Exploratory Digraph Navigation Using A*

Fabrice Mayran de Chamisso
CEA, LIST*
F-91191 Gif-sur-Yvette, France

Laurent Soulier
CEA, LIST
F-91191 Gif-sur-Yvette, France

Michaël Aupetit
Qatar Computing Research Institute

Abstract

We describe Exploratory Digraph Navigation as a fundamental problem of graph theory concerned with using a graph with incomplete edge and vertex information for navigation in a partially unknown environment. We then introduce EDNA*, a simple A* extension which provably solves the problem and give worst-case bounds on the number of edges explored by said algorithm. We compare the performance of this algorithm to a non-exploratory strategy using A* and discuss its relation to existing algorithms such as D* Lite, PHA* with early stopping, EWP or exploration algorithms.

1 Exploratory Digraph Navigation (EDN)

1.1 Exploratory Digraph Navigation problems

Imagine a mobile robot navigating in a physical world $\mathcal{W}$ whose representation is a directed graph (digraph) $\mathcal{G}^R$ as sketched on Figure 1. Dotted lines on the figure represents data not available in the robot's internal map $\mathcal{G}$, also called *currently known graph* [Felner *et al.*, 2004], of the world at planning time. $\mathcal{G}^R \backslash \mathcal{G}$ represents space not explored yet. From its current position, the robot wants to reach a goal position with minimal effort. While planning its path to the goal on $\mathcal{G}$, it realizes that it essentially has two choices: using a known (safe) path, which may be quite long, or trying to find a *shortcut* thanks to places whose knowledge is incomplete (think of rooms containing doors not yet opened). However, taking a shortcut is risky since the expected shortcut may actually be a dead end, forcing the robot to turn back and replan its path. The said shortcut may also actually be longer than the safe path. Depending on its internal state (battery level, mission, ...), the robot may want to privilege the safe path or hope for luck and try an exploratory path which may lead to a decrease of traveled distance relative to the safe path while increasing the robot's knowledge of its environment.

The typical use of Exploratory Navigation is together with robotic Simultaneous Localization And Mapping (SLAM) where a robot models its environment as a graph [Choset and Nagatani, 2001; Bosse *et al.*, 2004] or as an occupancy grid [Elfes, 1989] (where each pixel can be considered as a vertex connected to its neighbors if space is traversable). Other problems related to graph theory such as the Traveling Salesman's Problem or routing problems in changing networks (where a set of packets must travel in a network without knowledge of all the routers) may also benefit from EDN. In these problems, the *computational cost* (algorithm execution time) associated to planning is small but not necessarily negligible compared to the *navigational cost* (energy used or distance physically traveled summed on each move until destination is reached) associated to navigation and exploration.

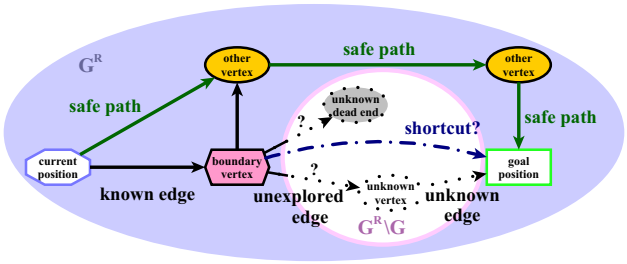

Figure 1: Typical example of Exploratory Digraph Navigation (EDN). At planning time, the navigating agent knows that the boundary vertex has yet unexplored edges, because for instance there is a door it has never opened. Traversing unexplored space may or may not reduce the navigational cost compared to using an already known and safe path to the goal. EDN decides whether or not to use exploration.

The navigating agent should take advantage of both charted and uncharted territory during path planning and navigation, whence the name "exploratory navigation". At the heart of EDN is a trade-off between safety guaranteed by exploitation of existing knowledge and short and long term efficiency obtained through exploration of uncharted territory, described for example by Argamon-Engelson et al. [1998].

1.2 Definitions

Let $\mathcal{G}^R$ be a graph made of a set of vertices $\mathcal{V}^R$ and a set of edges $\mathcal{E}^R \subset \mathcal{V}^R \times \mathcal{V}^R$. Let $\mathcal{G}$ be a subgraph of $\mathcal{G}^R$. Its set of vertices is $\mathcal{V}$ and its set of edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$. In addition to these vertices and edges, called internal vertices and explored edges respectively, $\mathcal{G}$ also has boundary vertices

*This research was supported by a DGA-MRIS scholarship

and unexplored edges. Unexplored edges are edges whose origin is in $\mathcal{V}$ but whose destination is unknown. Formally, the set of unexplored edges is $\mathcal{U} \subset \mathcal{V} \times \mathcal{V}^R$ and the set of boundary vertices is $\mathcal{B} \subset \mathcal{V}$. Boundary vertices are vertices which possess at least one unexplored outgoing edge: $\mathcal{B} = \{v \in \mathcal{V}, \exists e \in \mathcal{U}, \exists v' \in \mathcal{V}^R, e = (v, v')\}$. The boundary state of these vertices is known at planning time. From the definition follows that $\mathcal{G} = \mathcal{G}^R \Leftrightarrow \mathcal{U} = \emptyset$ or $\mathcal{B} = \emptyset$. $\mathcal{V} \cup \mathcal{B} \neq \mathcal{V}^R$ and $\mathcal{E} \cup \mathcal{U} \neq \mathcal{E}^R$ in general since some vertices and edges of $\mathcal{G}^R$ may be missing (not yet discovered) in $\mathcal{G}$.

