Information Markets for Multi-Robot Navigation Under Uncertainty

Information Markets for Multi-Robot Navigation Under Uncertainty

a thesis presented
by
Kathryn Wantlin
to
The Department of Computer Science

in partial fulfillment of the requirements
for the degree of
Bachelor of Arts (Honors)
in the subject of
Computer Science

Harvard University
Cambridge, Massachusetts
May 2021

Thesis advisors: Professor David Parkes and Dr. Sarah Keren Kathryn Wantlin

Information Markets for Multi-Robot Navigation Under Uncertainty

Abstract

We investigate the role of an information market in improving navigation outcomes for risk-averse multi-agent systems with incomplete state information. Given a partially observed graph representation of an environment map and the deployed agents, we first create a path planning heuristic that characterizes each agent's path preference, given the currently observable occupancy statuses of environment positions, by the expected utility of path traversal. Secondly, we utilize prior work in the field of decision theory to determine the value of receiving information on positions with uncertainty as well as the cost to obtain the information. Lastly, we create a suite of assignment algorithms to best determine how agents can cooperate to collect additional information and facilitate path planning updates. A synthetic dataset of environments, settings, and team compositions is created according to the operation specifications considered in this paper and is used to demonstrate that information exchange yields significant improvements in the global utility outcomes both in expectation and in practice.

Contents

Listing of figures

List of Tables

Acknowledgments

I WOULD LIKE TO EXPRESS my deepest appreciation to my advisors, Professor David Parkes and Dr. Sarah Keren, for being outstanding teachers and mentors throughout the process of writing this thesis. Through many months of exploration, feedback, and revision, they have contributed immensely to my development as a researcher, and this project would not be possible without their guidance. I am particularly grateful to Dr. Keren for her generosity with her time and for her role in helping to inspire this project.

I also thank Professor Ariel Procaccia for kindly offering to be my thesis reader.

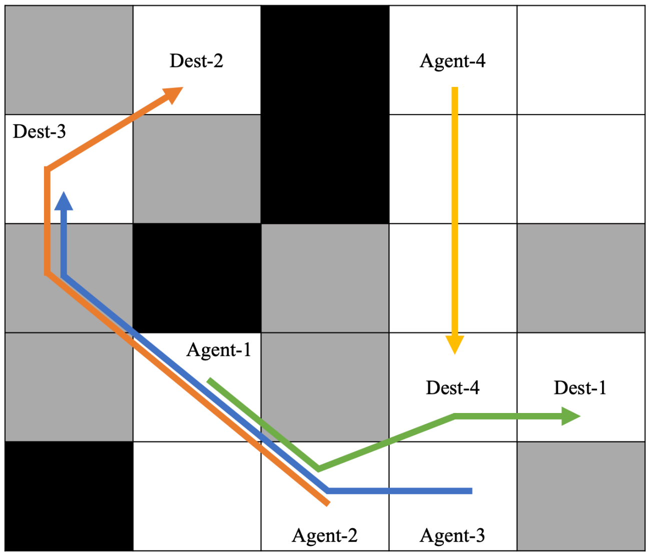
Lastly, I thank my parents for consistently supporting me in my studies during this challenging year of remote learning and research.

1
Introduction

Making decisions under uncertainty is a highly crucial aspect of robotic navigation, given that real-world deployed robots inherently operate with noisy or incomplete information due to hardware-level issues or lack of domain data. The most prominent sources of uncertainty that have been treated by the field fall under the umbrellas of localization and navigation, where small errors in sensor readings, wheel feedback, mapped environment data, and other inputs can lead to agent failures, damage, or simply less than optimal planning and coordination. In this paper, we focus on the uncertainty at the system level concerning an initial occupancy grid provided as a map to a set of agents who wish to navigate paths to pre-assigned destinations. We wish to understand how coordination between the agents can best resolve the uncertainty and aid agents in finding safe, short paths to reaching their destinations.

1.1 Motivation

Figure 1.1: Motivating Example



Consider the motivating example displayed in Figure 1.1. Here we have visualized a system of four agents in a two dimensional grid, having pre-assigned them to initial and destination positions. The white and black cells designate positions in which we have perfect information that the cell is free or blocked, respectively, and we have some uncertainty about all the grey cells. The green, orange, blue, and yellow lines signify the optimal base paths selected by Agents 1-4, respectively.

