Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Grid Pathfinding on the 2^k Neighborhoods

Nicolás Rivera
Department of Informatics
King's College London
London, UK

Carlos Hernández
Depto. de Ciencias de la Ingeniería
Universidad Andrés Bello
Concepción, Chile

Jorge A. Baier
Pontificia Universidad Católica de Chile
Center for the Semantic Web Research
Santiago, Chile

Abstract

Grid pathfinding, an old AI problem, is central for the development of navigation systems for autonomous agents. A surprising fact about the vast literature on this problem is that very limited neighborhoods have been studied. Indeed, only the 4- and 8-neighborhoods are usually considered, and rarely the 16-neighborhood. This paper describes three contributions that enable the construction of effective grid path planners for extended 2^k-neighborhoods. First, we provide a simple recursive definition of the 2^k-neighborhood in terms of the 2^{k-1}-neighborhood. Second, we derive distance functions, for any $k > 1$, which allow us to propose admissible heurisitics which are perfect for obstacle-free grids. Third, we describe a canonical ordering which allows us to implement a version of A* whose performance scales well when increasing k. Our empirical evaluation shows that the heuristics we propose are superior to the Euclidean distance (ED) when regular A* is used. For grids beyond 64 the overhead of computing the heuristic yields decreased time performance compared to the ED. We found also that a configuration of our A*-based implementation, without canonical orders, is competitive with the "any-angle" path planner Theta* both in terms of solution quality and runtime.

Introduction

Grid pathfinding is one of the most well-known problems in AI. It is important because of its applications, which include robotics (Lee and Yu 2009) and videogames (Björnsson et al. 2005). Furthermore, it still captures significant attention from the community. Notably, three editions of the Grid Path Planning Competition (GPPC) have been recently held (Sturtevant et al. 2015), putting to test latest advances in the area (Botea and Harabor 2013; Harabor and Grastien 2011; Uras, Koenig, and Hernández 2013).

Research on grid pathfinding has focused on simple 4-neighbor grids, in which cardinal movements are allowed, and 8-neighbor grids, which extend 4-neighbor movements with diagonal movements. Besides their simplicity, perhaps the main reason to focus on these grids is that good heuristics are well-known for them. Indeed, the Manhattan and Octile distances (Sturtevant and Buro 2005) are perfect heuristics for, respectively, 4- and 8-neighbor obstacle-free grids. Although distance functions on the 16-neighborhoods have been discovered and studied by researchers of the Computer Vision community (Marchand-Maillet and Sharaiha 1997), to our knowledge, evaluations of grid pathfinding over 16-neighborhoods (e.g., Nash 2012, Aine and Likhachev 2016) have never considered these heuristics and, rather, have used the Euclidean distance (ED).

Perfect heuristics for obstacle-free grids do not only allow algorithms like A* to find solutions quickly but they are a key enabler of other techniques for grid pathfinding. One example is the approach by Uras, Koenig, and Hernández (2013), winner of the 2013 GPPC, optimal track, which relies heavily on the Octile distance to compute subgoal graphs that are later exploited for fast pathfinding. Another example is FRIT (Rivera et al. 2014), the state-of-the-art real-time heuristic search pathfinding algorithm that uses the Octile/Manhattan distance to build an initial *ideal tree*.

In this paper we study grid pathfinding on the 2^k-neighborhoods. Our first contribution is a definition of such neighborhoods. A rather notable property we prove is that the radius of the smaller square that contains the movements is given by the Fibonacci series. Furthermore, we derive a distance function for the 2^k-neighborhood, the 2^k-tile heuristic, and prove its correctness. Our proof and construction works for any k, thus generalizing the Manhattan and Octile distances, and the distance function of Marchand-Maillet and Sharaiha. Our proof, however, seems much shorter and simpler than the one given by Marchand-Maillet and Sharaiha for the 16-neighborhood. Finally, we define canonical orderings (Harabor and Grastien 2011; Sturtevant and Rabin 2016) for the 2^k-neighborhood. These orderings are at the core of fast pathfinding algorithms.

We implemented two 2^k-grid path planners: regular A* and A* with our canonical orderings. We evaluated them over standard benchmarks. We tested different values of k and compared our 2^k-tile heuristic with the ED. Runtime of Canonical A* scales with k while this does not hold for regular A*. Our heuristics yield faster search compared to the ED, for every neighborhood with regular A*, and up to the 32-neighborhood with Canonical A*. We compared also with the any-angle path planner Theta*, and found that Theta* is outperformed by *regular* A* when $k = 32$.

Next we introduce background and then define the 2^k-neighborhood. Then we derive our 2^k-tile heuristic. We

continue proposing our canonical orderings, and finish with our empirical evaluation and conclusions.

Pathfinding in 4- and 8-neighbor Grids

An $m \times n$ grid, where m and n are positive naturals, is composed by mn cells, each of which is denoted by an ordered pair (x, y). In grid pathfinding we are given a grid and a set of obstacle cells in the grid, and the objective is to find a path from an initial cell (x_0, y_0) to a goal cell (x_g, y_g) that never goes through cell in a given set of obstacle cells. Given two integers m and n that are not both equal to 0, an (m, n) movement is one such that, when executed in cell (x, y) leads to cell $(x + m, y + n)$. An (m, n) movement is executable in cell (x, y) of grid G if $(x+m, y+n)$ is in G and no cell touched by the segment between defined by these two cells is an *obstacle*.[1] A path is a sequence of cells that result from the successive application of a sequence of executable moves. The cost of an (m, n) movement is $\sqrt{m^2 + n^2}$. The cost of a path is the sum of the cost of the movements in the path. Cell (x', y') is the neighbor of (x, y) if the former can be reached from the latter by performing one movement.

