An Optimal Any-Angle Pathfinding Algorithm

Daniel Harabor and **Alban Grastien**
NICTA and The Australian National University
Email: firstname.lastname@nicta.com.au

Abstract

Any-angle pathfinding is a common problem from robotics and computer games: it requires finding a Euclidean shortest path between a pair of points in a grid map. Prior research has focused on approximate online solutions. A number of exact methods exist but they all require supra-linear space and preprocessing time. In this paper we describe Anya: a new optimal any-angle pathfinding algorithm which searches over sets of states represented as intervals. Each interval is identified online. From each we select a representative point to derive a corresponding f-value for the set. Anya always returns an optimal path. Moreover it does so entirely online, without any preprocessing or memory overheads. This result answers an open question from the areas of Artificial Intelligence and Game Development: is there an any-angle pathfinding algorithm which is online and optimal? The answer is yes.

Introduction

Any-angle pathfinding is a navigation problem which appears in robotics and computer video games. It involves finding a shortest path between an arbitrary pair of points on a two-dimensional grid map but asks that movement along the path is not artificially constrained to the points of the grid. Within the game development community a simple and popular solution exists known as *string pulling* (Pinter 2001; Botea, Müller, and Schaeffer 2004). The idea is to compute a grid-optimal path in the first instance and smooth the result as part of a post-processing step that improves both its length and aesthetic appeal. String pulling has two disadvantages: (i) it requires more computation than just finding a path (ii) it only yields approximately shortest paths.

A number of algorithms improve on string-pulling by integrating post-processing into node-expansion during search. Field D* (Ferguson and Stentz 2005) uses linear interpolation to smooth paths one grid cell at a time. Theta* (Nash et al. 2007) introduces a shortcut each time a successful line-of-sight check is made from the parent of the current node to any of its successors. Meanwhile, Block A* (Yap et al. 2011) employs during search a pre-computed database of optimal Euclidean distances between pairs of points in a localised area. Each of these approaches improves on string pulling in terms of solution quality and, in many cases, running time. Unfortunately none are optimal. Accelerated A* (Šišlák, Volf, and Pěchouček 2009) is an any-angle algorithm that is conjectured to be optimal but for which no strong theoretical argument is made. Similar to Theta*, it differs primarily in that line-of-sight checks are performed from a set of expanded nodes rather than a single ancestor. The size of the set is only loosely bounded and, for challenging problems, can include a large proportion of nodes on the Closed List.

Visibility Graphs (Lozano-Pérez and Wesley 1979) and Tangent Graphs (Liu and Arimoto 1992) are optimal techniques that can solve a generalised form of the any-angle pathfinding problem. Their primary disadvantage is that each such graph requires quadratic space in the worst case and must be computed offline. Other exact approaches are based on the Continuous Dijkstra (Mitchell, Mount, and Papadimitriou 1987) paradigm. The most efficient of these algorithms (Hershberger and Suri 1999) pre-computes a planar subdivision of the map that can be used to extract a path in just logarithmic time. Unfortunately the precomputation assumes the starting location does not change.

In this work we introduce a new approach to any-angle pathfinding which addresses many of the shortcomings associated with existing research. Our method, Anya, bears some similarity with Continuous Dijkstra: instead of searching over the individual nodes of the grid we search over contiguous sets of states that form intervals. Each interval has a representative point used to derive an f-value and each is projected from one row of the grid onto another until the goal is reached. Anya does not rely on any precomputation, does not introduce any memory overheads (beyond what is required by e.g. A*) and always finds an optimal any-angle path.

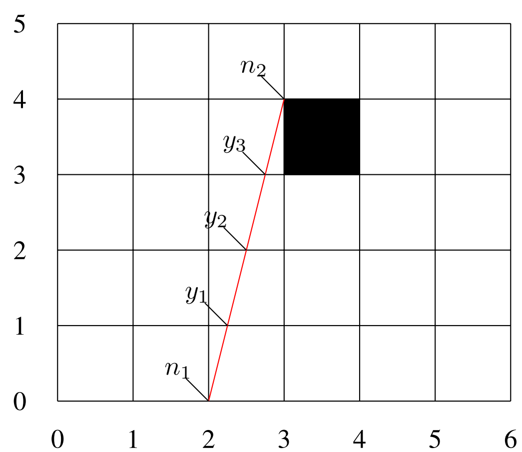

Figure 1: When pathfinding from n_1 to n_2 online algorithms such as Theta* only consider the discrete points of the grid and never any points y_i.

