Explaining plans at scale: scalable path planning explanations in navigation meshes using inverse optimization

Martim Brandão[*] and **Daniele Magazzeni**
King's College London
{martim.brandao, daniele.magazzeni}@kcl.ac.uk

Abstract

In this paper we propose methods that provide explanations for path plans, in particular those that answer questions of the type "why is path A optimal, rather than path B which I expected?". In line with other work in eXplainable AI Planning (XAIP), such explanations could help users better understand the outputs of path planning methods, as well as help debug or iterate the design of planners and maps. By specializing the explanation methods to path planning, using optimization-based inverse-shortest-paths formulations, we obtain drastic computation time improvements relative to general XAIP methods, especially as the length of the explanations increases. One of the claims of this paper is that such specialization might be required for explanation methods to scale and therefore come closer to real-world usability. We propose and evaluate the methods on large-scale navigation meshes, which are representations for path planning heavily used in the computer game industry and robotics.

1 Introduction

Path planners are traditionally not self-explanatory about their output. The result of successfully running a path planner is only a path, and so users may have problems understanding why a path is different from what was expected. Developers themselves may also have trouble debugging a large graph over which planning is run, for example in case they want a certain path to become optimal in the next version of the model.

Domain-independent methods for eXplainable AI Planning (XAIP), such as the Model Reconciliation work of [Chakraborti *et al.*, 2017], are theoretically also applicable to path planning. However, as we will argue in this paper, they currently lack the heuristics and domain-knowledge that would allow them to scale in large-scale problems in path planning. Computation speed is a requirement for interactive interfaces, for example when planners are used in human-in-the-loop designs, when safety-critical robots are deployed in dynamic environments, or when a speedy or interactive investigation of planner behavior is desirable.

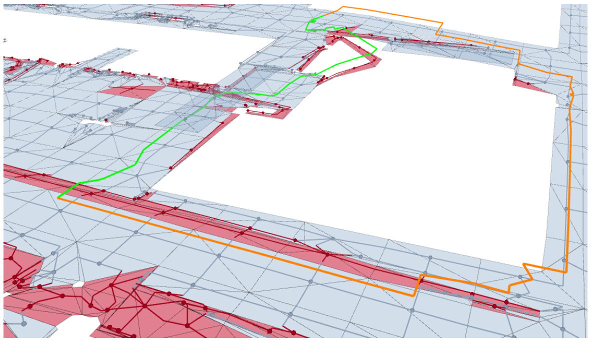


Figure 1: A user asks why the shortest path on a navigation mesh is the path in orange, and not the one in green (provided by the user). Each polygon is represented by a vertex in a graph (small spheres) and is connected by edges to other walkable vertices. Each NavMesh polygon can be of different terrain-types: "easy" (blue) or "hard" (red) that correspond to a different cost-per-distance.

In this paper we explore the connection between *path planning explanations* and the *inverse shortest path problem* [Burton and Toint, 1992; Ahuja and Orlin, 2001] on a specific set of types of explanation. We show that this framing allows to formulate explanation problems as numerical optimization, and thus leverage the speed of commercial optimization solvers. We focus on path planning problems in *navigation meshes* (NavMeshes) [Van Toll *et al.*, 2016; Mononen, 2014] which are popular representations of 2D and 3D environments used in real-world computer game products [MobyGames, 2019] and robotics applications [Brandao *et al.*, 2020].

In summary our contributions are the following:

- We show that explanations of path plans of the type "why is the shortest path A, rather than B?" can be formulated as inverse-shortest path problems;
- We propose two optimization-based formulations to provide explanations in NavMeshes;
- We show that solving these problems with commercial optimization solvers leads to fast and scalable computation of explanations compared to domain-independent XAIP methods.

[*] Contact Author

We evaluate the method in realistic large-scale path planning problems inspired by real-world robotics and computer-game domains.

One of the main take-away messages of the paper is that, while current explainable AI planning methods are general and thus widely applicable, they do not scale. Specializing explanation methods to specific planning domains can lead to drastic computation improvements in domains such as path planning, and thus contribute to real-world use.