We define Exploratory Digraph Navigation (EDN) as the problem of physically reaching a known destination $F \in \mathcal{G}$ from a known origin $O \in \mathcal{G}$ with minimal navigational cost and decent computational cost. Knowledge about $\mathcal{G}^R$ can only be acquired by physically traversing yet unexplored space, modeling it as vertices and edges and adding these to $\mathcal{G}$. $\mathcal{G}^R$ is thus the theoretical limit of $\mathcal{G}$ when all available space has been explored.

1.3 What Exploratory Digraph Navigation is not

To our knowledge, EDN has never been formulated as such. However, a number of close problems have already been formulated. Most of them are still open as of 2014.

The Canadian Traveler Problem (CTP) [Bar-Noy and Schieber, 1991; Karger and Nikolova, 2007] and the Stochastic Shortest Path Problem with Recourse (SSPPR) [Polychronopoulos and Tsitsiklis, 1996] are concerned with graphs where some edge traversal costs may be unknown (edges may not even exist) at planning time. However, the edge traversal cost is unknown but the extremities of explored and unexplored edges are still known, which is not true for unexplored edges in EDN. D* and D*Lite [Stentz, 1995; Koenig *et al.*, 2004; Koenig and Likhachev, 2005] are known approximate CTP/SSPPR solvers used for mobile robot planning. EDN can solve navigation problems on graphs $\mathcal{G}$ with unknown edge costs.

EDN is a generalization of graph exploration [Betke, 1991; Smirnov *et al.*, 1996; Awerbuch *et al.*, 1999; Panaite and Pelc, 1999; Dessmark and Pelc, 2004] since increasing the knowledge of $\mathcal{G}$ relative to $\mathcal{G}^R$ is not mandatory in EDN but any EDN solver should be able to perform exploration at least when O and F initially belong to different components of $\mathcal{G}$.

The shortest path between O and F in a partially unknown environment is provably computed by PHA* [Felner *et al.*, 2004] for future use at the expense of immediate navigational cost. EDN is a navigation problem aiming at *reaching F with minimum navigational cost*. As a side effect, a reusable path from O to F is found which hopefully exhibits a low (but not necessarily minimal) navigational cost. As opposed to the approach of Argamon-Engelson et al. [1998], EDN does not require the whole graph to be considered for each planning step and only considers exploration reducing the navigational cost to the goal (immediate reward).

1.4 Exploratory Digraph Navigation as a stochastic problem

EDN can be treated stochastically (stochastic edge or vertex traversal cost). The study of pathfinding on graphs with stochastic properties, pioneered notably by Loui [1983] is an active field of research, notably with the Markov Decision Process formulation [Puterman, 1994]. EDN can conform to the formalism of stochastic graphs if every boundary vertex is connected to every other boundary vertex or directly to F with virtual edges whose traversal costs are described by probability distributions. The resulting problem can then be solved using a utility criterion [Loui, 1983] or other techniques out of scope of this paper. However, the choice of the probability distribution for each virtual edge is not trivial notably due to destinations of unexplored edges possibly already belonging to $\mathcal{G}$ (unexplored loops). We therefore prefer a non-stochastic approach with a single number instead of a probability distribution to represent the length of virtual edges. Our approach of EDN nonetheless shares with a recent approach of the stochastic shortest path problem by Trevizan and Veloso [2014] its tunable search horizon and penalization of intermediary goal states.

1.5 Our contribution : the EDNA* algorithm

We propose the EDNA* (for Exploratory Digraph Navigation with A*) algorithm for exploratory navigation using (possibly) incomplete digraphs. EDNA* is a generalization of A* [Hart *et al.*, 1968] which considers shortcut hypotheses and the risk associated to them during A* expansion through a new degree of freedom. EDNA* can also be used for (goal-directed) exploration.

2 Describing the EDNA* algorithm

2.1 Notations

Let O_0 be the starting position of the navigating agent. O_n with $n > 0$ is the position of the navigating agent after the n^{th} run of EDNA*. O is always the starting position of the current run. We need the hypothesis that $\mathcal{G}^R$ is static (it does not change during the experiments) for the theoretical proofs. SLAM experiments not reported here show that EDNA* may also work on dynamically changing graphs though.

A path is said to be admissible in a graph if it has the lowest navigational cost amongst all possible paths. Multiple paths can be admissible. $\forall (X, Y) \in \mathcal{V}^2$, let $D(X, Y)$ be the (non-commutative) navigational cost from X to Y following an admissible path on $\mathcal{G}$. $\mathcal{G}$ can carry arbitrary per-vertex and per-edge traversal costs. If no path exists from X to Y, we take the convention that $D(X, Y) = \infty$.

EDNA*, like A*, requires an admissible but not necessarily consistent heuristic $\mathcal{H}$ [Dechter and Pearl, 1985] estimating the distance between two vertices. For a given vertex $X \in \mathcal{V}$, the value of the heuristic is written $\mathcal{H}(X, F, \mathcal{G})$, abridged to $\mathcal{H}(X)$ if unambiguous. The admissibility constraint ($\mathcal{H}$ never overestimates navigational costs) writes $\forall (X, Y) \in \mathcal{V}^2, \mathcal{H}(X, Y, \mathcal{G}) \leq D(X, Y)$. When not mentioned otherwise, the $\mathbb{R}^2$ metric space with $\mathcal{H}(X, F, \mathcal{G}) = ||F - X||$ (L_2 norm) is used since it is the most common space in path-planning applications.