A grid G is a 4-neighbor grid when the allowed movements for (x, y) are those executable movements in:

$$\mathcal{M}^4 = \{(m, n) : |m| + |n| = 1\},$$

which can be interpreted as vertical and horizontal movements of cost 1. In addition, the allowed movements for an 8-neighbor grid are:

$$\mathcal{M}^8 = \{(m, n) : |m|, |n| \leq 1\} \setminus \{(0, 0)\},$$

which extends $\mathcal{M}^4$ with diagonal movements.

Below we treat ordered pairs like vectors. As such, if k is natural or real number, and (x, y) is an ordered pair, then the notation $k(x, y)$ denotes the pair (kx, ky). Moreover, $(x, y) + (x', y')$ denotes $(x + x', y + y')$. We say that a vector (x, y) is in the first quadrant if $x, y \geq 0$. If two vectors (x, y) and (x', y') are in the first quadrant then we say that $(x, y) \preceq (x', y')$ if the angle with the Y axis formed by (x, y) is smaller than or equal to that of (x', y') (or, alternatively, (x', y') is to the right of (x, y), clockwise). Formally $(x, y) \preceq (x', y')$ iff $xy' \leq x'y$. Furthermore, $\|(x, y)\| = \sqrt{x^2 + y^2}$.

The 2^k-neighborhoods

We now define the 2^k-neighborhoods, i.e. $\mathcal{M}^{2^k}$, for any k. Below we define the succession $\mathcal{M}^{2^2}, \mathcal{M}^{2^3}, \ldots$ in terms of another succession $\mathcal{N}^0, \mathcal{N}^1, \ldots$, which is simpler to define. Thus for now, we focus our attention on the latter succession. Its first element, $\mathcal{N}^0$, contains the 4-neighborhood movements on the first (i.e., positive) quadrant. Therefore, $\mathcal{N}^0 = \langle (0, 1), (1, 0) \rangle$. Now if $\mathcal{N}^i = \langle a_0, \ldots, a_n \rangle$, $\mathcal{N}^{i+1}$ is constructed from $\mathcal{N}^i$ by inserting between each consecutive elements a_j and a_{j+1} the sum $a_j + a_{j+1}$. Formally,

[1]For diagonal movements sometimes another definition is adopted. More on this in our experimental section.

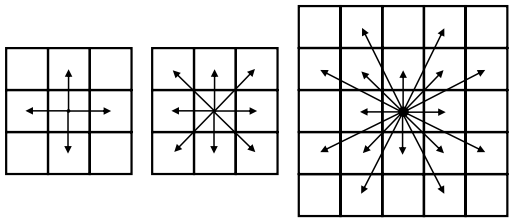

Figure 1: 4-, 8-, and 16-neighborhoods.

$\mathcal{N}^{i+1} = \langle b_0, b_1, \ldots, b_{2n} \rangle$, where

$$b_{2j} = a_j, \text{when } 0 \leq j \leq n, \text{ and} \quad (1)$$

$$b_{2j+1} = a_j + a_{j+1}, \text{when } 0 \leq j < n. \quad (2)$$

The first three elements of the succession are:

$$\mathcal{N}^0 = \langle (0, 1), (1, 0) \rangle \quad (3)$$

$$\mathcal{N}^1 = \langle (0, 1), (1, 1), (1, 0) \rangle \quad (4)$$

$$\mathcal{N}^2 = \langle (0, 1), (1, 2), (1, 1), (2, 1), (1, 0) \rangle \quad (5)$$

Observe that elements in $\mathcal{N}^i$ are pairwise linearly independent. This is an important property because it says that two movements are unique in the sense that they cannot by simulated by repeating other movements in the neighborhood. We can prove this fact by induction. Indeed, $\mathcal{N}^0$ is such that its two elements are linearly independent. Moreover, every new element added to $\mathcal{N}^{i+1}$ that was not in $\mathcal{N}^i$ is linearly independent from each element in $\mathcal{N}^i$, because it is the sum of two (linearly independent) elements of $\mathcal{N}^i$. We formalize this as:

Proposition 1 *If $u, v \in \mathcal{N}^i$, and there exists a k such that $u = kv$, then $u = v$.*

Now we give a definition for the full set of movements in terms of $\mathcal{N}$. Observe that, to generate a complete neighborhood from $\mathcal{N}^i$ we need to map every movement to all four quadrants. As a result, for the orthogonal movements ($(0, 1)$ and $(1, 0)$), we generate two new movements ($(0, -1)$ and $(-1, 0)$) while each non-orthogonal (x, y) generates 3 additional movements that correspond to changing the signs of x and y. In short, for every natural i, we define:

$$\mathcal{M}^{2^{2+i}} = \{(kx, k'y) \mid (x, y) \text{ is in } \mathcal{N}^i \text{ and } k, k' \in \{-1, 1\}\} \quad (6)$$