2 Related work

This work is related to recent research in "Model Reconciliation" [Chakraborti *et al.*, 2017] for general AI Planning tasks, which involves computing differences between the user's and planner's model of the problem. This could be, for example, a minimum set of changes to apply to the user's model so that the planner's path becomes optimal in that model. In [Chakraborti *et al.*, 2017], these differences are computed by searching directly in the space of models, i.e. building a tree that starts at the human's model of the problem and adds or removes a precondition, an effect, or initial state entry at each node, until arriving at a model where the planner's path is optimal. This process is complete but time-consuming, which is why in this paper we explore the use of optimization tools for speeding up the process in path planning. Therefore, we are here providing a *specialization* of model reconciliation methods to path planning. Our methods are also specialized in a different sense. We assume a particular configuration of the reconciliation process—that the human's model is not known, and we wish to obtain the minimum changes from the planner's model that lead to the shortest-path being the human's desired one.

Work in XAIP has also dealt with abstracted lower-dimensional model search spaces, both for explanations of model differences [Sreedharan *et al.*, 2018] and explanations of planner failure [Sreedharan *et al.*, 2019]. Similar abstractions have also been explored in the motion planning literature [Brandao and Havoutis, 2020], however with the goal of speeding up motion planning itself. In this paper we provide two alternative explanation methods specialized for path planning in navigation meshes, one of which is low-dimensional—related to the cost-per-distance of each terrain of a navigation mesh.

The "explicability" of a plan is a concept introduced by [Zhang *et al.*, 2017; Sreedharan *et al.*, 2017], and relates to the degree to which a planner's plan is close to the human's desired plan [Chakraborti *et al.*, 2019]. Here we also evaluate the explicability of produced plans, using a metric that is more suitable to path planning—Fréchet distance [Alt and Godau, 1995], which computes the similarity of curves in any dimension.

Another related body of research is on explanations for Multi-Agent Path Finding [Almagor and Lahijanian, 2020], where the goal is to obtain an intuitive explanation for why the agents' paths are non-colliding; and on explanations for continuous-space motion planning [Hauser, 2014; Kwon *et al.*, 2018], which give reasons for the failure to obtain a path. In this paper, however, we focus on a different type of explanation—explanation of the optimality of a path—that answers questions of the type "why is path A optimal, rather than B?". We show that such explanations are related to the problem of inverse shortest paths [Burton and Toint, 1992; Zhang and Pavone, 2016], which looks for a minimal change to graph weights that leads to a desired path being optimal.

Our work targets a specific type of path planning problems—planning on *navigation meshes* [Mononen, 2014; Van Toll *et al.*, 2016; Brandao *et al.*, 2020]. We focus on this particular representation of environments since it is widely and heavily used in computer games [MobyGames, 2019] and robotics [Brandao *et al.*, 2020], therefore increasing the potential for real-world impact and usefulness. Additionally, the structure of the path planning problem in NavMeshes is interesting due to the existence of terrain types, which make the inverse shortest paths problem combinatorial—thus requiring the use of Mixed-Integer Linear Programming (MILP) solvers.

3 Background

3.1 Shortest path

Let $G = (V, E, W)$ be a (directed) graph with vertices $v_i \in V$, edges $e_j \in E$, and a weight $w_j \in W$ associated with each edge. An edge e_j connects $v_{s(j)} \in V$ to $v_{t(j)} \in V$, where $s(j)$ is the index of the origin vertex, and $t(j)$ is the index of the target vertex of the edge. For convenience, weights can also be written as $w(e_j)$. The shortest path p^* is a sequence of consecutive edges $p^* = (e_1, ..., e_n)$, of any possible length n, starting at v_{start} and ending at v_{goal}, that minimizes $\sum_{k=1}^{n} w(e_k)$.

The problem can also be formulated as a linear program (LP) [Ahuja *et al.*, 1993]:

$$\min_{x \in \mathbb{R}_{0+}^{|V|}} \; w^{\mathsf{T}} x, \quad \text{s.t.} \quad Ax = b, \tag{1}$$

where x_j is equal to 1 if e_j belongs to the shortest path, and 0 otherwise. A_{ij} is equal to 1 if $s(j) = i$ (i.e. e_j leaves v_i), -1 if $t(j) = i$, and 0 otherwise. Finally, b_i is equal to 1 if $v_i = v_{\text{start}}$, -1 if $v_i = v_{\text{goal}}$, and 0 otherwise. The intuition behind this LP is that we pick a set of edges with minimum cost that connect the origin and target nodes (i.e. all nodes except origin and target have the same number of input and output edges). The LP is integral, meaning that an optimal solution will have x with components equal to either 0 or 1 [Ahuja *et al.*, 1993].