Given the grid environment, we wish to understand first how this combination of paths has been selected and if it most appropriately captures the payoffs possible in this setting. Uncertainty in robotic mapping is typically presented in the form of an occupancy grid where each cell is associated with a probability value that the cell is blocked.[16] Assuming we have this probabilistic expression of the environment and the payoffs for each state, it would be possible to characterize, in expectation, the preferences of each agent for particular paths and arrive at a set of base paths such as the one in Figure 1.1.

At this point, agents may still experience significant uncertainty; for example, in the illustrated setting, Agent 2 and Agent 3's paths require traversal over unknown cells in the map. Completely removing all sources of uncertainty may be infeasible, so we want to understand how best to minimize the uncertainty with a given set of resources.

Considering the agents to be resources themselves and looking at the distribution of their base paths, we see that some unknown positions may be more impactful than others. Namely, the unknown areas of the map that could most sway the agents' base paths to more preferred outcomes or simply decrease uncertainty of the current path option if no alternatives exist are highly valuable. Understanding how to quantify this information value for the system would allow us to determine which areas of the grid, if resolved, could bring the system to a more optimal global planning outcome. For example, since Agents 2 and 3 both pass through a common unknown cell, coordination between them would likely be desirable since the task of travelling to the unknown cell to resolve its true occupancy status could be assigned to one agent for the benefit of others.

1.2 Related Work

The literature surrounding this paper's methods fall primarily in the following categories: robotic navigation under uncertainty, cooperative mapping and sensing, market-based task allocation, decision theory's "value of information" definition, and combinatorial auctions.

Due to its prevalence in real-world applications, uncertainty has been investigated in several forms in the field of robotics. At the level of the individual agent, uncertainty from sources such as depth sensors, wheel feedback, and other hardware elements can create large compounding errors in navigation and localization. Lazanas and Latombe[11] developed a landmark method in response to uncertainty during both the planning and execution phases of single agent robotic operation. By establishing a baseline of certainty, which they term "islands of perfection", at certain landmark regions, they are able to develop a demonstrably reliable planner in response to sensing information. Other work by Takeda and Latombe[21] focuses on path planning in the multi-agent case and seeks to determine optimal paths that minimize expected errors, where error is determined by comparing the system's current model of the environment with individual sensing data. We will build on this idea of selecting paths by solving an optimization problem on the currently perceived uncertainty.

Outside of uncertainty in path planning, it is also important to note the related problem of uncertainty in localization, which has been treated within planning research to enable exploration of unknown environments.[1] In this paper, we will not deal with localization uncertainty, but our framework leaves room for extension into this problem area in future work. In our setting, we prioritize minimizing uncertainty and error based on agents' reported preferences. Likhachev and Stentz's[12] work in this area is highly relevant and also employs a graph representation of the state space similar to what we will use to characterize the environment. Their calculation of expected payoffs also lays the groundwork for an extension of this paper into replanning settings. Probabilistic treatments of the planning problem in similarly partially observed environments was studied earliest by Simmons and Koenig[18], who employ a topological map in their algorithm to direct goal-oriented actions.

Within the field of multi-agent systems, prior literature points to high potential for distributed sensing and planning to improve the distribution of actions between agents and reduce estimation error. One benefit arises when considering teams of heterogeneous agents, which may be specialized for completing certain actions over others. Parker's key contribution on the ALLIANCE architecture expresses this heterogeneity in terms of a "patience" factor and uses this to establish optimal task allocations amongst agents.[13] We will employ a similar idea beyond validating the efficacy of our path planning heuristic to show how best to divide the work of reducing uncertainty in an environment. Another benefit surrounds a system's potential to leverage its constituent agents as a proxy for an external information source, such as GPS or other "oracle" provider, which yields benefits particularly in situations with insufficient data or limited connectivity.[20] This is the assumption under which we develop our setting, where although we assume perfect communication between agents, we begin navigation with an incomplete map of the environment.