Proposition 2 *The cardinality of $\mathcal{M}^{2^k}$ is 2^k, for every $k \geq 2$.*

Proof: Observe that $|\mathcal{N}^0| = 2$ and that $|\mathcal{N}^{i+1}| = 2|\mathcal{N}^i| - 1$, which is a recurrence equation whose solution is $|\mathcal{N}^i| = 2^i + 1$. The size of $\mathcal{M}^{2^{2+i}}$ is $4(|\mathcal{N}^i| - 2) + 4$ because there are 4 orthogonal movements and non-orthogonal movements need to be considered 4 times. This yields $|\mathcal{M}^{2^{2+i}}| = 2^{2+i}$. ■

A property of our definition is that the size of the smallest square in which all movements in $\mathcal{N}^i$ can be circumscribed

grows exponentially (see Figure 1). Specifically, the *radius* of the square, defined as the largest of all coordinates of the movements of the region, grows like the Fibonacci series.

To establish this result let us define $Fib(i) = Fib(i-1) + Fib(i-2)$ for $i > 1$ and $Fib(0) = Fib(1) = 1$.

Theorem 1 *Let ℓ^i be the largest coordinate among all of the pairs in $\mathcal{N}^i$ then $\ell^i = Fib(i)$.*

To simplify notation, below we use $\mathcal{N}_k$ to denote the element in position k of sequence $\mathcal{N}$, where k may take values from 0 to $|\mathcal{N}| - 1$; that is, the first position of $\mathcal{N}$ is $\mathcal{N}_0$. To prove Theorem 1, we first observe that Equation 2 entails that $\mathcal{N}^i_k = \mathcal{N}^{i-1}_{(k-1)/2} + \mathcal{N}^{i-1}_{(k+1)/2}$, when k is and odd number. In addition, observe that $(k+1)/2$ and $(k-1)/2$ are consecutive and thus one of them is an *even* number. Therefore, if $(k+1)/2$ is even, then, via Equation 1:

$$\mathcal{N}^i_k = \mathcal{N}^{i-1}_{(k-1)/2} + \mathcal{N}^{i-2}_{(k+1)/4}. \tag{7}$$

Otherwise, if $(k-1)/2$ is even:

$$\mathcal{N}^i_k = \mathcal{N}^{i-1}_{(k+1)/2} + \mathcal{N}^{i-2}_{(k-1)/4}. \tag{8}$$

The proof of for Theorem 1 is straightforward from the following two lemmata.

Lemma 1 *Each coordinate of any pair in $\mathcal{N}^i$ is lower than or equal to $Fib(i)$.*

Proof: By induction on i, we verify by inspection that this is true in the base case ($i = 0$). Assume the property holds for every $i \in \{1, \ldots, n-1\}$. Now, we prove that for every k, $\mathcal{N}^n_k \leq Fib(n)$. Indeed, assume k is even. Then $\mathcal{N}^n_k = \mathcal{N}^{n-1}_{k/2}$. By the inductive hypothesis, $\mathcal{N}^n_k \leq Fib(n-1)$ and therefore we conclude $\mathcal{N}^n_k \leq Fib(n)$, because Fib is non-decreasing.

Now assume k is odd. Then we have two cases (Equations 7 and 8) which are proven analogously. The proof follows in the same way for both cases so we simply assume:

$$\mathcal{N}^n_k = \mathcal{N}^{n-1}_{(k-1)/2} + \mathcal{N}^{n-2}_{(k+1)/4} \tag{9}$$

By the induction hypothesis and the definition of Fib,

$$\mathcal{N}^n_k \leq Fib(n-1) + Fib(n-2) = Fib(n),$$

which concludes the proof. ■

Lemma 2 *Let F_i be the succession defined by $F_0 = 0$, and $F_i = 2F_{i-1} + (-1)^{i-1}$, for $i > 0$. Then the second coordinate of $N^i_{F_i}$ equals $Fib(i)$, for every i.*

Proof: The first observation is that F_i is formed by adding or substracting 1 to an even number ($2F_{i-1}$) and thus is odd, for every $i > 0$. Now the proof proceeds by induction on i. Note that for the base case ($i = 0$) the property holds. Assume the property holds for every $i \in \{1, \ldots, n-1\}$. Now we prove it also holds for n.

We now have two cases: n is even and n is odd. We focus only on the former case because the proof for the latter is analogous.

Because n is even, $F_n = 2F_{n-1} - 1$ and $(F_n + 1)/2 = F_{n-1}$ is odd. In addition,

$$\frac{F_n - 1}{4} = \frac{F_{n-1} - 1}{2} = \frac{2F_{n-2} + 1 - 1}{2} = F_{n-2}.$$

Now we use Equation 8 to write:

$$\mathcal{N}^i_k = \mathcal{N}^{i-1}_{F_{i-1}} + \mathcal{N}^{i-2}_{F_{i-2}}. \tag{10}$$

With the inductive hypothesis and the definition of Fib we obtain the desired result. ■

A Distance for the 2^k-Neighborhood

In grid path finding it is essential to use a good heuristic. In 4- and 8- neighbor grids, good heuristics are well-known: the Manhattan and the Octile Distance, respectively. These heuristics are actually distances because they correspond to the shortest path that can be traversed between an arbitrary location and the goal location, ignoring any obstacles.