3.2 Inverse shortest path

The inverse of problem (1) is when we wish to obtain a new weight vector w' that leads to a desired shortest-path p' corresponding to a desired x', with the goal of w' being as close as possible to w. This is also an LP [Ahuja and Orlin, 2001] which can be written as:

$$\min_{w',\pi,\lambda} \quad ||w' - w||_1 \tag{2a}$$

$$\text{s.t.} \quad \textstyle\sum_i A_{ij}\pi_i = w'_j \quad \forall_{j:x'_j=1} \tag{2b}$$

$$\textstyle\sum_i A_{ij}\pi_i + \lambda_j = w'_j \quad \forall_{j:x'_j=0} \tag{2c}$$

$$\pi \in \mathbb{R}^{|V|} \tag{2d}$$

$$\lambda \in \mathbb{R}^{|E|} \tag{2e}$$

$$\lambda_j \geq 0 \quad \forall_{j:x'_j=0} \tag{2f}$$

$$w' \in \mathbb{R}_+^{|E|}. \tag{2g}$$

3.3 Navigation meshes

In navigation meshes, graph vertices represent physical locations that lie either on the center or the border of walkable polygons. Edges in turn represent the possibility of navigating between two adjacent locations. Navigation mesh graphs are of a particular structure (see Figure 1). Each vertex v_i is associated with a geometric position $z_i \in \mathbb{R}^3$, and each edge e_j is associated with an Euclidean distance $d_j = ||z_{t(j)} - z_{s(j)}||$. Additionally, each vertex v_i is either at the center of a navigation-mesh polygon, or at the center of the intersection between two polygons—usually called a "portal". The set of non-portal vertices is $V^r \subset V$. Each polygon in the NavMesh (i.e. each non-portal vertex) is associated with a terrain-type $k = \{1, ..., K\}$, and each terrain type is associated with a cost-of-transport $c_k \in R^+$ (i.e. a cost per distance travelled). This means that, for example, moving on an edge e_j which lies on a polygon of terrain-type k_{example} requires a weight $w_j = d_j c_{k_{\text{example}}}$.

We can represent the terrain-types of all non-portal vertices using $l \in \{0,1\}^{K|V^r|}$, where $l_{k,i}$ is equal to 1 if vertex v_i is of terrain-type k, and 0 if it is not. Let $P(v_i) = \{0, 1\}$ indicate whether v_i is a portal or not, and $r(j) = \{k \in \{s(j), t(j)\} : P(v_k) = 0\}$ be a function that maps an edge index to the index of its non-portal vertex. Weights in NavMesh graphs are thus of the following form:

$$w_j = \textstyle\sum_{k=1}^{K} d_j c_k l_{k,r(j)}. \tag{3}$$

4 Method

We wish to provide explanations for questions of the type "why is path p^* optimal, rather than p'?". Following the work of [Wachter *et al.*, 2017], we compute actionable counterfactual explanations for such questions—which means providing an alternative possible world G' that is as close as possible to G but in which p' would be optimal. In other words, a user provides a path p' that they desire or that they had expected, and the method explains the map changes that would have to take place for that path to be optimal.

Given the structure of NavMeshes, we can either compute new terrain-type assignments l' or new values of the costs-per-distance c'.

4.1 Terrain-type explanation

Obtaining a new terrain-type assignment $l' \in \{0,1\}^{K|V^r|}$ that satisfies a desired shortest-path is an inverse shortest path problem similar to (2), which now becomes a Mixed-Integer Linear Programming (MILP) problem we call *Terrain-MILP*.

$$\boxed{\textit{Terrain-MILP:}}$$

$$\min_{l',\pi,\lambda} \quad ||l' - l||_1 \tag{4a}$$

$$\text{s.t.} \quad \textstyle\sum_i A_{ij}\pi_i = \sum_k d_j c_k l'_{k,r(j)} \quad \forall_{j:x'_j=1} \tag{4b}$$

$$\textstyle\sum_i A_{ij}\pi_i + \lambda_j = \sum_k d_j c_k l'_{k,r(j)} \quad \forall_{j:x'_j=0} \tag{4c}$$

$$\pi \in \mathbb{R}^{|V|} \tag{4d}$$

$$\lambda \in \mathbb{R}^{|E|} \tag{4e}$$

$$\lambda_j \geq 0 \quad \forall_{j:x'_j=0} \tag{4f}$$

$$l' \in \{0,1\}^{K|V^r|} \tag{4g}$$

$$\textstyle\sum_k l'_{k,i} = 1 \quad \forall_i, \tag{4h}$$

This formulation minimizes the number of nodes with terrain changes in (4a). Lines (4b-4f) are inverse shortest path constraints that correspond to those in (2b-2f), but where edge weights are expressed as a function of node terrain-types l'. Lines (4g-4h) enforce a single terrain-type per node.