Market-based methods of allocating tasks, where agents negotiate their individual interests to arrive at an optimal assignment, were first studied by Smith's Contract Net Protocol[19]. This work establishes a decentralized agent network and enables a negotiation scheme to solve the task distribution problem. As the literature and hardware evolved, the MURDOCH framework was introduced, which utilized a publish-subscribe architecture to facilitate negotiation.[8] This model is also now utilized by the popular open-source Robot Operating System suite, which supports realistic simulation of hardware for robotic software development.[16] MURDOCH also introduces the role of an "auctioneer", which we will similarly utilize to coordinate the computation of negotiation. Other market-based techniques vary approach using different coordination languages[4], introducing competition between agents[7], or focus on globally optimal assignment.[14]

To determine the value of items traded between agents, our system also touches on the concept of "value of information". With a founding in decision sciences, information value was first investigated in this way by Howard, who calculates the expected value based on the possible results of a given information request.[10] Zilberstein and Lesser extend this idea to develop a modern framework of information gathering planning.[24]

Under the umbrella of market-based methods, much literature exists specifically on auction methods, which have well-established performance competitiveness. Gerkey and Matarić survey several key algorithms within the multi-robot task allocation taxonomy, with a particular focus on iterative and online approaches.[9] Another of their papers serves as the inspiration for the greedy approach used in our setting, and although their Broadcast of Local Eligibility (BLE) algorithm technically falls under behavior-based methodologies, by replacing the utility heuristic they employ with a profit-based objective function, we can translate their robot-task interactions into robot-robot interaction.[23]

In trying to find an assignment procedure that more closely approximates the optimal assignment, auction-based methods exhibit suitable competitiveness and also facilitate the "bundling" actions that is useful to our setting. Approximating the determination of an optimal subset of assets, which might exhibit characteristics of both complements and substitutes, non-additive pricing, or other complicating features, can be achieved via combinatorial auction.[22] This method, which was first introduced by Rassenti, Smith, and Bulfin for airport time slot allocation, involves both a bidding phase and a winner-determination phase, and each of these phases have been studied individually to determine optimal algorithms.[17] Auctions also provide the benefit of gradually revealing participants' supply and demand functions, and in our setting, the supply side will be most challenging. However, unlike other auction applications, our setting does not necessarily require strict performance guarantees for honest bidding or maximum revenue collection. Instead, given relatively easier computation of the demand-side, we use the framework to determine the supply-side best response. Parkes and Ungar's work on the iBundle algorithm is particularly relevant for this purpose, and also introduces the idea of a "provisional allocation" which will allow the demand-side to update in response to supply-side negotiation.[15]

1.3 Contribution

The contribution of the paper is threefold. Firstly, given a probabilistic environment abstraction and a set of agent following a set of operation rules, we define a planning heuristic rooted in expected utility of traversing a particular path. Using this heuristic, we utilize prior work from decision theory to determine the value of receiving information that improves the knowledge set agents use to plan. By characterizing the value and cost of certain information bundles, we test a set of algorithms that return assignments of agents to bundles in a way that allows agents to cooperate with each other to improve the global navigation outcome. We explore several such algorithms to find computationally feasible and sufficiently optimal procedures for practical application, and demonstrate a statistically significant improvement in outcomes due to employing these assignment algorithms.

2
The Agent's Decision Problem

THE INVESTIGATION BEGINS by considering the decision problem of a single agent in a simple setting. This agent is located in an environment which we characterize as a two dimensional occupancy grid $O : \{c_i\} \rightarrow [0, 1]$ which associates a given cell $c = (x, y)$, defined by its x and y coordinates, with a probability α_c that the cell is occupied. If c is known to be free, then $\alpha_c = 0$, and if c is known to be blocked, then $\alpha_c = 1$. For all other ”unknown” cells, there is some uncertainty about the true occupancy status. Within this grid, assume that the agent knows with complete certainty its own initial position c_{init} and its destination position c_{dest}, and that $\alpha_{c_{init}} = \alpha_{c_{dest}} = 0$. There may also be other agents present in the environment, with their own respective initial and destination positions. Information about the grid's state is shared completely among all agents and updated instantaneously as agents move within the environment. Assume the agent also knows that there is a reward $r(c_{dest})$ located at c_{dest} which it will claim upon arriving at that cell and that no other agent can claim, even if other agents were to move onto this cell. Once a reward has been claimed by an agent, it cannot be reclaimed. Agents' movement is constrained to step-wise movement to adjacent cells, including along grid diagonals, and each step incurs a movement cost of 1. We assume that agents will not move onto known blocked cells, but attempting to move onto an unknown cell will reveal its true occupancy status to all agents in the environment. Assume agents have distinct initial and destination positions and that multiple agents occupying the same grid cell will not present an issue. The agent then seeks to determine the optimal simple path from its c_{init} to c_{dest}.