So now we focus on answering a more general question: Given an arbitrary i, what is the cost of the shortest path between $(0, 0)$ and (x, y) when only movements in $\mathcal{N}^i$ can be applied?

The problem can be formalized as an Integer Program (IP). Indeed, if $\mathcal{N}^i = \langle v_0, \ldots, v_n \rangle$, we want to solve the following IP, $\mathcal{P}^i$:

$$\text{Minimize } \sum_{i=0}^{n} \alpha_i \|v_i\| \tag{11}$$

subject to:

$$\sum_{i=0}^{n} \alpha_i v_i = (x, y), \quad \alpha_i \geq 0, \quad \text{for every } i \in \{0, \ldots, n\}, \tag{12}$$

where each α_i is an integer variable that intuitively represents how many times movement v_i is applied.

Below we prove that the linear-programming (LP) relaxation of $\mathcal{P}^i$ has always an *integer* solution. Before proving such a result, we turn our attention to the LP relaxation of $\mathcal{P}^i$, that we denote $\mathcal{P}^i_{\text{LP}}$, studying its properties and solution.

A Solution to $\mathcal{P}^i_{\text{LP}}$

Our first result establishes that we can focus on a simpler, two-variable problem instead of the original n-variable problem.

Theorem 2 *Assume $\mathcal{N}^i = \langle v_0, \ldots, v_n \rangle$, and let j be such that $a_j \preceq (x, y) \preceq a_{j+1}$. Then $\mathcal{P}^i_{LP}$ is equivalent to $\hat{\mathcal{P}}^i_{LP}$, defined as:*

$$\text{Minimize } \alpha_j \|v_j\| + \alpha_{j+1} \|v_{j+1}\| \tag{13}$$

subject to:

$$\alpha_j v_j + \alpha_{j+1} v_{j+1} = (x, y), \tag{14}$$

Proof: Assume that the optimal solution to $\mathcal{P}^i_{\text{LP}}$ is an assignment, σ, to all variables α_i such that $\sigma(\alpha_k) > 0$, for some k such that $k < j$. (The same proof below can be slightly modified to accommodate the case in which $k > j+1$, but we omit this for simplicity.) Now consider the curve formed by putting all vectors v_i one after the other, with v_k put first. Because the curve ends in (x, y) and starts in $(0, 0)$, it must itersect the ray generated by vector v_j. This curve can then be "split" into two parts: the part that goes before such an intersection, and the part that goes after the intersection. Moreover, there is one of the vectors that crosses the ray of v_j, and such a vector cannot be v_k. Let us assume this vector is v_T.

Now we formalize the fact that the sum can be separated in two parts, one containing the vectors before the intersection; the other, containing the vectors after it. This means (x, y) can be expressed as the sum of vectors whose indices are in two disjoint sets: A^- and A^+, representing, respectively the vectors before and after the intersection. Note that A^- is non-empty since it must contain k. Furthermore, for some positive value m and some non-negative value $\beta \leq \alpha_T$ we have that:

$$\sum_{\ell \in A^-} \alpha_\ell v_\ell + \beta v_T = m v_j, \quad \text{for some } m \tag{15}$$

$$m v_j + (\alpha_T - \beta) v_T + \sum_{\ell \in A^+} \alpha_\ell v_\ell = (x, y) \tag{16}$$

Note that Equation 16 yields a different assignment, say σ', that satisfies all constraints of $\mathcal{P}^i_{\text{LP}}$ and which assigns to 0 all α_i such that $i \in A^-$. Furthermore $D(\sigma') < D(\sigma)$. Indeed, this new assignment has replaced a curve of the original solution by a *straight line*.

The argument above can be used with any assignment that uses vectors whose index is either lower than j or greater than $j+1$. We conclude therefore that the optimal solution must be such that $\alpha_i = 0$, for every i such that $0 \leq i < j$ or such that $j+1 < i \leq n$. Therefore, the solution to $\mathcal{P}^i_{\text{LP}}$ is equivalent to that of $\hat{\mathcal{P}}^i_{\text{LP}}$.

Observe, finally that $\hat{\mathcal{P}}^i_{\text{LP}}$ has only one feasible assignment yielded by the solution to Equation 14, which must be such that both α_j and α_{j+1} be positive. ■

Given Theorem 2, a solution to $\mathcal{P}^i_{\text{LP}}$ can be computed very easily since $\hat{\mathcal{P}}^i_{\text{LP}}$ has *only one* feasible point. Specifically, this is the assignment to α_j and α_{j+1} that solves the system of two linear equations of Equation 14. Observe also that the solution to $\mathcal{P}^i_{\text{LP}}$ can therefore be computed in constant time.

The following result finally establishes that a solution to $\hat{\mathcal{P}}^i_{\text{LP}}$ has an integer solution, for every $i \geq 0$, and therefore that it is a solution for $\mathcal{P}^i$.

Theorem 3 *$\hat{\mathcal{P}}^i_{LP}$ has an integer optimal solution, for every $i \geq 0$.*

Proof: We prove that the values for α_j and α_{j+1} that satisfy Equation 14, are integer, for every i. The proof is by induction on i. For the base case, observe that for $i = 0$ the solution is $\alpha_j = x$ and $\alpha_{j+1} = y$, and thus integer.