4.2 Costs-of-transport explanation

The other kind of explanation involves finding out counterfactual values for the per-terrain cost-of-transport vector $c \in \mathbb{R}_+^K$ that would satisfy the desired path. This involves solving the following problem:

$$\boxed{\textit{COT-MILP:}}$$

$$\min_{c',\pi,\lambda} \quad ||c' - c||_1 \tag{5a}$$

$$\text{s.t.} \quad \textstyle\sum_i A_{ij}\pi_i = \sum_k d_j c'_k l_{k,r(j)} \quad \forall_{j:x'_j=1} \tag{5b}$$

$$\textstyle\sum_i A_{ij}\pi_i + \lambda_j = \sum_k d_j c'_k l_{k,r(j)} \quad \forall_{j:x'_j=0} \tag{5c}$$

$$\pi \in \mathbb{R}^{|V|} \tag{5d}$$

$$\lambda \in \mathbb{R}^{|E|} \tag{5e}$$

$$\lambda_j \geq 0 \quad \forall_{j:x'_j=0} \tag{5f}$$

$$c' \in \mathbb{R}_+^K, \tag{5g}$$

where constraints are the same as in Terrain-MILP, but the variable is now c'. Note that, as we will show in the Experiments section, this problem will often be infeasible due to the low flexibility of the search space (i.e. c').

4.3 Metrics

We will use two metrics to characterize explanations:

- Explanation length: this is used as a proxy of complexity. We compute it as the number of terrain changes $||l' - l||_1/2$ in terrain-type explanations, and as $||c' - c||_1$ in cost-of-transport explanations.
- Explicability distance: this measures the degree to which an explanation actually leads to the optimal path being the desired. We compute this as the Fréchet distance between the the two paths $D(p', p^*_{\text{new}})$, where p^*_{new} is the shortest path on the new graph G'.

5 Experiments

5.1 Experimental setup

For the experiments in this paper we used the map of a large building, of which the 3D model is publicly available—the Barcelona Robotics Laboratory[1]. The 3D model consists of a campus of multiple buildings and outside corridors, stairs, lamp posts, etc. The NavMesh of this model is shown in Figure 1. There are multiple ways to cross the campus (around through the left, right, or by crossing the patio and climbing the stairs) and the optimal path depends on both distance and the cost of travelling on stairs vs flat ground. The hypothetical use case is of a robot that moves throughout campus running errands, delivering packages or guiding visitors, similar to [Rosenthal and Veloso, 2012]. This could be a wheeled mobile robot with tracks, or a legged robot with stair-climbing functionality such as in [Brandao *et al.*, 2020].

We ran the procedure in [Brandao *et al.*, 2020] to automatically generate terrain-type assignments to the model[2]. The procedure uses measurements of local curvature to assign a terrain-type, and leads to an assignment of terrain type 1 ("easy") to flat ground and 2 ("hard") to stairs and areas that are close to walls and obstacles. We assume the cost-of-transport of the hypothetical robot to be 1 unit per meter on easy terrain and 8 units per meter on hard terrain. We use the Recast toolkit [Mononen, 2014] to generate a navigation mesh from this model and run planning and explanation methods on the underlying graph. The graph is of size $|V| = 4935$ and $|E| = 6056$.

To solve Terrain-MILP and COT-MILP we use the commercial solver MOSEK[3] on the Python optimization interface cvxpy [Diamond and Boyd, 2016]. On the NavMesh used for our experiments, Terrain-MILP consists of 24675 scalar variables, 3814 integer variables and 33707 constraints; while COT-MILP has 17087 scalar variables and 26119 constraints. Measured computation times in this paper include problem construction time, which is responsible for most of the computation (around 60 seconds on average).

5.2 Example explanations

Figure 2 shows three examples of explanations obtained by our methods. We will call these examples Problem 1-3.

