Minimizing Robot Navigation Graph for Position-Based Predictability by Humans

Extended Abstract

Sriram Gopalakrishnan
School Of Computing & AI, Arizona State University
Tempe, USA
sgopal28@asu.edu

Subbarao Kambhampati
School Of Computing & AI, Arizona State University
Tempe, USA
rao@asu.edu

ABSTRACT

When multiple humans and robots are moving in spaces like restaurants, hospitals, or banks, making the robot's movements easy to predict can help the humans co-navigate the space with the robots. Since people would be busy with their own goals, they are not paying close attention to the prior movements, or goals of multiple robots. So predictability from the robot's current position alone would help. With this in mind, we propose using an algorithm to lay out fixed paths for the different tasks the robots would do, such that predictability from only the current position alone is optimized, and motion costs are kept within acceptable bounds.

KEYWORDS

Robot Navigation; Navigation Graphs; Human-Robot Interaction; Position-Based Predictability

ACM Reference Format:
Sriram Gopalakrishnan and Subbarao Kambhampati. 2022. Minimizing Robot Navigation Graph for Position-Based Predictability by Humans: Extended Abstract. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 3 pages.

1 INTRODUCTION

Existing research on robot navigation with multiple humans and robots uses a complex interplay of sensing, high level path planning, predicting human motion, and reactive behavior (collision avoidance) [5]. Navigation in larger groups of people is much harder as things like crowd dynamics need to be considered as well ([6], [7], [1]). We take a different approach and compute a restricted *navigation-graph* that limits the robot's motion, and keeps motion costs within a predefined bound. Limiting motion was previously done with Automated/Autonomous Guided Vehicles (AGVs) which is a mature technology, and already in use for industrial settings, and even hospitals [2]. AGVs followed a predefined path laid out with special tape to move between positions. In this paper, we propose to algorithmically computing the AGV grid (as a directed graph) for more predictable trajectories to humans; predictability is paramount for human robot interactions [5] and more predictability would improve adoption of AGV's in everyday settings like restaurants. We do this by minimizing the number of "Branching-Vertices", which are vertices (positions) with more than one outgoing edge. Fewer branching vertices in the navigation graph mean fewer possible trajectories from any position; this allows easier prediction or bounding of the robot's motion from *just the current position*, which we call *position-based predictability*. This matters since the human is not paying much attention to the robot, and may only know the current position. We don't expect that a person will predict the entire future path of the robot; they'd only predict the immediate next steps, enough to decide their own path. This is what we aim to make easier.

In this paper, we formalize the problem of minimizing the navigation-graph for position-based predictability. We then introduce measures for position-based predictability, followed by a hill-climbing algorithm to minimize the graph and optimize for position-based predictability. For a more complete description of our work, please see our full paper on arxiv [4].

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online.

2 PROBLEM FORMULATION

The problem of graph minimization for position-based predictability is given by the tuple, $P_{min} = \langle G, T, C, W \rangle$ and the terms are defined as follows:
G is a directed graph with vertices and edges *(V,E)*. It captures all allowed motion of the robot.
T is the set of ordered vertex pairs associated to tasks, that need to remain connected in order to do those tasks.
C gives the cutoff (distance) cost associated to each task in T.
W returns the weight of each pair in T. This can be the probability or importance weight. The objective is to reduce the input graph G to improve position-based predictability measures (as follows).

2.1 Measures for Position-Based Predictability

We will use two intuitive measures for position-based predictability, which focus on branching vertices. Intuitively, more branching vertices imply more paths for the human to consider, and so makes it harder to navigate.

Weighted Prediction Cost (*WPC*): WPC captures the prediction cost for tasks in T. There are two components multiplied together in WPC (Equation 1): the first counts the number of branching vertices that appear on the paths for the robots tasks; the second is the sum of the branching factor of those branching vertices.

$$WPC(G,T,W) = \sum_{t \in T} \sum_{v \in SPV(G,t)} W(t) * 1[deg^+(G,v) > 1] \times \sum_{t \in T} \sum_{v \in SPV(G,t)} W(t) * deg^+(G,v) \quad (1)$$

where $SPV(G,t)$ returns the shortest path vertices for the input task (t) in the graph (G) being evaluated, and $deg^+(G,v)$ returns the outdegree of the vertex(v) in the graph(G). If there are multiple shortest paths for a task, then the path that contributes the least cost to WPC is used.

Weighted Ratio *NV/NBV*: where NV is the number of vertices and NBV is the number of branching vertices. This captures the average number of sequential steps a robot can take with no branching. This is also weighted by the task weights. Larger NV/NBV value implies longer path segments with no branching, and makes predicting or bounding the next few steps from the current position trivial. NV/NBV is formalized as:

$$NV/NBV(G,T,W) = \frac{\sum_{t\in T}\sum_{v\in SPV(G,t)} W(t)}{\sum_{t\in T}\sum_{v\in SPV(G,t)} W(t) * 1[deg^+(G,v) > 1]} \quad (2)$$

An example of the type of optimization that this work does is shown in the rectilinear grid-graph in Figure 1. The blue vertices are the terminal vertices and vertices highlighted in red are branching vertices. In the figure, we show two graph optimizations (minimizations); one that focuses only on graph size (number of vertices and edges), and another minimization that optimizes for position-based predictability (fewer branching vertices *and* graph size). The former is related to a graph problem known as Strongly Connected Steiner Subgraph(SCSS) [3] problem(see full paper).

