Proceedings of the Twelfth International
Symposium on Combinatorial Search (SoCS 2019)

Brigitte, a Bridge-Based Grid Path-Finder

Alban Grastien
Data61, Australia
alban.grastien@data61.csiro.au

Abstract

We present BRIGITTE, a new path-finding algorithm for 8-connected grids. BRIGITTE partitions the map into large *regions* and computes *bridges* between every pair of regions. At runtime, BRIGITTE finds the best bridge for the path query and builds the path from the bridge. BRIGITTE competes favourably compared to CH-SG-R and COPP, although she currently requires extensive pre-processing.

1 Problem Definition and Brigitte

We are interested in the optimal path-finding problem over an 8-connected uniform-cost grid, with four cardinal moves of length 1 and four ordinal moves of length $\sqrt{2}$. "Corner-cutting" is also allowed as illustrated by the south-east move from C on Fig. 1, used to illustrate the following definitions.

Definition 1 (Region) *A* region $\mathcal{R}$ *is a non-empty set of cells that is* grid-convex*, i.e., it satisfies:*

1. $\forall\{s,t\} \subseteq \mathcal{R}$*. all the cells of one of the optimal paths between* s *and* t *belong to* $\mathcal{R}$*; and*
2. $\forall\{s,t\} \subseteq \mathcal{R}$*. the grid distance* $d(s,t)$ *between* s *and* t *equals the octile distance* $h(s,t)$*, i.e., the distance if there was no obstacle.*

For instance Q in Fig. 1 cannot be added to the left region as $d(A,Q) = 3 + 2\sqrt{2} > 5 = h(A,Q)$. By definition computing the distance between any two cells s and t from the same region is simple (2nd item). Furthermore it is also simple to compute an optimal path between these cells. Practically the path can be computed by looking at the cell p in one of the maximum two directions from s towards t: if p is in the same region then it is on an optimal path to t, otherwise the cell p' reached from s through the other direction is. We use the notation $\pi_{\mathcal{R}}(s,t)$ for such a "region path".

Definition 2 (Abutment) *An* abutment *of region* $\mathcal{R}$ *is a triple* $\langle r,a,b\rangle$ *where i)* a *and* b *belong to* $\mathcal{R}$*, ii)* r *is a cell which is called the* root*, and iii) all cells* $c \in \mathcal{R}$ *that satisfy* $\vec{rc} = \alpha\cdot\vec{ra} + \beta\cdot\vec{rb}$ *(*$\alpha,\beta \geq 0$*) are visible from* r*. We define the* abutment path $\pi_{ab}(p)$ *as follows:*

- *if* $\vec{rp} = \alpha\cdot\vec{ra} + \beta\cdot\vec{rb}$ *(*$\alpha,\beta \geq 0$*) then* $\pi_{ab}(p) = \pi_{\rightarrow}(p,r)$ *is the (grid-discretised) straight path from* p *to* r*:*
- *otherwise,* $\pi_{ab}(p) = \pi_{\mathcal{R}}(p,x)\pi_{\rightarrow}(x,r)$*, i.e., a region path to* x *followed by a straight path from* x *to* r *where* x *is either* a *or* b*, whichever leads to the shortest path. Notice that* $|\pi_{\mathcal{R}}(p,x)\pi_{\rightarrow}(x,r)| = h(p,x) + h(x,r)$*.*

Clearly computing the path $\pi_{ab}(p)$ or its distance is simple because it only requires to i) determine whether p is in the cone and then ii) use the simple procedures described before.