In Problem 1 (Figure 2a), the shortest path between two points in the map is shown in orange. It involves going around a stair-area ("hard" terrain, shown in red), and around a building (empty space in white) through the right until the goal (in the top of the image). A user then provides the green path and asks why that was not the shortest path instead. The motivation for asking this question could be a wish to understand the reasoning behind the planner (i.e. update the user's mental model of the problem). Another option could be that the user is a developer that is debugging the robot's model and behavior. The developer believes the robot should actually take this path, and therefore wishes to know the minimal changes to apply to the map in order to make sure that becomes the new preferred path.

Problem	Method	Expl. length	Dist. to B	Comp. time
Prob 1	COT-MILP	1	0	58.2
	Terrain-MILP	14	0	66.5
Prob 2	COT-MILP	2	10.1	72.2
	Terrain-MILP	34	0	69.6
Prob 3	COT-MILP	2	9.6	73.5
	Terrain-MILP	19	0	73.6

Table 1: Comparison between cost-of-transport and terrain-type explanations

Figure 2b shows the terrain-based explanation Terrain-MILP. With these new terrain assignments, the shortest path does indeed become the user's preference. As the figure shows, only polygons along the stairs would need to change terrain-type in order for the user's path to become optimal (one polygon is on the stairs on the bottom of the figure, the other polygons on the stairs near to the top). The alternative cost-of-transport-based explanation COT-MILP, shown in Figure 2c, would basically reveal that increasing the cost-of-transport of "easy" terrain to 3.15 would be enough to make the user's path optimal. This case would not involve any change of terrains. As Table 1 shows , this corresponds to a change of 1 variable (instead of the 14 terrain changes in the terrain-type explanation).

Cost-of-transport solutions are, however, not always applicable. For example in Problem 2 (Figure 2d), the user asks why the shortest path does not go around an obstacle (small white gap) through the left and then use the stairs. Our terrain-type explanation method can find a change of terrain assignments that would lead to such a shortest path (Figure 2e). However, the cost-of-transport formulation is infeasible here and we can thus only find a shortest path that is close to the desired one, but not the same. The Fréchet-distance between this path and the desired one is 10.1 meters (Table 1).

Problem 3 (Figure 2g), which was obtained by assuming a cost-of-transport of 50 for "hard" terrain, is similar to Problem 2 in that the terrain-type explanation method finds a 0-distance solution, but the cost-of-transport explanation is infeasible (distance 9.6m).

Finally, even if our terrain-type-explanation method involves solving a MILP, its solving time was similar to that of the cost-of-transport method, and across all problems—around 60-70s as shown in Table 1.