It is not possible to add a shortcut discovery strategy to A* by modifying $\mathcal{H}$ for two reasons. First, taking a shortcut is risky so that $\mathcal{H}$ should be increased, which in turn is in conflict with the admissibility constraint on $\mathcal{H}$. Second, a boundary vertex can be found on an admissible path, in

which case there is no reason for penalizing it. For these reasons, EDNA* defines a second heuristic, $\mathcal{R}$, the *shortcut risk heuristic*, on boundary vertices. $\mathcal{R}$ estimates the navigational cost of a shortcut while taking into account the risk that this cost was underestimated. Given the origin O of an EDNA* planning step and $Z \in \mathcal{B}$, the value of the heuristic writes $\mathcal{R}(O, F, Z, \mathcal{G})$. We insist on the fact that $\mathcal{R}$ does not replace $\mathcal{H}$, even on boundary vertices where it is used to add information to $\mathcal{H}$. $\mathcal{R}(O, F, Z, \mathcal{G})$ is abridged to $\mathcal{R}(O, Z)$ or $\mathcal{R}(Z)$ if there is no contextual ambiguity. There is no constraint on $\mathcal{R}$ (it can be negative or even infinite) even though we will see that there is no interest in having $\mathcal{R}(O, F, Z, \mathcal{G}) < D(O, Z) + \mathcal{H}(Z, F, \mathcal{G})$. The value of $\mathcal{R}$ relative to $D(O, Z) + \mathcal{H}(Z, F, \mathcal{G})$ determines how reluctant the navigating agent is to try and discover shortcuts.

Finally, we define an EDNA* run as the combination of a planning phase, purely computational, where a path to some vertex $X \in \mathcal{V}$ (not necessarily F) is computed on $\mathcal{G}$, and of a plan execution (or navigation) phase, purely navigational, where the agent physically navigates on $\mathcal{W}$ from its current position to the position $x \in \mathcal{W}$ corresponding to $X \in \mathcal{G}^R$ while updating its current position on $\mathcal{G}$. When at X, if $X \neq F$ then $X \in \mathcal{B}$ and the agent takes the "most promising" unexplored edge simultaneously on x and X and follows it until it reaches another vertex. We embed the concept of "most promising" or "best" edge into $\mathcal{R}$ by saying that for each boundary vertex, its $\mathcal{R}$ value is computed with the best edge in mind. For instance, the best edge may be the one whose angular difference with the goal vertex F in a metric space is the best amongst all unexplored edges on X. The most promising edge can also be chosen randomly, possibly resulting in higher navigational costs on average.

2.2 Planning with EDNA* (algorithm 1)

Like A* [Hart *et al.*, 1968], EDNA* planning has two phases. The first one is a vertex expansion phase where vertices X are considered in order of increasing expected navigational cost $\delta(X) = D(O, X) + \mathcal{H}(X, F)$ and starting with O. The priority queue used to store vertices and their navigational cost $(X, \delta(X))$, also known as *open set*, is called Q. If $\mathcal{H}$ is not consistent, a list S (called *closed set*) of already encountered vertices is maintained since vertices can be encountered again with a lower δ. In the second phase, an actual path is computed starting at F or $Z \in \mathcal{B}$ and going back to O. The interested reader will report to open-source implementations of A* for details and implementation of the second phase. EDNA* has the same algorithmic complexity as A*. Planning is done on $\mathcal{G}$ and does not imply any movement in $\mathcal{W}$.

2.3 Properties of the EDNA* planning algorithm

At the end of the algorithm, EDNA* returns a path to F or to a boundary vertex Z from which exploration must take place. If no path from O to F exists on $\mathcal{G}$ and there is no boundary vertex accessible, the algorithm can't find a path. If no path exists from O to F and there is at least one boundary vertex, the algorithm will always give the shortest path leading to the boundary vertex Z with the lowest $\mathcal{R}(O, F, Z)$ encountered during expansion. It will thus perform similarly to Agent-Centered A* [Smirnov *et al.*, 1996] if

Algorithm 1 Planning with EDNA*

Input: $O, F, \mathcal{G}, \mathcal{H} : X \rightarrow \mathcal{H}(X, F, \mathcal{G})$
Input: $\mathcal{R} : Z \rightarrow \mathcal{R}(O, F, Z, \mathcal{G})$
dest $\leftarrow \emptyset$; best_distance $\leftarrow \infty$; $Q \leftarrow (O, \delta(O))$; $S \leftarrow \emptyset$
while $Q \neq \emptyset$ **do**
  pop $(X, \delta(X))$ from Q; $S \leftarrow (X, \delta(X))$
  if $\delta(X) >$ best_distance **then**
    break //EDNA* early stopping criterion
  if $X = F$ **then**
    dest $\leftarrow F$; best_distance $\leftarrow \delta(F)$
    break //traditional A* stopping criterion
  if $X \in \mathcal{B}$ **then**
    if $\mathcal{R}(O, F, X) \leq$ best_distance **then**
      dest $\leftarrow$ X; best_distance $\leftarrow \mathcal{R}(O, F, X)$
  for all neighbors T of X **do** compute $\delta(T)$
    if not ($(T \in S$ **and** $\delta(T) \geq \delta(T)_S)$ **or**
      $(T \in Q$ **and** $\delta(T) \geq \delta(T)_Q)$) **then**
      $Q \leftarrow (T, \delta(T))$
if dest $= \emptyset$ **then return** failure //no path can be found
unroll a path from dest back to O and return the path found

$\forall Z \in \mathcal{B}, \mathcal{R}(O, F, Z, \mathcal{G}) <= D(O, Z) + \mathcal{H}(Z, F, \mathcal{G})$. EDNA* is a greedy algorithm since the path choice is always optimized for the current origin O. Finally, EDNA* always expands less vertices than A* during one planning phase since vertices expanded by EDNA* would be expanded by A* but the early stopping criterion based on $\mathcal{R}$ (see algorithm) may stop the algorithm before the whole A* search space has been expanded. However, F may not be reached in one run, so that EDNA*'s search space may end up being bigger than A*'s.