Now we assume that the solution for $\hat{\mathcal{P}}^k_{\text{LP}}$ is integer. Let us assume that Equation 14 for $\hat{\mathcal{P}}^{k+1}_{\text{LP}}$ is:

$$\alpha_j v_j + \alpha_{j+1} v_{j+1} = (x, y), \tag{17}$$

Beacause v_j and v_{j+1} are consequtive pairs in $\mathcal{N}^{k+1}$, one of them is in $\mathcal{N}^k$ whereas the other is the sum of two elements in $\mathcal{N}^k$. Without loss of generality, let us assume the former is v_{j+1} and that the latter is v_j. Then we can rewrite Equation 17 as $\alpha_j v_{j-1} + \alpha_j v_{j+1} + \alpha_{j+1} v_{j+1} = (x, y)$, or equivalently:

$$\alpha_j v_{j-1} + (\alpha_j + \alpha_{j+1}) v_{j+1} = (x, y), \tag{18}$$

where $v_{j-1}, v_{j+1} \in \mathcal{N}^k$. Now we use the inductive hypothesis to conclude that the system of equations given by (18) has an integer solution; hence α_j and $\alpha_j + \alpha_{j+1}$ are integer, which ultimately implies α_{j+1} is integer too. ■

The following result is straightforward from Theorem 3. We infer our heuristics and canonical orderings from there.

Corollary 1 *Assume $\mathcal{N}^{k-2} = \langle v_0, \ldots, v_m \rangle$, and that cell (x, y) is in the positive quadrant. Furthermore, let j be such that $v_j \preceq (x, y) \preceq v_{j+1}$. Then (x, y) can be reached optimally from $(0, 0)$ on the 2^k-neighborhood using an integer combination of v_j and v_{j+1}.*

The 2^k-Tile Heuristic for 2^k-Grids

From Theorems 2 and 3, we obtain the following algorithm that computes length of the optimal path to a point (x, y) in a 2^k-connected neighborhood, assuming $\mathcal{N}_{k-2} = \langle v_0, \ldots, v_m \rangle$:

1. Determine an j such that $v_j \preceq (x, y) \preceq v_{j+1}$. This can be done by iterating over the elements of $\mathcal{N}_{k-2}$. This can be checked efficiently using a binary search scheme.
2. Solve the system of two linear equations given by Equation 17, and return $\alpha_j \|v_j\| + \alpha_{j+1} \|v_{j+1}\|$.

Algorithm 1 shows heuristics for some 2^k-neighborhoods.

Canonical Orderings for 2^k

We have formally characterized the 2^k-neighborhoods and given admissible heuristics which are perfect for obstacle-free grids. Yet an important issue of 2^k-neighborhoods is the increase in branching factor, which is exponential with k. By just observing this fact one might, at first sight, discard an A* implementation of 2^k-neighborhoods. Indeed, with each A* expansion we would need to generate 2^k successors, many of which potentially have to be added to the Open list, incurring in much overhead.

Nevertheless an opposing and quite interesting observation is that in obstacle-free 2^k-grids, the number of optimal paths between two cells may *decrease exponentially* as k increases. To see this, let us consider the number of paths between $(0, 0)$ and $(4, 2)$. When $k = 2$ (4-neighborhood), we reach $(4, 2)$ with any sequence of moves that has 4 vertical moves and 2 horizontal moves, yielding $\frac{6!}{4!2!} = 15$ paths. When $k = 3$ (8-neighborhood), we require 2 diagonal (1,1)

Algorithm 1: Heuristics for the 8-, 16-, and 32-neighborhoods

```
function h_8(x, y)
    if x > y then swap x and y
    return (y − x) + √2 x
function h_16(x, y)
    if x > y then swap x and y
    if 2x < y then
        return (y − 2x) + √5 x
    else
        return √5(y − x) + √2(2x − y)
function h_32(x, y)
    if x > y then swap x and y
    if 3x < y then
        return (y − 3x) + √10 x
    else if 2x < y then
        return √10(y − 2x) + √5(3x − y)
    else if 3x < 2y then
        return √5(2y − 3x) + √13(2x − y)
    else
        return √13(y − x) + √2(3x − 2y)
```

movements plus 2 vertical movements, yielding $\frac{4!}{2!2!} = 6$ optimal paths. Finally, when $k \geq 4$ there is only one optimal path with two $(2, 1)$ moves.

Thus even though the branching factor increases exponentially with k, the number of optimal paths between two points may decrease exponentially with k. Therefore, at least at a theoretical level, it is not obvious that increasing k should degrade performance of a standard A*.

But recent developments in 8-neighbor grid pathfinding can also be leveraged for the 2^k-neighborhoods. Canonical orderings—recently described by Sturtevant and Rabin (2016), but originally underlying in the *Jump Point Search* (JPS) (Harabor and Grastien 2011; 2014) algorithm—is one of these. A canonical ordering is a total order between paths on an 8-neighbor obstacle-free grid where paths that have diagonal movements first are preferred to those that use diagonal movements later.