2.1 Evaluating Paths

First, we must define a function that maps from c_{init}, c_{dest}, and O to an optimal path. To construct our definition of optimality, note that with the given input information, the agent will exhibit preferences for certain paths. For example, the agent might prefer to take less risky paths which are both shorter and less likely to be obstructed. We imagine a setting where agents exploring an environment under uncertainty will fail upon attempting to move onto a blocked cell and become inoperable, no longer able to take further actions or collect reward. In order to quantify its path preference, the agent can attach to each possible path an expected utility of traversing this path to reach c_{dest}. With this construct, the agent can determine the utility of any path from c_{init} to c_{dest} and select the path with highest corresponding utility. We seek an efficient algorithm to determine this optimal path.

To determine the utility of a path consider the following: characterize a path p as a list of $|p| = N$ points $[c^p_0, c^p_1, c^p_2, ..., c^p_N]$, where the nth point in path p has a known probability of being obstructed $\alpha_{c^p_n}$. Assume also that c^p_0 is a cell adjacent to c_{init} (the first cell an agent moves to while following this path) and $c^p_N = c_{dest}$. The expected utility of p must capture the probability and value of all possible events that could occur during its traversal, which consist of either safely arriving at a certain cell or attempting to move to a cell only to find it obstructed. The value associated with the event that the agent reaches c_{dest} without being obstructed is $r(c_{dest}) - N$. The value of arriving at any other point in p without having been obstructed is 0, and the value of safely traversing p up until a point c^p_n, where failure occurs due to c^p_n being obstructed, is $-n$. Therefore, we can calculate the expected utility of traversing path p as follows:

$$EU(p) = [r(c_{dest}) - |p|] \prod_{n \in |p|} (1 - \alpha_{c^p_n}) - \sum_{n \in |p|} [n\alpha_{c^p_n} \prod_{k \in [n-1]} (1 - \alpha_{c^p_k})] \tag{2.1}$$

Calculating expected utility rather than expected cost has benefits for our application. In addition to allowing for easier generalization to scenarios with more specialized reward schemes, working with expected utility ensures that we are properly accounting for the probability of reaching the destination given accumulated path traversal risk. We will specifically set $r(c_{dest})$ to be larger than the maximum possible cost to prevent counterintuitive behavior. In two-dimensional grid environments, this can simply be set to the number of cells in the grid.

2.2 Computing the Optimal Path

Now the agent must solve the problem of finding the maximum utility, single-source, single-destination path within a directed, weighted, cyclic graph. To see that this is the case, note that we can build a graph to represent O where each grid cell c is made a vertex and each vertex has at most 8 edges, where each neighboring vertex is an adjacent cell on the grid that can be reached in the grid via one step. Nodes for "center" cells have 8 edges, "corner" cells 3 edges, and other "boundary" cells 5 edges. The weight of each edge ab, where a and b are the two vertices incident to this referenced edge, is characterized by both the α_b of the corresponding out-neighbor b of a, as well as the step cost of 1 between vertices. Call this graph representation G and the set of its vertices V.

We propose an algorithm to solve this problem in polynomial time with a modified Dijkstra's algorithm.[3] We first note that removing cycles will always serve to increase the expected utility of the resulting simple path since cycling introduces undesirable additional step-wise cost and risk terms to the expected utility of the path. This allows us to use a Dijkstra-like search with the guarantee of termination.

For simplicity, note that in the following algorithm outline s is the source vertex (previously dubbed c_{init}), d is the destination vertex (previously dubbed c_{dest}) and $U(x)$, our proxy for the "distance" metric that is typically used in Dijkstra, signifies the expected utility of a path p' from s to x if we were to receive $R = r(c_{dest})$ at x. $U(x)$ thus captures the expected cost of the best path found so far up to vertex x and also captures expected reward based on the vertices traversed so far. Note that this means that once the algorithm terminates, $U(d)$ will contain the expected utility of a path to c_{dest} that is consistent with Equation 2.1. In practice, we keep track of the maximum $U(x)$ from the source vertex to any given vertex x and the optimal edge connections between vertices in dictionary-like data structures as $U[]$ and $prev[]$, respectively. Let "getPath(prev[], s, d)" be a helper function that takes $prev[]$, s, and d and returns the path from s to d given the current edge connections stored in $prev[]$. Algorithm 1 outlines our base path algorithm.