2.4 Navigation with EDNA* (algorithm 2)

In order for the agent to actually move in the world, a *navigation* algorithm compatible with the EDN paradigm is required. The simple algorithm 2 performs the navigation task while being independent of the underlying planner (here EDNA*). Physically following a path or exploring an edge means that the navigating agent simultaneously moves in $\mathcal{W}$, which incurs a navigational cost, and updates its current position on $\mathcal{G}$. Vertex recognition and map update must be handled by a separate algorithm, such as those used for topological SLAM (see [Bosse *et al.*, 2004] for example). Figure 2 shows an example of EDNA* reducing the navigational cost compared to A* thanks to a shortcut.

2.5 Convergence proof and exploration variant

Let e and e_{max} be the amount of explored edges in the component around O_0 of $\mathcal{G}$ and $\mathcal{G}^R$ respectively.

Theorem 1 (Navigation success and upper bound)**.** *If $\mathcal{G}^R$ is strongly connected, EDNA* will always lead the navigating agent to $F \in \mathcal{G}^R$ from any $O_0 \in \mathcal{G}^R$. A maximum of $1 + (e_{max} - e)$ EDNA* navigation runs will be necessary.*

Corollary 1 (Complete graph exploration)**.** *If F is unreachable from O_0 because both vertices belong to a different component in $\mathcal{G}^R$, at most $(e_{max} - e)$ EDNA* navigation runs will lead to exploration of the whole component containing*

Algorithm 2 Navigation with EDNA*

Input: $\mathcal{W}, O, F, \mathcal{G}, \mathcal{H} : X \rightarrow \mathcal{H}(X, F, \mathcal{G})$
Input: $\mathcal{R} : Z \rightarrow \mathcal{R}(O, F, Z, \mathcal{G})$
$n = 0, O_n \leftarrow O$ // origin
while $O_n \neq F$ **do**
  call EDNA* Planning$(O_n, F, \mathcal{G}, \mathcal{H}, \mathcal{R})$
    $\rightarrow$ Returns path to O_{n+1}
  physically follow path from O_n to O_{n+1} //takes time
  $n \leftarrow n + 1$
  if $O_n = F$ **then break** //successfully reached F
  physically explore best edge E to reach a vertex O_{n+1}
  $n \leftarrow n + 1$, insert E in $\mathcal{G}$ //new edge
  while $(O_n \notin \mathcal{G})$ **do**
    insert O_n in $\mathcal{G}$ //new vertex
    if $O_n \notin \mathcal{B}$ **then break** //dead end
    physically explore best edge E to reach O_{n+1}
    $n \leftarrow n + 1$, insert E in $\mathcal{G}$ //new edge

O_0 before returning an error. The $1 + (e_{max} - e)$ bound is tight.

Sketch of the proof: an EDNA* run either leads to F, returns an error or registers at least one new edge in $\mathcal{G}$, so that at most $(e_{max} - e)$ runs would lead to $e = e_{max}$. As EDNA* is exactly A* when the graph is completely known ($\mathcal{G} = \mathcal{G}^R$), a last EDNA* run on $\mathcal{G}^R$ will lead to F if reachable or return an error if not reachable, but after having discovered the component containing O_0 (otherwise, there would be a boundary vertex which EDNA* would target). The bound is tight considering a chain of vertices connected in $\mathcal{G}^R$ but not in $\mathcal{G}$.

Using $F \notin \mathcal{G}^R$, corollary 1 can be used to achieve complete exploration of an unknown graph prioritizing a specific position or direction. The position of F will be used to bias shortcut discovery, but as the vertex can't be reached, the whole subgraph around O_0 will be explored, starting with the part closest to F.

2.6 Theoretical study of the $\mathcal{R}$ heuristic

$D_{\mathcal{G}^R}$ indicates that the navigational cost is computed on $\mathcal{G}^R$ instead of $\mathcal{G}$. From a theoretical point of view, five cases are particularly interesting regarding the $\mathcal{R}$ heuristic:

1. early stopping to privilege exploration:
   $\forall Z \in \mathcal{B}, \mathcal{R}(O, F, Z, \mathcal{G}) <= D(O, Z) + \mathcal{H}(Z, F, \mathcal{G})$
2. optimal case:
   $\forall Z \in \mathcal{B}, \mathcal{R}(O, F, Z, \mathcal{G}) <= D(O, Z) + \mathcal{H}(Z, F, \mathcal{G})$
   and $\forall X \in \mathcal{G}, \mathcal{H}(X, F, \mathcal{G}) = D_{\mathcal{G}^R}(X, F)$
3. optimal improvement over uninformed A* on $\mathcal{G}$:
   $\forall Z \in \mathcal{B}, \mathcal{R}(O, F, Z, \mathcal{G}) = D(O, Z) + D_{\mathcal{G}^R}(Z, F)$
4. improvement over uninformed A* on $\mathcal{G}$:
   $\forall Z \in \mathcal{B}, \mathcal{R}(O, F, Z, \mathcal{G}) > D(O, Z) + D_{\mathcal{G}^R}(Z, F)$
5. EDNA* reduced to A* on $\mathcal{G}$:
   $\forall Z \in \mathcal{B}, \mathcal{R}(O, F, Z, \mathcal{G}) \rightarrow \infty$