5.3 Scalability

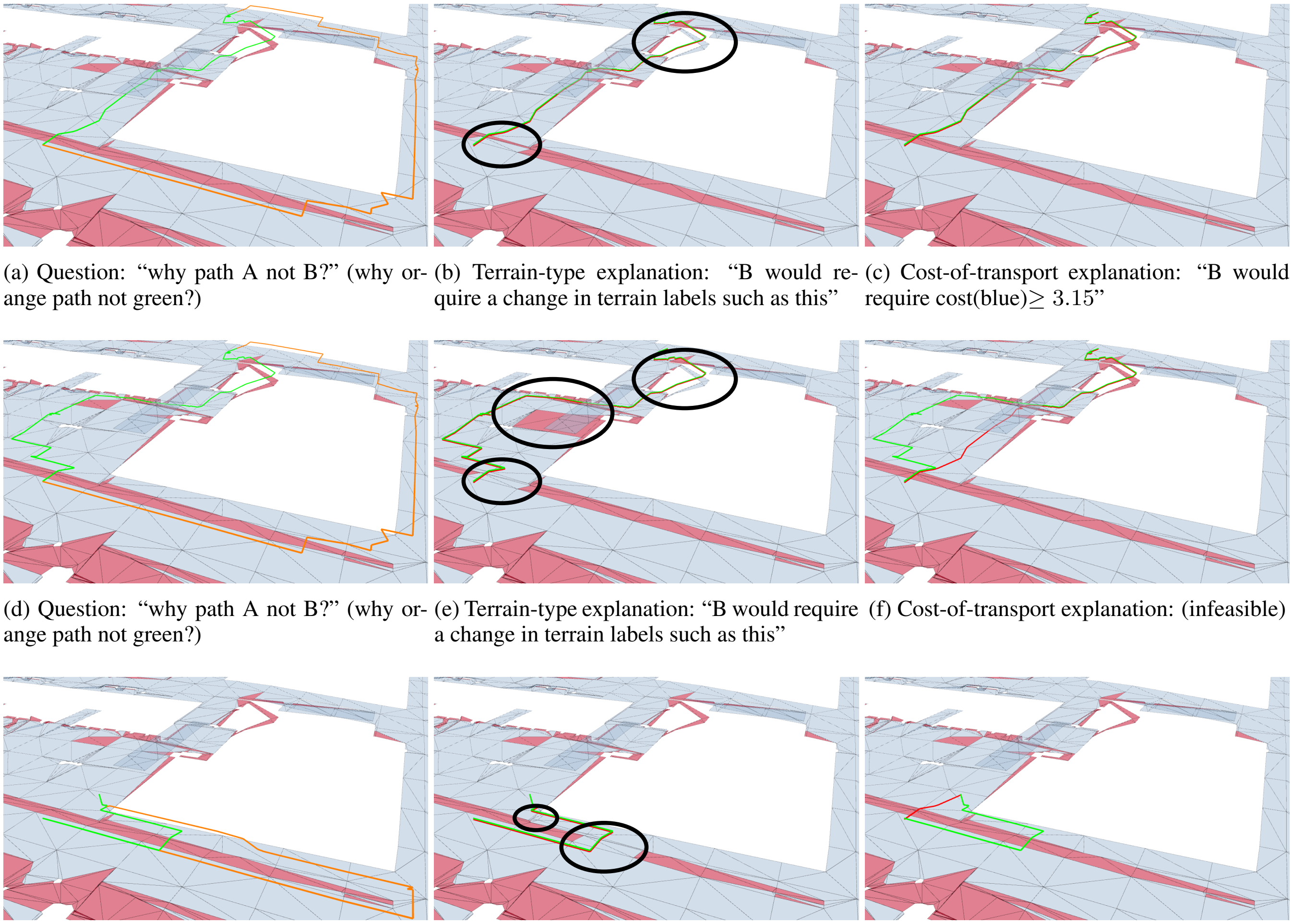
We will now see that the computation times of Terrain-MILP, shown in Table 1, are considerably faster than domain-independent model search methods used in Model Reconciliation [Chakraborti *et al.*, 2017], especially as the explanation length increases.

To compare our method to general Model Reconciliation methods in XAIP, we implemented A* model search as in

[1] http://www.iri.upc.edu/research/webprojects/pau/datasets/BRL/

[2] https://github.com/ori-drs/gaitmesh

[3] https://www.mosek.com/



(a) Question: "why path A not B?" (why orange path not green?)

(b) Terrain-type explanation: "B would require a change in terrain labels such as this"

(c) Cost-of-transport explanation: "B would require cost(blue)≥ 3.15"

(d) Question: "why path A not B?" (why orange path not green?)

(e) Terrain-type explanation: "B would require a change in terrain labels such as this"

(f) Cost-of-transport explanation: (infeasible)

(g) Question: "why path A not B?" (why orange path not green?)

(h) Terrain-type explanation: "B would require a change in terrain labels such as this"

(i) Cost-of-transport explanation: (infeasible)

Figure 2: Three example problems (one per line) where a user asks "why is path A the shortest, and not path B?" by providing the expected/desired path B. A is shown in orange, B in green. The second column shows our terrain-type explanations (terrain changes highlighted), while the third column shows cost-of-transport explanations. The red paths shown in the center and right columns are the shortest-paths in the new models, which might not be equal to the desired path in green (indicating failure to solve the optimization problem exactly).

[Chakraborti *et al.*, 2017]. We initially used the official implementation[4]. However, the runtime was too slow for even the smallest problems due to the use of AI Planning solvers (Fast Downward and VAL). Therefore, to be fair to the algorithm we replaced the solution routines by Dijkstra shortest-path computations, which run in a fraction of the computation time. The rest of the implementation used was as in [Chakraborti *et al.*, 2017]—each state in the search tree is a full model (i.e. a NavMesh graph), and each expansion involves switching one terrain label for another. The process continues until we arrive at the goal state (i.e. until the desired path is the shortest). The same heuristic state expansion as proposed in [Chakraborti *et al.*, 2017] is also used for speed. Note that this assumes a particular Model Reconciliation process, where the human's mental model is not necessarily known, and we want to find the model that is closest to ground-truth but where the shortest path is the user's desired one.