Preliminaries

A *grid* is a planar subdivision consisting of $W \times H$ square cells. Each cell is an open set of *interior* points which are all *traversable* or all *non-traversable*. The vertices associated with each cell are called the *discrete* points of the grid. Edges in the grid can be interpreted as open intervals of *intermediate* points; each one representing a transition between two discrete points. Each type of point $p = (x, y)$ has a unique coordinate where $x \in [0, W]$ and $y = [0, H]$. In the case of discrete points both x and y are limited to the subset of integer values from the respective range.

A discrete or intermediate point is traversable if it is adjacent to at least one traversable cell. Otherwise it is non-traversable. A discrete point which is common to exactly four adjacent cells is called an *intersection*. Any intersection where three of the adjacent cells are traversable and one is not is called a *corner*. Two points are *visible* from one another if there exists a straight-line path connecting them that does not: (i) pass through any non-traversable point or (ii) pass between two diagonally-adjacent non-traversable cells.

An *any-angle path* π is a sequence of points $\langle p_1, \ldots, p_k \rangle$ where each p_i is visible from p_{i-1} and p_{i+1}. The *length* of π is the cumulative distance between every successive pair of points $d(p_1, p_2) + \ldots + d(p_{k-1}, p_k)$, where $d((x, y), (x', y')) = \sqrt{(x - x')^2 + (y - y')^2}$ is a uniform Euclidean distance metric. We will say $p_i \in \pi$ is a *turning point* if the segments (p_{i-1}, p_i) and (p_i, p_{i+1}) form an angle not equal to $180°$. It is well-known that the turning points in optimal any-angle paths are corner points; see e.g. (Mitchell, Mount, and Papadimitriou 1987).

Principle of Anya

Consider the any-angle instance depicted in Figure 1. The start point is $n_1 = (2, 0)$ and the target point is $n_2 = (3, 4)$. A popular online algorithm[1] for solving such problems is Theta* (Nash et al. 2007). This method computes an any-angle path by only considering the set of discrete points from the grid. Each time such a point is reached Theta* "pulls the string". Thus when node n_2 is generated its g-value is not the length of the grid-constrained path from n_1 to n_2 but rather the length of the direct path $\langle n_1, n_2 \rangle$.

The problem with this approach is that the solution-cost estimate (or f-value), from a parent node to each of its successors, may not be monotonically increasing. The monotone condition is necessary to guarantee that an optimal solution, if one exists, is always found. For instance: Theta* can generate n_2 from the intermediate point $p = (3, 3)$. When p is expanded we have $f(p) = d(n_1, p) + h(p, n_2) = 4.16$. To satisfy the monotone condition we require that $f(n_2) \geq 4.16$. However Theta* computes $f(n_2) = d(n_1, n_2) + h(n_2, n_2) = 4.12$. Clearly p should be expanded after n_2 but in this case the opposite occurs.

In order to avoid this mistake we would need to consider, in addition to the set of discrete points from the grid, all the points y_i shown in Figure 1. The problem is that the number of such points can be very large: each edge of the grid, together with its discrete endpoints, forms a $[0, 1]$ interval that can be intersected by the optimal path at any point $0 \leq \frac{w}{h} \leq 1$; here w (resp. h) is an integer in $\{0, \ldots, W\}$ (resp. $\{0, \ldots, H\}$). This is a set whose members are reducible to a Farey Sequence. For any given n (in our case $n = \max(W, H)$) the cardinality of the corresponding set of elements is known to be quadratic in n (Graham, Knuth, and Patashnik 1989)(Ch. 9). We are therefore motivated to consider an alternative approach: instead of evaluating each y_i node individually we will evaluate together all the nodes from the corresponding interval in which each y_i appears.

Algorithm

Definition 1 *A* grid interval I *is a set of contiguous pairwise visible points from any row of the grid. Each interval is defined in terms of its endpoints* a *and* b*. With the possible exception of* a *and* b*, each interval contains only intermediate and discrete non-corner points.*

Identifying intervals is simple: any row of the grid can be naturally divided into maximally contiguous sets of traversable and non-traversable points. Each traversable set forms a tentative interval which we split, repeatedly, until the only corner points are found at a or b.

A significant advantage of Anya is that we construct intervals on-the-fly. This allows us to start answering queries immediately and for any discrete start-target

[1]By online we mean that no preprocessing is required.

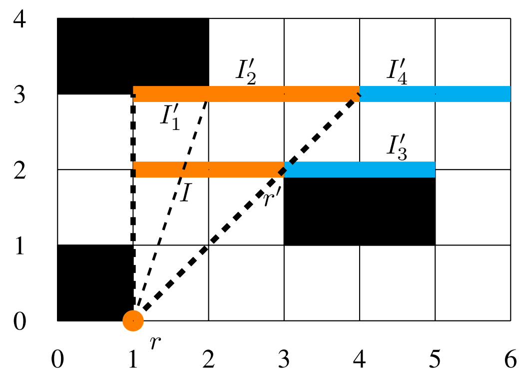

Figure 2: (I, r) has four successors: (I'_1, r) and (I'_2, r) which are observable and (I'_3, r') and (I'_4, r') which are not. Notice that intervals of traversable points exist left of I but the local path through I to each such point is not taut.