Case 1 sees the expansion phase stopping at the first boundary vertex encountered due to the early stopping criterion. If $\forall X \in \mathcal{G}, \mathcal{H}(X) = 0$ (Dijkstra's algorithm), with F unreachable (or no F at all) and with unitary traversal costs, this would result in the greedy algorithm *Nearest Neighbor (NN)* [Koenig and Smirnov, 1996]. NN's penalty in terms of traversed edges on a non-directed graph during exploration relative to traversed edges if the graph was known is only worst-case logarithmic in the number of edges [Megow *et al.*, 2011]. Bounds for algorithms operating on digraphs are still an open problem [Megow *et al.*, 2011] but due to its greedy nature, we expect EDNA* to reach a competitive ratio of $n-1$ with n the number of vertices in $\mathcal{G}^R$ [Förster and Wattenhofer, 2012]. EDNA*'s relocation strategy (using known edges to travel from a newly completed vertex to a boundary vertex) is however not specifically optimized for exploration tasks compared to that of Albers and Henzinger [1997] or Fleischer and Trippen [2005].

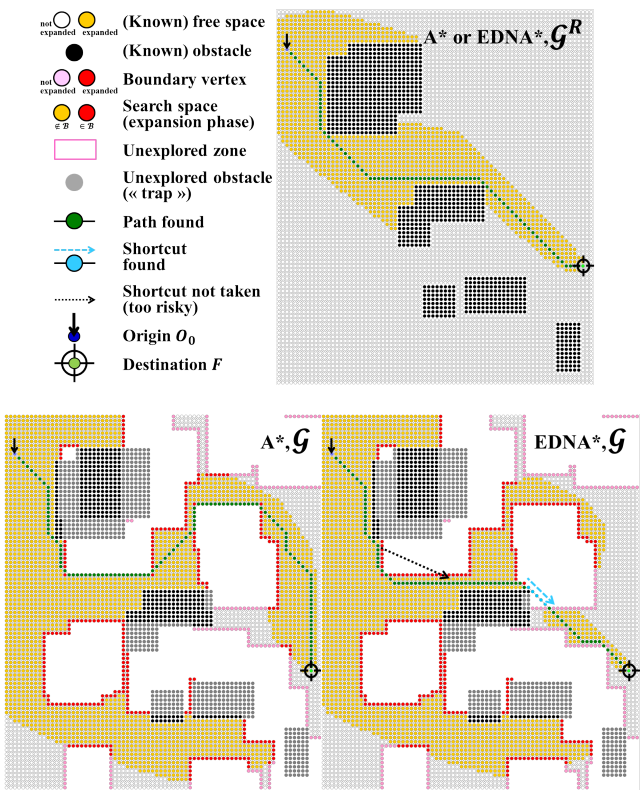

Figure 2: EDNA* shortcut discovery as described in algorithms 1 and 2. 8-way connected grid-like graphs used for a better understanding of the principles at work.

In case 2, $\mathcal{H}$ is fully informed, so that A* on $\mathcal{G}^R$ would only expand vertices directly on admissible paths (optimal search space). It is possible to prove that EDNA* on $\mathcal{G}$ with such heuristic choices will follow a path admissible on $\mathcal{G}^R$ using navigation on known parts of the path and exploration on yet to be discovered parts. Case 3 yields the same result in terms of path followed but with a bigger search space.

Case 4 results in EDNA* on $\mathcal{G}$ always overestimating the navigational cost of shortcuts (in case 5, all shortcuts are considered of infinite cost). This means that the stopping criterion can only be triggered if an exploratory path's navigational cost is actually lower than all non-exploratory ones.

However, it may not be triggered if a path is found from O to F with a cost between $D_{\mathcal{G}^\mathcal{R}}(O,F)$ and the lowest $\mathcal{R}$. This case is the one that we try to achieve in practice, the hard part being that setting an upper bound on the navigational cost of a specific shortcut is not always possible.

For a given problem which EDNA* must solve, if bounds can be given on $D(Z,F)$ for every possibly interesting boundary vertex Z, then bounds can be given on EDNA*'s suboptimality in terms of navigational cost: the worst case detour relative to A* on $\mathcal{G}^\mathcal{R}$ in the fourth case is the estimated $D(O,Z) + D(Z,F)$ minus the actual one.

3 Experiments

3.1 Finding a reference algorithm

Both EDNA* and D* (Lite) [Stentz, 1995; Koenig *et al.*, 2004; Koenig and Likhachev, 2005] can provide suboptimal (greedy) navigation in a static CTP/SSPPR situation, such as on grid-like graphs. D* and its variants use a free-space assumption ("unexplored space is traversable") resulting in a systematical underestimation of the navigational cost. As a consequence, transforming unexplored vertices to vertices with all edges unblocked makes EDNA* with $\mathcal{R} = 0$ follow the D*Lite path but without D*Lite's search space optimization. D* (Lite) can't work on digraphs where nothing is known about yet unexplored vertices.

It is possible to use PHA* [Felner *et al.*, 2004] for navigation by stopping the algorithm when F is reached for the first time (Felner et al. do not mention this possibility). In this situation, the path returned would be either the A* path on $\mathcal{G}$ (infinitely overestimating the length of shortcuts, obtainable with EDNA* and $\mathcal{R} \to \infty$) or that of iterated Agent-Centered A* [Smirnov *et al.*, 1996] (obtainable with EDNA* and $\mathcal{R} = 0$). Not stopping PHA* early would result in a lot of physical movements increasing the navigational cost and unnecessary for a navigation task. PHA* can't balance physical exploration of unknown parts and navigation on known parts even though it ends up using both.