Conjecture 2.2.1. Given graph G with vertices V, source node s, destination node d, and reward R, Algorithm 1 for Optimal Path Determination will return the optimal path P^* and corresponding expected utility $U(d)$.

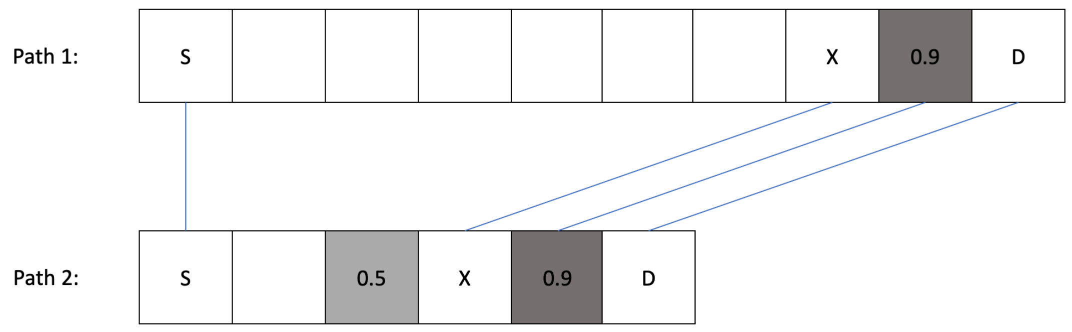
We conjecture that this algorithm returns the optimal solution. Like Dijkstra's algorithm, our

Algorithm 1 Optimal Path Determination

```
    Input: Graph $G$ with vertices $V$, source node $s$, destination node $d$, and reward $R$
    Output: Max expected utility $U(d)$ and corresponding optimal path $P^*$
1:  procedure MOD-DIJKSTRA(G,V,s,d)
2:      for $x \in V - \{s\}$ do
3:          $U(x) \leftarrow -\infty$
4:          $prev[x] \leftarrow$ None
5:      $U(s) \leftarrow R$
6:      $Q \leftarrow \{s\}$
7:      $P^* = []$
8:      while $Q \neq \{\}$ do
9:          $Q' \leftarrow \{\}$
10:         for x in A do
11:             for out-neighbors y of x do
12:                 if $U(y) > U(x|$"xy" next edge$)$ then
13:                     $U(y) \leftarrow U(x|$"xy" next edge$)$
14:                     $prev[y] \leftarrow x$
15:                     $Q' \leftarrow Q' \cup \{y\}$
16:                     if $y = d$ then
17:                         $P^* =$ getPath(prev[], s, d)
18:         $Q \leftarrow Q'$
    return U(d), P*
```

algorithm has polynomial runtime and uses the same update rule for determining optimal edge connections in *prev*[] and the associated expected utilities. However, unlike Dijkstra's, we use a simple queue as opposed to a priority queue and stores the optimal path to *d* particular separately. This allows us to explore shorter paths (in number of steps) first, replacing those with longer paths if we determine them to offer sufficient safety benefits. It also ensures that if make further modifications to *prev*[] after finding an optimal solution (due to the queue being nonempty), those potentially non-optimal edge connections will not be erroneously returned.

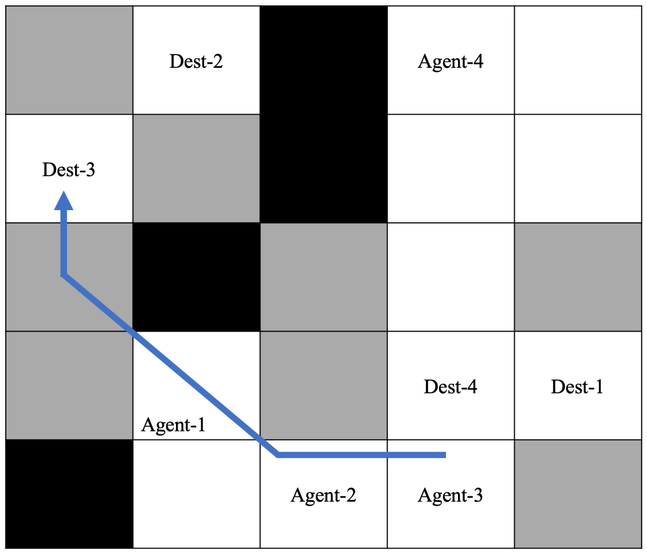
Figure 2.1: Pathfinding Algorithm Example



