimension is kept, which tends to capture most of the available variance.

Figure 3 shows results from the Hierarchical Open Graph (HOG) search platform for five groups of heuristics: the default "octile" heuristic (where diagonal moves cost 1.5), *Differential (1)* and *(3)* (sets of one and three differential heuristics, respectively) and *Euclidean (1)* and *(3)* (sets of one and three one dimensional optimal Euclidean heuristics, respectively). The sets of heuristics in each group are combined with each other and the default heuristic by taking the maximum over all available heuristic values. We see that optimal Euclidean heuristics based on W^+_{diff} are an improvement (albeit small) over differential heuristics, which optimize using the weights W_{diff}. It is prudent to note here that differential heuristics can be computed more quickly and scale to much larger problems. Yet, we have shown that the *concept* of differential heuristics can be improved upon as suggested by our optimization interpretation of their construction.

Word Search. This domain's states represent four-letter words in English. The goal is to change a start word into a goal word by changing one letter at a time. For example, there is an edge between the words 'fore' and 'fork', but no edge between 'fore' and 'pork'. All transitions are unit cost, enabling us to take the ceiling of any heuristic value. We used the largest connected subgraph of words as the search space, which is densely connected and has 4,820 states.

To construct differential heuristics, we use the same pivot layout discussed earlier to situate sets of 6 and 18 pivots. But here we optimize a multidimensional Euclidean heuristic under a *uniform* weight matrix W, and consider the top 6 and 18 dimensions of this single embedding. The performances of these heuristics are compared in Figure 4. Note that differential heuristics actually excel on the longest problems. This is because over half of these longest problems (of which there are roughly 180) start or end on the state for the word 'upas', which is often chosen as a pivot point. However, across the vast majority of paths, the multidimensional Euclidean heuristic offers a substantial improvement over combining one dimensional differential heuristics.

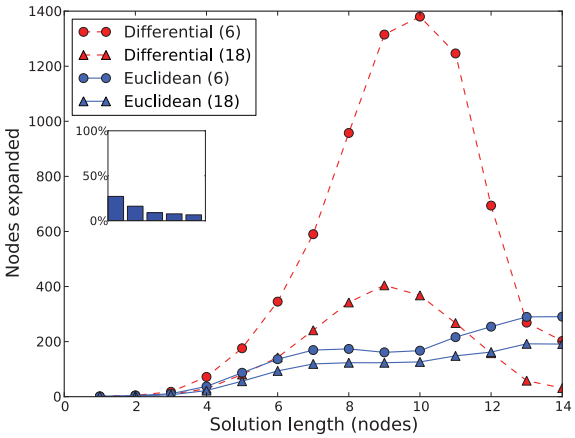


Figure 4: Word search results, averaged over 10,000 random problems. Inset plot shows variance over many dimensions.

Conclusion

This paper introduced a novel approach to constructing admissible and consistent heuristics as the solution to a constrained optimization problem – one that has already been studied in the field of dimensionality reduction. The approach generalizes differential heuristics beyond a single dimension. Furthermore it enabled a number of insights, revealing a bias in the implied objective of differential heuristics, and showing how the success of differential heuristics is tied to the intrinsic dimensionality of the search space as recovered by our proposed optimization.

The observed connection between heuristic search and dimensionality reduction appears highly profitable. Heuristic search is a natural application for manifold learning techniques of which MVU is just one. The work in this paper can be extended in a number of directions: scaling to larger problems (Weinberger et al. 2006), fundamentally different objective functions, or away from Euclidean metrics and into non-Euclidean geometries (Walter 2004). We have only touched the surface of what could grow into a highly beneficial relationship between these two subfields.

Acknowledgments

This research was supported by iCORE and NSERC. Thanks to A. Felner and anonymous reviewers for valued feedback.

References

Bauer, R.; Delling, D.; Sanders, P.; Schieferdecker, D.; Schultes, D.; and Wagner, D. 2008. Combining Hierarchical and Goal-Directed Speed-Up Techniques for Dijkstra's Algorithm. In *Proceedings of the 7th Workshop on Experimental Algorithms*, volume 5038 of *Lecture Notes in Computer Science*, 303–318. Springer.

Bowling, M.; Wilkinson, D.; and Ghodsi, A. 2006. Subjective Mapping. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, 1569–1572.

Culberson, J., and Schaeffer, J. 1998. Pattern Databases. *Computational Intelligence* 14(3):318–334.

Geisberger, R.; Sanders, P.; Schultes, D.; and Delling, D. 2008. Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks. In *Proceedings of the 7th Workshop on Experimental Algorithms*, volume 5038 of *Lecture Notes in Computer Science*, 319–333. Springer.

Goldberg, A. V., and Harrelson, C. 2005. Computing the Shortest Path: A* Search Meets Graph Theory. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 156–165.

Goldenberg, M.; Felner, A.; Sturtevant, N.; and Schaeffer, J. 2010. Portal-Based True-Distance Heuristics for Path Finding. In *Proceedings of the 3rd Annual Symposium on Combinatorial Search (SoCS)*.

Hart, P.; Nilsson, N.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics* 4(2):100–107.

Ng, T. S. E., and Zhang, H. 2002. Predicting Internet Network Distance with Coordinates-Based Approaches. In *IEEE International Conference on Computer Communications (INFOCOM)*, 170–179.

Passino, K. M., and Antsaklis, P. J. 1994. A Metric Space Approach to the Specification of the Heuristic Function for the A Algorithm. *IEEE Transactions on Systems, Man, and Cybernetics* 24:159–166.

Sturtevant, N., and Geisberger, R. 2010. A Comparison of High-Level Approaches for Speeding up Pathfinding. In *Proceedings of the 4th Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, 76–82.

Sturtevant, N.; Felner, A.; Barrer, M.; Schaeffer, J.; and Burch, N. 2009. Memory-Based Heuristics for Explicit State Spaces. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, 609–614.

Sturtevant, N. 2007. Memory-Efficient Abstractions for Pathfinding. In *Proceedings of the 3rd Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, 31–36.

Toh, K. C.; Todd, M. J.; and Tutuncu, R. H. 1999. SDPT3 – a Matlab software package for semidefinite programming. *Optimization Methods and Software* 11:545–581.

Walter, J. A. 2004. H-MDS: A New Approach for Interactive Visualization with Multidimensional Scaling in the Hyperbolic Space. *Information Systems* 29(4):273–292.

Weinberger, K. Q., and Saul, L. K. 2006. An Introduction to Nonlinear Dimensionality Reduction by Maximum Variance Unfolding. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*.

Weinberger, K. Q.; Sha, F.; Zhu, Q.; and Saul, L. K. 2006. Graph Laplacian Regularization for Large-Scale Semidefinite Programming. In *Advances in Neural Information Processing Systems 19*, 1489–1496.

Zhou, R., and Hansen, E. 2004. Structured Duplicate Detection in External-Memory Graph Search. In *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI)*, 683–689.