Sturtevant and Rabin (2016) showed how to incorporate the notion of canonical orderings within best-first search (A*, in particular) for 8-neighbor grids. In summary, there are two elements that need to be incorporated. First, we distinguish between two types of nodes: *bidimensional* and *unidimensional*. The former nodes may generate successors in more than one direction, whereas the latter may only generate one successor in a single direction. The initial state of the search is a bidimensional node that has 8 successors. To make the description more precise, a bidimensional node $(x + d, y + d)$ whose parent is (x, y) (thus, $d \in \{-1, 1\}$) may expand nodes $(x + 2d, y + 2d)$, $(x + d, y + 2d)$ and $(x + 2d, y + d)$. The first is a bidimensional node, whereas the latter two are set to be unidimensional. Furthermore, if $(x + d, y)$ is a unidimensional node whose parent is (x, y) (thus, $d \in \{-1, 1\}$) may expand only $(x + 2d, y)$ and such a successor is set as unidimensional. The description is analogous for unidimensional the form $(x, y + d)$. If (x', y') is the successor of (x, y) following the rules described above, we say that (x', y') is a canonical successor of (x, y).

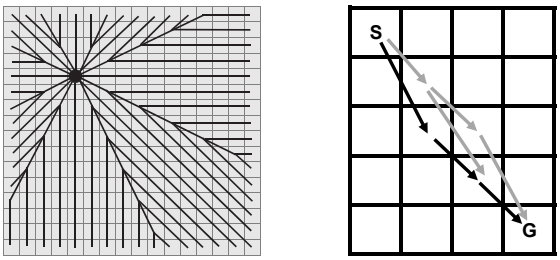

Figure 2: Left: An ilustration of the preferred canonical paths on the 16-neighborhood. Right: A preferred canonical path (in black) and non-preferred paths between S and G (in gray).

The second element that needs to be incorporated are *jump points*. These are certain cells in the grid which, if expanded, have to be considered as *bidimensional*, even if they were generated by a unidimensional node. More generally, these are nodes that have *forced successors*; that is, they will generate successors that do not follow the rules imposed by the canonical order. Forced successors are required to preserve completeness. In the proximity of an obstacle, it may happen that a cell is not the canonical successor of any cell and thus we must guarantee such a cell is generated by some other reachable cell. Formally, a node (x', y') is a forced successor of (x, y) if (1) (x', y') is a successor of (x, y) and (2) (x', y') is not a canonical successor of any other node (x'', y''). The geometric conditions under which a cell is set to be a jump point are described in detail by Sturtevant and Rabin (2016).

Canonical A*, the version of A* that results from imposing canonical successors, has the interesting advantage that average branching factor is reduced, and, moreover, because of the ordering, each cell can only be generated once which also has an impact on performance (because, potentially, the number of times a state may enter the Open list is reduced).

Defining a canonical ordering for 2^k is surprisingly simple. This is because by Corollary 1 we know that each cell in the obstacle-free grid is reached optimally by applying only two types of moves from $\mathcal{N}^{k-2}$, v_j and v_{j+1}, for some j. To define a canonical ordering we simply need to define which of these movements we prefer. To subsume the definition for 8-neighbor grids, we prefer to use first the movement whose subindex is an odd number. Observe that this means we prefer $(1, 1)$ over $(1, 0)$ on 8-connected grids.

Given this canonical ordering it is simple to define canonical successors in a manner that is analogous to 8-neighbor grids. Indeed, there are two types of nodes: unidimensional and bidimensional. The initial state is a bidimensional node with 2^k successors. Like above, unidimensional nodes may expand a single nodes in the direction from which they where expanded. A bidimensional node (x, y) that was expanded by applying movement v_j in N_{k-2} (we restrict to the positive quadrant for simplicity) may expand only 3 succesors $(x, y) + v_j$, $(x, y) + v_{j-1}$, and $(x, y) + v_{j+1}$, where the first is a bidimensional node and the 2 following are unidimensional nodes. An important observation is that

			2^k-tile Heuristics			Euclidean Heuristics		
N	**Cost**	**Mov**	**Exp**	**Perc**	**Time**	**Exp**	**Perc**	**Time**
				Starcraft				
4	804.2	804	55,501	313,662	10.59	110,911	918,104	25.42
8	668.0	570	52,840	595,331	12.98	69,373	619,202	17.40
16	642.5	425	52,884	607,963	17.17	58,182	601,855	18.56
32	637.0	335	53,147	621,548	24.16	55,165	618,450	24.87
64	635.4	273	53,217	657,585	37.94	54,161	650,121	38.12
128	634.8	230	53,279	690,190	60.32	53,769	685,635	60.84
				BG				
4	304.6	305	8,025	37,310	1.55	16,161	109,059	3.46
8	247.4	207	6,881	62,412	1.67	9,943	73,383	2.40
16	239.0	159	6,818	65,002	2.07	8,066	69,749	2.42
32	237.1	128	6,771	65,910	2.89	7,370	70,011	3.12
64	236.5	105	6,711	70,723	4.50	7,052	72,990	4.69
128	236.3	88	6,694	75,605	7.49	6,895	77,309	7.69
				WC3				
4	245.3	245	6,979	33,532	1.37	14,160	99,299	3.19
8	206.8	179	6,696	64,297	1.65	9,102	69,477	2.23
16	199.8	139	6,772	68,030	2.14	7,684	69,219	2.40
32	198.1	113	6,803	69,741	3.05	7,192	71,280	3.19
64	197.5	93	6,770	75,119	4.84	6,987	75,450	4.95
128	197.3	80	6,764	80,179	8.05	6,889	80,515	8.15

Table 1: A* on the 2^k-Neighborhood.