When A* model search succeeds to find an explanation, this explanation is the same as what is obtained by our Terrain-MILP method. However, the combinatorial nature of the problem leads to an exponential increase in computation time with explanation length, as shown in Figure 3. Each point in the figure corresponds to a different problem (Problem 1 and two variations with different values of c; Problems 1 and 3 were excluded since they took longer than 2 hours to compute). As the figure shows, the use of commercial MILP-specialized solvers leads to a constant solve time, compared to a much higher time for A* model search.

These results show that in order for explanation methods to scale, it might be required to specialize the algorithm to specific problem instances—instead of using general explanation

[4] https://github.com/TathagataChakraborti/mmp/

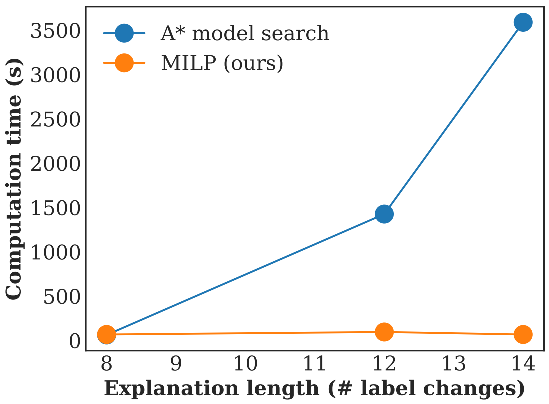


Figure 3: Scalability of the inverse shortest path formulation (ours) vs A* model search (domain independent method).

algorithms.

6 Conclusion

In this paper we introduced optimization-based methods for solving two types of explanation on navigation meshes—cost-of-transport and terrain-type explanations. Using these methods we obtain explanations for the optimality of a path on a navigation mesh, i.e. a user may provide an alternative path and ask why it is not optimal, and the methods compute minimal changes to the model that lead the user's path to become optimal. These explanations are therefore actionable counter-factual explanations in the style of [Wachter *et al.*, 2017]—they provide users with suggestions for map changes that lead expected paths to become optimal.

In this paper we showed examples of both types of explanation. We did not, however, study the effectiveness of each at improving users' understanding of the model—and this is an important future direction of research.

The terrain-type explanation method involves solving a MILP problem, and is basically a specialization of model reconciliation methods from eXplainable AI Planning (XAIP) to path planning. This specialization comes with the benefit of increased speed and scalability due to the applicability of commercial MILP solvers. We showed that with such solvers we can compute explanations for path planning at approximately constant computation times, while the A* model search method employed by state-of-the-art domain-independent XAIP methods [Chakraborti *et al.*, 2017] needs exponentially higher computation times as the explanation length increases. We believe optimization solvers could also be leveraged for more general task planning problems, and another direction for future research is on the use of MILP-formulations for solving domain-independent XAIP problems.

One more conclusion from this paper is that some types of explanation cannot answer a question exactly (i.e. are not "explicable" in the sense of [Zhang *et al.*, 2016]). The counter-factual model used as an explanation may have a shortest path that is *closer* to the user's desired path but not exactly the same. This may be undesirable, since it does not explain the desired path exactly, but it may also be desirable in some contexts. For example, it may be hard for users to specify (or "draw") their desired paths exactly, and such an explanation could help users clarify their questions. Additionally, computing a set of explanations of varying complexity-explicability trade-offs could also potentially help users understand the model. These ideas are related to those of explicability-explainability trade-offs [Sreedharan *et al.*, 2017] but deserve further study in the context of path planning, human-AI teaming, and explanation interfaces.

Acknowledgments

This work was supported by the Air Force Office of Scientific Research under award number FA9550-18-1-0245.

References

[Ahuja and Orlin, 2001] Ravindra K. Ahuja and James B. Orlin. Inverse optimization. *Operations Research*, 49(5):771–783, 2001.

[Ahuja *et al.*, 1993] Ravindra K Ahuja, James B Orlin, and Thomas L Magnanti. *Network flows: theory, algorithms, and applications*. Prentice-Hall, 1993.

[Almagor and Lahijanian, 2020] Shaull Almagor and Morteza Lahijanian. Explainable multi agent path finding. In *19th International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*, 2020.