Definition 3 *A* bridge *between two regions* $\mathcal{R}_1$ *and* $\mathcal{R}_2$ *is a triple* $\langle ab_1, ab_2, \delta\rangle$ *where* ab_1 *and* ab_2 *are abutments of* $\mathcal{R}_1$ *and* $\mathcal{R}_2$*, and* $\delta = d(r_1,r_2)$*, called the* core distance*, is the distance between the two roots* r_1 *and* r_2 *of* ab_1 *and* ab_2*. Given a bridge* $br = \langle ab_1, ab_2, \delta\rangle$ *between the two regions* $\mathcal{R}_1$ *and* $\mathcal{R}_2$ *and given two cells* $s \in \mathcal{R}_1$ *and* $t \in \mathcal{R}_2$*, the* bridge path $\pi_{br}(s,t)$ *is the path* $\pi_{ab_1}(s)\pi^*(r_1,r_2)\overline{\pi_{ab_2}(t)}$ *where* r_1 *and* r_2 *are the roots of* ab_1 *and* ab_2 *and* $\overline{\pi}$ *is the reverse of* π*. In this path, the optimal path* $\pi^*(r_1,r_2)$ *is called the* core path*.*

Computing the distance of the bridge path $\pi_{br}(s,t)$ is simple, because it is the sum $d(s,r_1) + \delta + d(r_2,t)$ of the distances of two abutment paths and the (known) core distance δ. Computing the bridge path itself is also simple if computing the core path is simple.

A bridge *covers* a pair of cells if the bridge path between these cells is optimal and either of its abutment paths is non-empty. For two regions $\mathcal{R}_1$ and $\mathcal{R}_2$, we say that a collection B of bridges is *complete* for this pair of regions if for all pair of cells $s \in \mathcal{R}_1$ and $t \in \mathcal{R}_2$, either t is visible from s or there exists a bridge $br \in B$ that covers this pair of cells.

Definition 4 *A* bridge network *is a pair* $\mathcal{N} = \langle \mathbb{R}, \mathbb{B}\rangle$ *where* $\mathbb{R}$ *is a partition of the grid into regions and* $\mathbb{B}$ *is a map that associates each pair of regions with a set of bridges that is complete for this pair.*

BRIGITTE computes a bridge network as a pre-processing stage, and uses this network to answer a path query $\langle s,t\rangle$. BRIGITTE determines the optimal path by checking whether s is visible from, or belongs to the same region as, t (in which case the path can be easily computed). Otherwise, BRIGITTE retrieves the collection of bridges that support the regions of the two cells, identifies the bridge that minimises the bridge distance, and returns the corresponding bridge path. This requires to compute the core path, iteratively. In practice, each bridge also stores the "sub-bridge" that supports the two abutment cores.

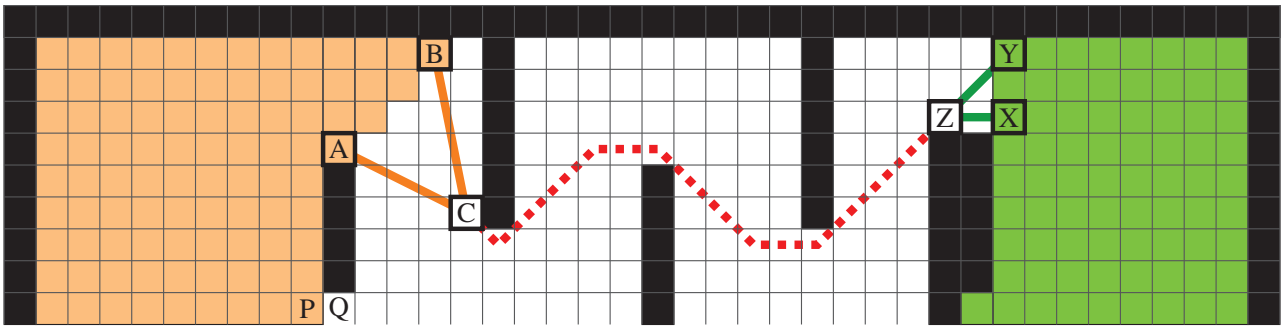

Figure 1: Two regions represented by the pale orange and pale green backgrounds. Two abutments $\langle C, A, B\rangle$ and $\langle Z, X, Y\rangle$. One bridge defined as the two abutments and the distance of the red dashed line. A bridge path over this bridge i) exits the left region between A and B, ii) reaches C, iii) follows the dashed line to Z, and iv) enters the right region between X and Y. Such a path can be suboptimal (e.g., one starting from P), which is why a pair of regions is generally supported by several bridges.