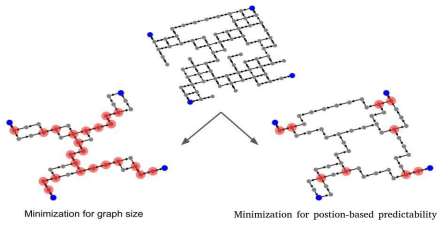

Figure 1: Example of graph minimization for graph size, and minimization for position-based predictability. Blue vertices are the terminal vertices that must stay connected, and vertices highlighted in red are branching vertices

3 METHODOLOGY

To optimize the graph, we use a hill-climbing search. We start with a population of diverse paths connecting the terminals T. At the first step, we assign a random path for each pair in T, and during hill-climbing we replace or remove paths in order to improve the score of the graph resulting from the combination of paths used. The critical part of our approach is how a graph is scored. We use two scores, a *Graph-size cost* or GSC and a *Branching-Vertices Cost* (BVC). GSC counts vertices and edges, and BVC is GSC multiplied by the number of branching vertices. GSC captures the objective function of the Strongly Connected Steiner Subgraph(SCSS) problem, and so is used as the baseline; if using GSC gave competitive results, then it would imply other SCSS algorithms ought to be considered as well. Equations for the scoring functions, pseudocode, and algorithm details can be found in the full version of this paper on arxiv [4].

4 EXPERIMENTS AND RESULTS

We tested our algorithm on randomly generated 20x20 grid-graphs. We randomly drop 20% of the vertices, and 20% of the remaining edges. Lastly we arbitrarily select the terminal vertices. All ordered pairs of terminal vertices define the set T. We vary the number of terminal vertices as $\{3,4,6,8\}$. We also vary cutoff cost C for the path costs between the terminal vertices. The cutoff cost is set as a multiple of the shortest path cost. We vary C as $\{1,2,3,5\}$ where 1, means only optimal paths are considered. Lastly, the weights W are randomly assigned to all tasks in the range $[0,1)$ and normalized so they would sum to 1. In total, the experimental settings include every combination of the number of terminal vertices and cutoff bounds. For each combination, we generate 10 random graphs (setting the random seed in the code from 0 to 9) and ran the algorithm with both cost functions (GSC and BVC) on the same graphs to compare them. We found BVC did appreciably better than GSC for the position-based predictability measures defined earlier. As GSC captures the objective of SCSS problem, the results tell us that optimizing for the SCSS problem is not a surrogate for this problem's objective. For detailed results see [4].

We also conducted human studies to show the effect of branching vertices on reasoning about paths. We presented two closely matched problems of robot navigation with a human following a fixed path in the same space as shown in Figure 2. We asked the user to determine at what step a robot would collide with the human, and measured how long it took them. The problem with fewer branching vertices took on average 47.5 seconds *less* than the other. We verified statistical significance with the paired t-test analysis; p-value of 0.0006 (t-statistic -3.849) meant we could reject the null-hypothesis of no difference in time taken. This is expected as more branching vertices result in more paths to consider; this makes it harder to plan conflict-free paths in the space.

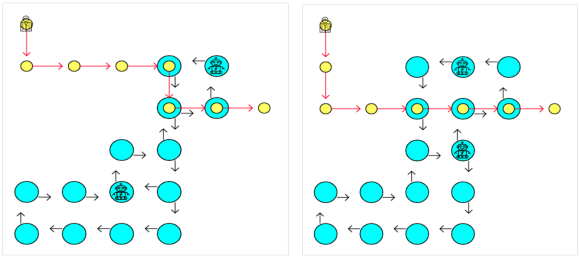

Figure 2: Problems given in the human subject experiments; problem 1 (left) has more branching vertices and problem 2 (right) has fewer.

ACKNOWLEDGEMENTS

This research is supported in part by ONR grants N00014- 16-1-2892, N00014-18-1- 2442, N00014-18-1-2840, N00014-9-1-2119, AFOSR grant FA9550-18-1-0067, DARPA SAIL-ON grant W911NF19- 2-0006 and a JP Morgan AI Faculty Research grant.

REFERENCES

[1] Yuying Chen, Congcong Liu, Bertram E Shi, and Ming Liu. 2020. Robot navigation in crowds by graph convolutional networks with attention learned from human gaze. *IEEE Robotics and Automation Letters* 5, 2 (2020), 2754–2761.

[2] Hamed Fazlollahtabar and Mohammad Saidi-Mehrabad. 2015. *Autonomous guided vehicles*. Vol. 20. Springer.

[3] Jon Feldman and Matthias Ruhl. 2006. The directed Steiner network problem is tractable for a constant number of terminals. *SIAM J. Comput.* 36, 2 (2006), 543–561.

[4] Sriram Gopalakrishnan and Subbarao Kambhampati. 2020. Minimizing Robot Navigation-Graph For Position-Based Predictability By Humans. *arXiv preprint arXiv:2010.15255* (2020). https://arxiv.org/pdf/2010.15255.pdf

[5] Thibault Kruse, Amit Kumar Pandey, Rachid Alami, and Alexandra Kirsch. 2013. Human-aware robot navigation: A survey. *Robotics and Autonomous Systems* 61, 12 (2013), 1726–1743.

[6] Peter Trautman and Andreas Krause. 2010. Unfreezing the robot: Navigation in dense, interacting crowds. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 797–803.

[7] Dizan Vasquez, Billy Okal, and Kai O Arras. 2014. Inverse reinforcement learning algorithms and features for robot navigation in crowds: an experimental comparison. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 1341–1346.