More generally, EDNA* uses the exploration risk as a degree of freedom which none of the above algorithms does. EWP [Argamon-Engelson *et al.*, 1998] considers balancing exploration and navigation, but no algorithm to do it during planning is described. As a consequence, each iteration of EWP must compute the navigational cost from the current position to any $v_1 \in \mathcal{B}$ (algorithm not explained in the article) and estimate the cost from it to any $v_2 \in \mathcal{B}$ which is orders of magnitude more compute-intensive than our approach. When using exploration, EWP navigates to the best v_1 from where it must reach the best v_2 using exploration even though another $v \in \mathcal{V}$ is reached while going from v_1 to v_2. As a consequence, the navigational cost from O_0 to F ends up being always higher than that of our approach in the same situation.

Our objective is to demonstrate shortcut discovery i.e. EDNA* reaching F from O_0 with a lower navigational cost than a non-exploratory strategy such as A*, whence the comparison of EDNA* to A*. Map knowledge improvement due to adding newly discovered zones to $\mathcal{G}$ is not evaluated but is expected to strongly reinforce the interest of EDNA* since future traversals can benefit from currently discovered shortcuts, dead ends and loops.

3.2 A simple choice of $\mathcal{R}$ for experiments

We chose to use a risk heuristic $\mathcal{R}(O,F,Z,\mathcal{G}) = D(O,Z) + \alpha\,(1 - \beta.cos(\theta))\,\mathcal{H}(Z,F,\mathcal{G})$ in an Euclidean space with the L_2 norm as $\mathcal{H}$. θ is the angle between an edge going out from Z and the vector from Z to F, β is a parameter to describe an angular penalty and α is a parameter to describe the total penalty in the sense that if $\beta = 0$, α describes by how much a shortcut's navigational cost is overestimated relative to $\mathcal{H}(Z,F,\mathcal{G})$. $(\alpha \leq 1, \beta = 0)$ for instance gives the first case of the previous section. This proposal of heuristic is inspired by the way humans actually find shortcuts in an unknown environment (possible shortcuts are edges which go in the right direction and would potentially save a lot of time or distance). For a high enough risk factor α, $\mathcal{R}$ should fall in case 4 of the previous section. It should be noted that $\mathcal{R}$ does not take advantage of the planarity of a graph.

3.3 Benchmark protocol

Let $\mathcal{N}_{A*}$ and $\mathcal{N}_{EDNA*}$ be the navigational cost from O_0 to F using A* and EDNA* on a graph $\mathcal{G}$. The navigational cost change from A* to EDNA* on $\mathcal{G}$ is $\mathcal{N}_\mathcal{G} = \frac{\mathcal{N}_{A*} - \mathcal{N}_{EDNA*}}{\mathcal{N}_{A*}}$. A ± 0.1 change means that if the cost following the A* path is 100 units, the one following the EDNA* path is 100 ∓ 10. Similarly, a computational cost change $C_\mathcal{G}$ is computed using the number of vertices visited during the expansion phase of the algorithms which reflects execution time.

$n_{\mathcal{G}^\mathcal{R}} = 200$ random planar digraphs $\mathcal{G}^R$ with about 9 000 vertices and 12 500 dual-way edges each are generated, with square-shaped possibly-overlapping (SSPO) obstacles of varying size. Generating graphs allows control over obstacle shapes and obstacle density. $\mathcal{G}^R$ intends to mimic road networks or Generalized Voronoi Graphs [Choset and Nagatani, 2001] while being more labyrinthine and misleading in order to test the worst-case behavior of EDNA*. $\mathcal{G}$ is obtained by removing a fraction ϕ of the edges of $\mathcal{G}^R$ to create SSPO unexplored zones. In each *experiment attempt*, $\mathcal{G}$ is created from $\mathcal{G}^R$ and O_0 and F are randomly chosen. Due to the edge removal procedure, O_0 and F may belong to different components of $\mathcal{G}$, leading to A* being unable to find a path but EDNA* still finding one. Such cases represent more than 50% and up to 95% (high ϕ) of all experiment attempts. They are eliminated as giving EDNA* an infinite navigational advantage over A*, thus strongly favoring A* with respect to EDNA*. In these cases, a new experiment attempt is done until a *successful experiment* occurs, where A* finds a path from O_0 to F. Successful experiments are accumulated until an *informative experiment* occurs, where $\mathcal{N}_\mathcal{G} \neq 0$ ($\mathcal{N}_\mathcal{G} = 0$ means $\mathcal{R}$ is not used). There are n_s successful experiments including the last one which is informative. On each $\mathcal{G}^R$, 10 000 informative experiments are done, uniformly sampling 100 values in $[1;9]$ for α and 100 values in $[10\%;50\%]$ for ϕ.

The estimators used for each (α,ϕ) to retrieve the average navigational and computational cost changes plotted on figures 3 and 4 are $\bar{\mathcal{N}} = \frac{1}{n_{\mathcal{G}^R}} \cdot \sum_{\mathcal{G}^R} \left(\frac{1}{n_s} \cdot \sum_\mathcal{G} \mathcal{N}_\mathcal{G} \right)$ and $\bar{C} = \frac{1}{n_{\mathcal{G}^R}} \cdot \sum_{\mathcal{G}^R} \left(\frac{1}{n_s} \cdot \sum_\mathcal{G} C_\mathcal{G} \right)$. Both estimators average on

all $\mathcal{G}^R$ the (poor) average on all successful experiments on a given $\mathcal{G}^R$. Arithmetically averaging changes is intuitive but highly unfavorable to EDNA* since a 0.9 degradation (navigational cost multiplied by 1.9) balances a 0.9 improvement (navigational cost divided by 10) of EDNA* over A*.