	2^k-tile Heuristics			Euclidean Heuristics		
N	**Exp**	**Perc**	**Time**	**Exp**	**Perc**	**Time**
			Starcraft			
4	58,617	195,956	8.79	110,941	888,288	23.67
8	53,620	587,413	10.72	69,386	557,109	14.28
16	53,174	516,861	11.65	58,191	473,421	12.53
32	53,259	495,083	12.56	55,173	467,565	12.59
64	53,304	505,937	13.80	54,166	490,821	12.62
128	53,335	546,201	15.29	53,774	535,795	14.20
			BG			
4	10,859	28,767	1.60	16,192	105,458	3.39
8	7,950	68,281	1.51	9,957	65,254	2.00
16	7,223	55,004	1.49	8,079	52,756	1.66
32	6,946	51,589	1.53	7,382	50,600	1.58
64	6,785	52,365	1.63	7,062	52,457	1.57
128	6,752	57,027	1.89	6,903	56,930	1.69
			WC3			
4	8,283	23,617	1.25	14,183	97,017	3.08
8	7,089	62,609	1.37	9,112	61,797	1.84
16	6,926	54,974	1.46	7,692	52,467	1.61
32	6,859	52,956	1.53	7,198	51,358	1.55
64	6,810	54,515	1.62	6,992	53,671	1.55
128	6,792	59,054	1.88	6,893	58,432	1.67

Table 2: Canonical A* on the 2^k-Neighborhood.

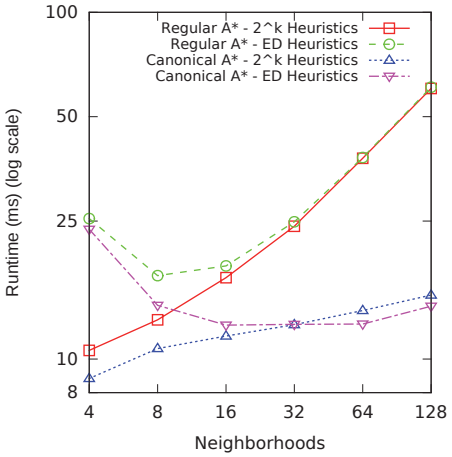

Figure 3: Average runtime in Starcraft maps.

most bidimensional nodes, like in 8-neighbor grids, have a branching factor of 3, independent of k. Figure 2 shows an illustration of canonical paths on the 16-neighhorhood.

Forced neighbors are defined as described above. A cell is a jump point if it is a jump point on the 8-neighborhood. The proof of the following result is straightforward but requires an enumeration of cases (omitted for space).

Theorem 4 *Canonical A* on the 2^k-Neighborhood is complete.*

Empirical Findings

The objective of our evaluation was threefold. First, we wanted to investigate the impact of using 2^k-neighbor grids in pathfinding with A*, the most standard heuristic search algorithm, comparing the performance of the 2^k-tile heuristic with the Euclidean heuristic. Second, we wanted to investigate the impact of using canonical orders in pathfinding with A* in 2^k-neighbor grids, and to compare regular A* with canonical A*. Third, we wanted to do a preliminary study on the performance of Theta* (Daniel et al. 2010) and pathfinding with 2^k-neighbor grids.

All algorithms have a common code base and use a standard binary heap for *Open*. All experiments were ran on a 2.60GHz Intel Core i7 under Linux.

We used maps from the MovingAI repository (Sturtevant 2012). Specifically, we used five Starcraft maps of size 1024×1024 cells (Cauldron, Expedition, Octopus, PrimevalIsles and TheFrozenSea), all 75 maps of size 512×512 from the video game Baldur's Gate (BG) and all 36 maps of size 512×512 maps from the video game World of Warcraft III (WC3). For each Starcraft map we generated 150 random solvable problems. For each BG and WC3 map we generated 50 random solvable problems. We present averages of 750 problems on Starcraft (5×150), 3750 problems on BG (75×50) and 1800 problems on BG (36×50).

Table 1 shows the average solution cost (Cost), agent moves (Mov), cell expansions (Exp), heap percolations (Perc) and runtime (Time) of regular A* for different neighborhoods. We evaluated six values for 2^k: $4, 8, 16, 32, 64$ and 128. The expansions, percolations, and runtime are shown for A* with the 2^k-tile heuristic and A* with Euclidean heuristic. We make the following observations.

- Solution cost improves when k increases in all three benchmarks. The improvements are marginal as k increases over 5 (32-neighborhood).
- The runtime increases with k in all three benchmarks. This can be explained by the larger branching factor.
- A* runs faster using the 2^k-tile heuristic than using the Euclidean heuristic, for every k. Nonetheless, for $k \geq 8$ (256-neighborhood) A*, used with ED is slightly faster due to the overhead in computing our heuristic.

Table 2 shows cell expansions, heap percolations and runtime of A* with canonical orders for the two heuristics.

With both, runtime increases smoothly. We can observe that Canonical A* is more efficient than regular A*, especially for higher values of k. Here, the 2^K-tile heuristic is more efficient than the Euclidean heuristic up to $k = 5$ (32-Neighborhood), for higher values of k, the opposite happens. This can be explained by the overhead to compute the heuristics. In regular A* the overhead of cell expansions dominate the overhead to calculate the heuristics. In Canonical A*, the overhead of cell expansions is reduced drastically. Figure 3 shows in the Starcraft maps, how the runtime increases when the amount of neighbor increases.