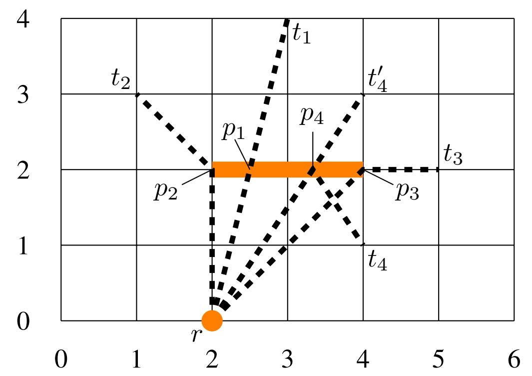

Figure 3: An illustration of Lemmas 1 and 2. The points t_1 and t'_4 correspond to the case where the line intersects the interval; t_2 and t_3 where it does not; t_4 where the mirrored target t'_4 must be used.

pair. Similar algorithms (e.g. Continuous Dijkstra) require a preprocessing step before any queries can be answered and then only from a single fixed start point.

Definition 2 *A search node (I, r) is a tuple where I is an interval and $r \notin I$ is a* root *point chosen s.t. each $p \in I$ is visible from r. The identity of r is always the most recent turning point on a path from the start point s to any $p \in I$. To represent the start node, set $I = [s]$ and assume r is a point located off the plane and visible only from s. The cost from r to s in this case is zero.*

The successors of a search node n are identified by computing intervals over sets of traversable points from the same row of the grid as n and from rows immediately adjacent. We want to guarantee that each point in such a set can be reached from the root of n via a local path which is *taut*. Taut simply means that if we "pull" on the endpoints of the path we cannot make it any shorter.

Definition 3 *(I', r') is a* successor *of (I, r) if each $p' \in I'$ is reached by a taut path $\langle r, p, p' \rangle$ that begins at r and passes through some $p \in I$, and r' is the last common point of these paths. Additionally, the subpath $\langle p, p' \rangle$ must not intersect any interval $J \neq I'$.*

We begin with the set of traversable points that are visible from r through I and divide this set into $0 \leq k$ adjacent closed grid intervals. We will say that each such interval is *observable* and generate for each a corresponding successor node (I', r') with root $r' = r$.

Not all successors are observable. For example, the taut path from r can intersect I at an endpoint b which is also a corner point. In this case we reach a set of traversable points that are either adjacent to I or adjacent to the set of observable successors. Each such point is visible from $p = b$ but not from r. From this set of non-visible points we build a single half-open interval $I' = [a', b')$ s.t. I' is open at the endpoint closest to b. We will say I' is *non-observable* and generate a corresponding successor (I', r') with root $r' = b$. Figure 2 shows examples of both observable and non-observable successors. To evaluate a search node $n = (I, r)$ we select a point $p \in I$ which has a minimum f-value with respect to a target point p. We compute:

$$f(p) = g(r) + d(r, p) + h(p, t) \tag{1}$$

where $g(r)$ is the length of the optimal path from the start point to the root, $d(r, p)$ is the straight line distance from r to p and $h(p, t)$ is an admissible heuristic function that lower-bounds the cost of reaching t from p.

Finding this point p is not trivial in general. For certain heuristics however, it is easy. Assume for instance that the heuristics is the straight-line distance between p and t (ignoring obstacles): $h(p, t) = d(p, t)$. The point p can be identified thanks to the following two lemmas.

Lemma 1 *Let t and r be two points and let I be an interval s.t. the row of I is between the rows of t and r. Then the point p of I with minimum f-value is point in I closest to the intersection of (t, r) with the row of I.*

Proof: If (t, r) intersects I in p_{i}, then the minimum value of $d(r, p) + h(p, t)$ is $d(r, t)$ which is obtained by choosing $p = p_{\mathrm{i}}$ (by the triangle inequality). Otherwise choose p as the end point of I on the side where (r, t) intersects the row of I. □

If the precondition of Lemma 1 is not satisfied, it is possible to replace t by its mirrored version t' through I which does satisfy the precondition.

Lemma 2 *The mirrored point t' of target t through interval I is such that $d(p, t) = d(p, t')$ for all $p \in I$.*

Lemma 2 is a trivial geometrical result. Both lemmas are illustrated on Figure 3.