$\beta \sim 0.25$ was experimentally observed to give EDNA* the best navigational improvements over A*. Tests were also carried on 8-way connected grid-like graphs like that of Figure 2 for which $\beta \sim 0.125$ gave better results.

3.4 Results and discussion

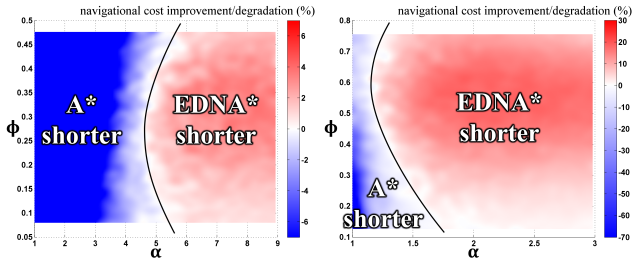

Figure 3: Average navigational cost ($\bar{\mathcal{N}}$) changes for successful experiments, random (left) and grid-like (right) graphs.

EDNA* vs A* navigational cost ($\bar{\mathcal{N}}$) comparison

Figure 3, left shows EDNA* with $\alpha > 5$ always reducing $\bar{\mathcal{N}}$ compared to A* in average, demonstrating successful shortcut discovery. With $\alpha \sim 8$, informative experiments show navigational costs divided by up to 14 when switching from A* to EDNA*. With $\alpha > 8$, EDNA*'s success decreases with α because most experiments are not informative (early stopping is not triggered). The maximum performance gain of EDNA* over A* on random graphs is 3.9% with our experimental protocol unfavorable to EDNA* (since experimental results approximately follow a Gaussian distribution, the 95% confidence interval is [0.5%; 7.3%]). We observed that selection of the sole experiments where A* reaches F causes the A* shorter/EDNA* shorter limit to be biased towards high α when ϕ increases, thus further disadvantaging EDNA*. Results on grid-like graphs (Figure 3, right) show up to 20% improvement of navigational cost of EDNA* over A*.

EDNA* vs A* computational cost ($\bar{\mathcal{C}}$) comparison

Figure (4, right) shows an increase from A* to EDNA* of $\bar{\mathcal{C}}$ when multiple EDNA* runs are required. This penalty for EDNA*, which exceeds 100% with low α and high ϕ is most of the time located between 0 and 100%, especially in the high α zone (which is the interesting zone as seen in the paragraph on $\bar{\mathcal{N}}$). Figure (4, left) shows the same result as Figure (4, right) but on a single EDNA* run where the stopping criterion allows EDNA* to always expands less space than A*. With a high ϕ and a low α, this difference reaches 100% (EDNA* explores close to no vertices). With a low ϕ and a high α, each run of EDNA* performs only marginally better than A* but there are less runs, so that EDNA*'s and A*'s performances are equivalent (with $\phi = 0$ or $\alpha \to \infty$, EDNA* would fall back to A*). EDN's focus is on navigational cost but the figures show a computational cost comparable to that of A*, which has been proved optimal in a non-exploratory situation [Hart *et al.*, 1968].

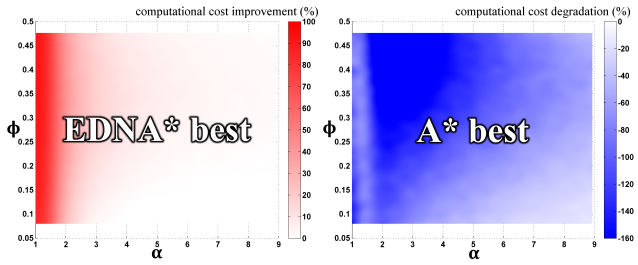

Figure 4: Average computational cost ($\bar{\mathcal{C}}$) changes for successful experiments on random graphs, up to the first boundary vertex (left) and up to F (right).

4 Conclusion

We present the Exploratory Digraph Navigation problem and the EDNA* algorithm to solve it. With a $\mathcal{R}$ heuristic sufficiently overestimating shortcut lengths, the agent using EDNA* will on average move from 3.9% to more than 20% less than using A* to reach F, with best EDNA* runs dividing the navigational cost by up to 14. Moving less leads for instance to energy savings, especially if the path is intended to be taken multiple times like in routing problems. Shortcuts, dead ends and loops found during exploration phases are stored in $\mathcal{G}$. The impact of this knowledge reinforcement has yet to be evaluated. Minimizing computational cost is not the main focus of EDN problems where traversal is much slower than planning. Nonetheless, even if EDNA* tends to expand about twice as many vertices as A* during the whole navigation, it expands less per planning step which is interesting for real-time systems where single computations must terminate quickly. Moreover, if O and F are not connected on $\mathcal{G}$, A* fails and an exploration algorithm has to be used. EDNA* on the contrary does not fail and automatically resorts to exploration, thus combining the navigation (A*) and the exploration algorithm. The relative amount of navigation and exploration can be tuned through $\mathcal{R}$. Additional data such as metric or semantic properties or planarity could be integrated into $\mathcal{H}$ and $\mathcal{R}$ for better results.

When using an exploratory strategy, it happens that the navigating agent starts to thoroughly explore a dead end. Ideas to deal with this situation have already been formulated for exploration algorithms [Smirnov *et al.*, 1996]. We are working on a non-stubbornness criterion which would detect this situation on the current vertex X by comparing $D(O_0, X) + \mathcal{H}(X, F)$ to $\mathcal{H}(O_0, F)$. Triggering this criterion would lead to an increase of $\mathcal{R}$, causing EDNA* to try finding a safe path instead of hoping for a non-existing shortcut. Another possible solution to the issue would be to give the navigating agent an energy budget which would decrease with each detour while $\mathcal{R}$ would increase.