The size of the bridge network is essentially the total number of bridges in the network. This number is bounded by the number of cell pairs, which makes it quadratic. Given a grid, we call *zoom-in* a copy of this grid where each original cell is split into $z \times z$ cells, and such that the convexity is maintained. Then the grid can be partitioned in essentially the same set of regions and, except for some rare instances, the same set of bridges could be defined.

2 Empirical Evaluation and Future Works

We compared BRIGITTE (grastien.net/ban/projects/brigitte.htm) to the fastest existing competitors, CH-SG-R (Uras and Koenig 2018) and COPP (Salvetti et al. 2018), on the benchmarks from the 2012 Grid based Path-Planning Competition (Sturtevant 2012).

Pre-processing is expensive for BRIGITTE, and some of the grids could not be processed in reasonable time. Therefore we have no results for the RANDOM maps, some of the bigger ROOMS maps, as well as some STARCRAFT maps. The hardest grid that we processed is BLACK LOTUS, for which CH-SG-R requires 0.7s, COPP 3.5h, and BRIGITTE roughly 2 days. Clearly the pre-processing is still an issue and will need to be sorted.

The aggregated results for the grids that BRIGITTE was able to handle are given on Figure 2. BRIGITTE answers path queries in roughly 81% of the runtime required by COPP, and 26% of CH-SG-R's. These results are distributed over all benchmarks, although for some benchmarks COPP and BRIGITTE are similar. Memory-wise, BRIGITTE requires some amount of memory but quite reasonably so. The text representation of the bridge networks caps generally at 20Mb except for BLACK LOTUS, which requires 100Mb (down to 30Mb for a zipped version).

In conclusion BRIGITTE, while still in her infancy, manages to improve performance over already excellent solvers. The main concern is the amount of pre-processing that makes the approach currently impractical. Pre-processing will never be cheap (as for COPP) but we believe it can be eased. In particular the current implementation pushes for long abutments and short core bridges. We think that a bridge network with short abutments is simpler to construct, but we need to verify how it will affect performance. Also notice that pre-processing can be parallelised as each pair of regions can be processed independently.

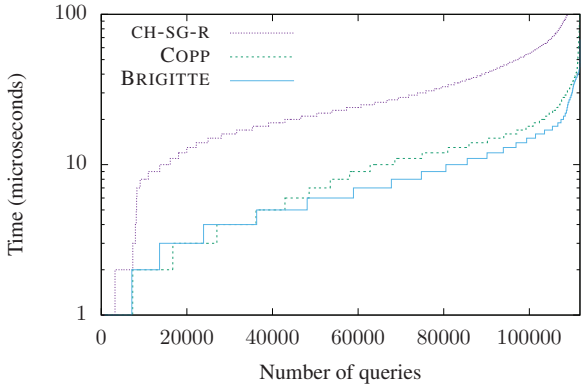

Figure 2: Aggregated results: ordered runtime of the path-finding problems. Below is better.

3 Acknowledgement

We want to thank Nathan Sturtevant, Adi Botea, Mattia Chiari, Sven Koenig, Tansel Uras, and Daniel Harabor for their advices and access to their code.

References

Salvetti, M.; Botea, A.; Gerevini, A. E.; Harabor, D.; and Saetti, A. 2018. Two-oracle optimal path planning on grid maps. In *28th International Conference on Automated Planning and Scheduling (ICAPS-18)*, 227–231.

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

Uras, T., and Koenig, S. 2018. Understanding subgoal graphs by augmenting contraction hierarchies. In *27th International Joint Conference on Artificial Intelligence (IJCAI-18)*, 1506–1513.