Finally, we implemented 2^k-neighborhood on top of the regular A* code used in (Uras and Koenig 2015) (Code available at: http://idm-lab.org/anyangle). The original code was used to evaluated any-angle search algorithms. We compared Theta* with the octile distance as heuristic with regular A* with the 2^K-tile heuristic with different neighborhoods on the Dragon Age: Origins maps. We used the scenarios of the MovingAI repository as test cases. The main observation is that from $2^k = 32$, the solution cost returned by regular A* is better than the solution cost returned by Theta*, and for $2^k = 32$, regular A* is 1.37 times faster than Theta*. For higher values of 2^k, regular A* is slower than Theta*.

Summary and Perspectives

We presented three key contributions towards the construction of effective grid path planners on the 2^k-neighborhood. First, we formally define the 2^k-neighborhood; second, we propose the 2^k-tile heuristic, and third, we define canonical orderings. We show that, indeed, using larger neighborhoods yields better solutions with a small sacrifice in runtime. Moreover, we showed that our Regular A* implementation is competitive with Theta*, a state-of-the-art sub-optimal any-angle path planner.

This research opens the door to future work that could investigate the use of 2^k-neighborhoods in any-angle incremental search, a problem relevant in robotics, and could compare its performance with optimal any-angle path planners such as Anya (Harabor et al. 2016). Integration of these techniques into state-of-the-art subgoaling techniques and jump point search, which currently yield very fast planners, is also a promising avenue of research.

Acknowledgements

We thank Antonio Díaz for his implementation of the 2^k neighborhood on top of the Theta* Codebase. We thank Dietrich Daroch for his comments on an earlier draft of this paper. Nicolás Rivera acknowledges funding from the Becas Chile initiative. Carlos Hernández and Jorge Baier are grateful to Fondecyt which partly funded this work through grants 1150328 and 1161526.

References

Aine, S., and Likhachev, M. 2016. Truncated incremental search. *Artificial Intelligence* 234:49–77.

Björnsson, Y.; Enzenberger, M.; Holte, R. C.; and Schaeffer, J. 2005. Fringe search: Beating A* at pathfinding on game maps. *CIG* 5:125–132.

Botea, A., and Harabor, D. 2013. Path planning with compressed all-pairs shortest paths data. In Borrajo, D.; Kambhampati, S.; Oddi, A.; and Fratini, S., eds., *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS)*. AAAI.

Daniel, K.; Nash, A.; Koenig, S.; and Felner, A. 2010. Theta*: Any-angle path planning on grids. *Journal of Artificial Intelligence Research* 39:533–579.

Harabor, D. D., and Grastien, A. 2011. Online graph pruning for pathfinding on grid maps. In Burgard, W., and Roth, D., eds., *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI)*. AAAI Press.

Harabor, D. D., and Grastien, A. 2014. Improving jump point search. In Chien, S. A.; Do, M. B.; Fern, A.; and Ruml, W., eds., *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS)*. AAAI.

Harabor, D.; Grastien, A.; Oz, D.; and Aksakalli, V. 2016. Optimal any-angle pathfinding in practice. *Journal of Artificial Intelligence Research* 56:89.

Lee, J.-Y., and Yu, W. 2009. A coarse-to-fine approach for fast path finding for mobile robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009. IROS 2009.*, 5414–5419. IEEE.

Marchand-Maillet, S., and Sharaiha, Y. M. 1997. Discrete convexity, straightness, and the 16-neighborhood. *Computer Vision and Image Understanding* 66(3):316–329.

Nash, A. 2012. *Any-Angle Path Planning*. Doctor of Philosophy, University of Southern California.

Rivera, N.; Illanes, L.; Baier, J. A.; and Hernández, C. 2014. Reconnection with the ideal tree: A new approach to real-time search. *Journal of Artificial Intelligence Research* 50:235–264.

Sturtevant, N. R., and Buro, M. 2005. Partial pathfinding using map abstraction and refinement. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI)*, 1392–1397.

Sturtevant, N. R., and Rabin, S. 2016. Canonical orderings on grids. In Kambhampati, S., ed., *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, 683–689. IJCAI/AAAI Press.

Sturtevant, N. R.; Traish, J. M.; Tulip, J. R.; Uras, T.; Koenig, S.; Strasser, B.; Botea, A.; Harabor, D.; and Rabin, S. 2015. The grid-based path planning competition: 2014 entries and results. In Lelis, L., and Stern, R., eds., *Proceedings of the 8th Symposium on Combinatorial Search (SoCS)*, 241. AAAI Press.

Sturtevant, N. 2012. Benchmarks for grid-based pathfinding. *Transactions on Computational Intelligence and AI in Games* 4(2):144 – 148.

Uras, T., and Koenig, S. 2015. An empirical comparison of any-angle path-planning algorithms [short paper]. In *Proceedings of the 8th Symposium on Combinatorial Search (SoCS)*, 206–210. AAAI Press.

Uras, T.; Koenig, S.; and Hernández, C. 2013. Subgoal graphs for optimal pathfinding in eight-neighbor grids. In Borrajo, D.; Kambhampati, S.; Oddi, A.; and Fratini, S., eds., *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS)*. AAAI.