[Alt and Godau, 1995] Helmut Alt and Michael Godau. Computing the fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5(1-2):75–91, 1995.

[Brandao and Havoutis, 2020] Martim Brandao and Ioannis Havoutis. Learning sequences of approximations for hierarchical motion planning. In *30th International Conference on Automated Planning and Scheduling (ICAPS)*, Jun 2020.

[Brandao *et al.*, 2020] Martim Brandao, Omer Burak Aladag, and Ioannis Havoutis. Gaitmesh: controller-aware navigation meshes for long-range legged locomotion planning in multi-layered environments. *IEEE Robotics and Automation Letters*, 5(2):3596–3603, 2020.

[Burton and Toint, 1992] Didier Burton and Ph L Toint. On an instance of the inverse shortest paths problem. *Mathematical programming*, 53(1-3):45–61, 1992.

[Chakraborti *et al.*, 2017] Tathagata Chakraborti, Sarath Sreedharan, Yu Zhang, and Subbarao Kambhampati. Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. In *26th International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.

[Chakraborti *et al.*, 2019] Tathagata Chakraborti, Anagha Kulkarni, Sarath Sreedharan, David E Smith, and Subbarao Kambhampati. Explicability? legibility? predictability? transparency? privacy? security? the emerging landscape of interpretable agent behavior. In *29th International Conference on Automated Planning and Scheduling (ICAPS)*, volume 29, pages 86–96, 2019.

[Diamond and Boyd, 2016] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[Hauser, 2014] Kris Hauser. The minimum constraint removal problem with three robotics applications. *The International Journal of Robotics Research (IJRR)*, 33(1):5–17, 2014.

[Kwon *et al.*, 2018] Minae Kwon, Sandy H Huang, and Anca D Dragan. Expressing robot incapability. In *2018 ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 87–95, 2018.

[MobyGames, 2019] MobyGames. Games using recast. *https://www.mobygames.com/game-group/middleware-recast*, 2019.

[Mononen, 2014] Mikko Mononen. Recast navigation. *https://github.com/recastnavigation/recastnavigation*, 2014.

[Rosenthal and Veloso, 2012] Stephanie Rosenthal and Manuela Veloso. Mobile robot planning to seek help with spatially-situated tasks. In *26th AAAI Conference on Artificial Intelligence (AAAI)*, 2012.

[Sreedharan *et al.*, 2017] Sarath Sreedharan, Subbarao Kambhampati, et al. Balancing explicability and explanation in human-aware planning. In *2017 AAAI Fall Symposium Series*, 2017.

[Sreedharan *et al.*, 2018] Sarath Sreedharan, Siddharth Srivastava, and Subbarao Kambhampati. Hierarchical expertise level modeling for user specific contrastive explanations. In *27th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4829–4836, 2018.

[Sreedharan *et al.*, 2019] Sarath Sreedharan, Siddharth Srivastava, David Smith, and Subbarao Kambhampati. Why can't you do that hal? explaining unsolvability of planning tasks. In *28th International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

[Van Toll *et al.*, 2016] Wouter Van Toll, Roy Triesscheijn, Marcelo Kallmann, Ramon Oliva, Nuria Pelechano, Julien Pettré, and Roland Geraerts. A comparative study of navigation meshes. In *9th International Conference on Motion in Games*, pages 91–100. ACM, 2016.

[Wachter *et al.*, 2017] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harvard Journal of Law & Technology*, 31, 2017.

[Zhang and Pavone, 2016] Rick Zhang and Marco Pavone. Control of robotic mobility-on-demand systems: a queueing-theoretical perspective. *The International Journal of Robotics Research (IJRR)*, 35(1-3):186–203, 2016.

[Zhang *et al.*, 2016] Yu Zhang, Sarath Sreedharan, Anagha Kulkarni, Tathagata Chakraborti, Hankz Hankui Zhuo, and Subbarao Kambhampati. Plan explicability for robot task planning. In *RSS Workshop on Planning for Human-Robot Interaction: Shared Autonomy and Collaborative Robotics*, 2016.

[Zhang *et al.*, 2017] Yu Zhang, Sarath Sreedharan, Anagha Kulkarni, Tathagata Chakraborti, Hankz Hankui Zhuo, and Subbarao Kambhampati. Plan explicability and predictability for robot task planning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1313–1320. IEEE, 2017.