References

[Albers and Henzinger, 1997] Susanne Albers and Monika Rauch Henzinger. Exploring unknown environments. In *SIAM Journal on Computing*, pages 416–425, 1997.

[Argamon-Engelson *et al.*, 1998] Shlomo Argamon-Engelson, Sarit Kraus, and Sigalit Sina. Utility-based on-line exploration for repeated navigation in an embedded graph. *Artificial Intelligence*, 101(12):267 – 284, 1998.

[Awerbuch *et al.*, 1999] Baruch Awerbuch, Margrit Betke, Ronald L. Rivest, and Mona Singh. Piecemeal graph exploration by a mobile robot. *Information and Computation*, 152(2):155 – 172, 1999.

[Bar-Noy and Schieber, 1991] Amotz Bar-Noy and Baruch Schieber. The Canadian Traveller Problem. In *Proceedings of the Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '91, pages 261–270, Philadelphia, PA, USA, 1991. Society for Industrial and Applied Mathematics.

[Betke, 1991] Margrit Betke. Algorithms for exploring an unknown graph. Master's thesis, MIT, Laboratory for Computer Science, 1991.

[Bosse *et al.*, 2004] Michael Bosse, Paul Newman, John Leonard, and Seth Teller. Simultaneous localization and map building in large-scale cyclic environments using the atlas framework. *International Journal of Robotics Research*, 23(12):1113–1139, December 2004.

[Choset and Nagatani, 2001] H. Choset and Keiji Nagatani. Topological Simultaneous Localization And Mapping (SLAM): toward exact localization without explicit localization. *Robotics and Automation, IEEE Transactions on*, 17(2):125–137, 2001.

[Dechter and Pearl, 1985] Rina Dechter and Judea Pearl. Generalized best-first search strategies and the optimality of A*. *J. ACM*, 32(3):505–536, July 1985.

[Dessmark and Pelc, 2004] Anders Dessmark and Andrzej Pelc. Optimal graph exploration without good maps. *Theoretical Computer Science*, 326(13):343 – 362, 2004.

[Elfes, 1989] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.

[Felner *et al.*, 2004] A. Felner, R. Stern, A. Ben-Yair, S. Kraus, and N. Netanyahu. PHA*: Finding the shortest path with A* in an unknown physical environment. *Journal of Artificial Intelligence Research*, 21:631–670, 2004.

[Fleischer, 2005] Rudolf Fleischer. Exploring an unknown graph efficiently. In *Proc. 13th Annu. European Sympos. Algorithms*, pages 11–22. Springer-Verlag, 2005.

[Förster and Wattenhofer, 2012] Klaus-Tycho Förster and Roger Wattenhofer. Directed graph exploration. In *Principles of Distributed Systems*, volume 7702 of *Lecture Notes in Computer Science*, pages 151–165. Springer Berlin Heidelberg, 2012.

[Hart *et al.*, 1968] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, July 1968.

[Karger and Nikolova, 2007] David Karger and Evdokia Nikolova. Exact algorithms for the Canadian Traveller Problem on paths and trees. Technical report, MIT, Artificial Intelligence Laboratory, www.csail.mit.edu, July 2007.

[Koenig and Likhachev, 2005] S. Koenig and M. Likhachev. Fast replanning for navigation in unknown terrain. *Robotics, IEEE Transactions on*, 21(3):354–363, June 2005.

[Koenig and Smirnov, 1996] Sven Koenig and Yury Smirnov. Graph learning with a nearest neighbor approach. In *Proceedings of the Ninth Annual Conference on Computational Learning Theory*, COLT '96, pages 19–28, New York, NY, USA, 1996. ACM.

[Koenig *et al.*, 2004] Sven Koenig, Maxim Likhachev, and David Furcy. Lifelong Planning A*. *Artificial Intelligence*, 155(12):93 – 146, 2004.

[Loui, 1983] Ronald Prescott Loui. Optimal paths in graphs with stochastic or multidimensional weights. *Commun. ACM*, 26(9):670–676, september 1983.

[Megow *et al.*, 2011] Nicole Megow, Kurt Mehlhorn, and Pascal Schweitzer. Online graph exploration: New results on old and new algorithms. In *Automata, Languages and Programming*, volume 6756 of *Lecture Notes in Computer Science*, pages 478–489. Springer Berlin Heidelberg, 2011.

[Panaite and Pelc, 1999] Petrior Panaite and Andrzej Pelc. Exploring unknown undirected graphs. *Journal of Algorithms*, 33(2):281 – 295, 1999.

[Polychronopoulos and Tsitsiklis, 1996] George H. Polychronopoulos and John N. Tsitsiklis. Stochastic Shortest Path Problems with Recourse. *Networks*, 27(2):133–143, 1996.

[Puterman, 1994] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

[Smirnov *et al.*, 1996] Yury Smirnov, Sven Koenig, Manuela M. Veloso, and Reid G. Simmons. Efficient goal-directed exploration. In *Proceedings of the National Conference on AI*, pages 292–297. AAAI Press, 1996.

[Stentz, 1995] Anthony Stentz. The focussed D* algorithm for real-time replanning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1995.

[Trevizan and Veloso, 2014] Felipe W. Trevizan and Manuela M. Veloso. Depth-based Short-Sighted Stochastic Shortest Path Problems. *Artificial Intelligence*, 216:179–205, November 2014.