The algorithm terminates when we expand a node (I, r) s.t. $t \in I$. By Lemma 1 and 2 the f-value of this interval is guaranteed to be minimum with respect to t. To extract a path we simply follow parent pointers until we reach the start node. The root points associated with the search nodes we encounter are the turning points on the optimal any-angle path from s to t.

Correctness and Optimality

To prove correctness and optimality, we show (i) the optimal path appears in the search space and (ii) when the target is expanded we have found an optimal path.

Theorem 1 *For any point p that appears on a row, there exists a node in the search tree that corresponds to the optimal path from s to p if such a path exists.*

Proof: Sketch, by induction. Consider an optimal path $\pi_k = \langle p_1, \ldots, p_k \rangle$ where $s = p_1$ and p_{k-1} is either a point one row apart from p_k (similar to a y_i point mentioned in Figure 1) or a corner point on the same row. By induction, there is a node (I, r) in the search tree that represents the optimal path to $p_{k-1} \in I$. Now following Definition 3, there is a node (I', r') that is a successor of (I, r) such that $p_k \in I'$ while $r' = r$ if p_k is visible from r and $r' = p_{k-1}$ otherwise, and the node represents the optimal path. □

We now assume the search space is explored by A* search employing Equation (1).

Theorem 2 *The first expanded node that contains the target t corresponds to the optimal path to t.*

Proof: Sketch. First we notice that the f value of a node is indeed the minimal value of all the nodes in the interval, which means that f is an under estimate of the actual cost to the target. Second we notice that, given a search node (I, r) and its successor (I', r'), for each point $p' \in I'$, the f value of p' is bigger than the f value of some point $p \in I$ ($p = r'$ if $r' \neq r$; p is the intersection of I and (r, p') otherwise); the f function is therefore monotonically increasing. Finally, the f function of a search node (I, r) is the length of the path if $t \in I$. Hence the f function of the nodes representing a sub-optimal path to t will eventually exceed the optimal path distance, while the f function of the nodes representing the optimal path will always remain under this value. □

Conclusion

We study any-angle pathfinding: a problem commonly found in the areas of robotics and computer games. It involves finding a shortest path between two points in a grid but asks that the path is not artificially constrained to the points of the grid. We give a new algorithm for this problem: Anya. Our approach works by representing sets of points from the grid as intervals and considers all points from an interval together at the same time. From each interval we select a representative point which has a minimum f-value. We show that this approach is both complete and optimal. Moreover, it requires no preprocessing and relies on no special data structures during search. Any-angle pathfinding has received significant attention from the AI and Game Development communities but until now it has been an open question whether any optimal online algorithm exists. Anya answers this question in the affirmative.

Acknowledgements

We thank Adi Botea and Patrik Haslum for helpful suggestions during the development of this work. NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

Botea, A.; Müller, M.; and Schaeffer, J. 2004. Near optimal hierarchical path-finding. *J. Game Dev.* 1(1):7–28.

Ferguson, D., and Stentz, A. 2005. Field D*: An interpolation-based path planner and replanner. In *ISRR*, 239–253.

Graham, R. L.; Knuth, D. E.; and Patashnik, O. 1989. *Concrete mathematics - a foundation for computer science*. Addison-Wesley.

Hershberger, J., and Suri, S. 1999. An optimal algorithm for Euclidean shortest paths in the plane. *SIAM J. Comput.* 28(6):2215–2256.

Liu, Y.-H., and Arimoto, S. 1992. Path planning using a tangent graph for mobile robots among polygonal and curved obstacles. *Int. J. Robot Research* 11:376–382.

Lozano-Pérez, T., and Wesley, M. A. 1979. An algorithm for planning collision-free paths among polyhedral obstacles. *Commun. ACM* 22(10):560–570.

Mitchell, J. S. B.; Mount, D. M.; and Papadimitriou, C. H. 1987. The discrete geodesic problem. *SIAM J. Comput.* 16(4):647–668.

Nash, A.; Daniel, K.; Koenig, S.; and Felner, A. 2007. Theta*: Any-angle path planning on grids. In *AAAI*, 1177–1183.

Pinter, M. 2001. Toward more realistic pathfinding. *Game Developer Magazine* 8(4).

Šišlák, D.; Volf, P.; and Pěchouček, M. 2009. Accelerated A* trajectory planning: Grid-based path planning comparison. In *4th Workshop on Planning and Plan Execution for Real-World Systems*.

Yap, P.; Burch, N.; Holte, R. C.; and Schaeffer, J. 2011. Block A*: Database-driven search with applications in any-angle path-planning. In *AAAI*.