To understand how the simple queue benefits us, consider the example displayed in Figure 2.1. Assume that the agent has the two path options as shown, where the individual cells are depicted. Blue lines link identical cells between the two paths, and we assume that all non-linked cells are distinct. The source and destination nodes are labelled with "S" and "D", and the α values of unknown grey cells are also included. Assume the reward at "D" is 100. Overall, the agent is deciding between two paths which intersect at cell X and afterward move along the same cells. Before intersecting at X, Path 1 traverses more cells, which are all free, while Path 2 traverses fewer cells, one of which is unknown.

Applying Algorithm 1, if we were to follow the updates along Path 1, we would see that the algorithm would find $U(x) = 100 - 7 = 93$. If we follow updates along Path 2, the algorithm would find $U(x) = 0.5(-2) + 0.5(100 - 3) = 47.5$. However, ultimately the true expected utility of Path 1 is $0.9(-8) + 0.1(100 - 9) = 1.9$ and the expected utility of Path 2 is $0.5(-2) + (0.5)(0.9)(-4) + (0.5)(0.1)(100 - 5) = 1.95$. Therefore, Path 2 is the optimal choice, but if we had performed our algorithmic search to X through Path 1 first before exploring through Path 2, then we would assign the utility at X to be 93 and would not continue exploring Path 2 past X. Therefore, it is important for us to perform our Breadth-First-Search using a simple queue to explore shorter paths first and switch to longer paths only if we find them to be superior alternatives. The full proof of optimality of Algorithm 1 is let for future work.

Figure 2.2: Base Path Example



Let's consider an example to illustrate expected utility path determination. Figure 2.2 visualizes a possible environment, where blocked cells are black, free cells are white, and unknown cells (all with $\alpha = 0.5$) are grey. The four agents' initial and destination positions are labelled. Assume each agent's destination reward is 25. The blue line denotes Agent 3's optimal base path. From our expected utility formulation, we see that Agent 3's base path has expected utility $\frac{1}{2}(-3) + \frac{1}{2}(25 - 4) = 9$, since there is a $\frac{1}{2}$ probability that the agent will fail after 3 steps and $\frac{1}{2}$ probability that the agent will successfully arrive at it's destination after 4 steps and receive a reward of 25.

Also note that in environments where no base path can be found due to the density and configuration of blocked points, we will consider the agent's optimal base path to be no movement, with expected utility of 0.

2.3 Value of Information

From the above example, we can see that especially in environments with little known information, the uncertainty of paths can be quite high. To improve upon this would require obtaining additional information about the state of the environment. This information could be gained by the agent's traversal of the environment, or via other agents in the environment with which an agent can interact. To decide which information to collect from the environment or other agents, an agent needs a heuristic to, before traversing its base path, assess the tradeoffs of information acquisition and exchange. We therefore wish to define a function mapping from a *waypoint* $w \in W = \{c | \alpha_c \in (0, 1)\}$ and the agent's c_{init}, and c_{dest} to *value of information*.

There are several options for types of information an agent might like to request, but for now, consider the case where the agent can obtain additional information on the occupancy status of waypoint cells W. The agent must then decide if it might like to query for information on a bundle $W_i \subseteq W$. To evaluate which bundle W_i, the agent must develop a metric for how much it values having information about the occupancy status of points in W_i. Note that the information could be that cells are either free or blocked, and having perfect information in either direction is valuable; if the agent discovers new free cells, it could potentially find a shorter, safer path to its destination, and if the agent discovers new blocked cells, it could potentially forego its base path for a safer alternative. These considerations will factor into the way agents value each W_i.

As already hinted, the value of information is highly connected to the notion of path improvement; if an agent's plan will be no better off as a result of obtaining a certain set of information, then that set of information would have no value. Therefore, it is reasonable to characterize value of information as the expected improvement in utility when comparing the path determined after gaining new information with the base path.[24] This is also consistent with prior definitions from the field of decision theory.

Consider the optimal base path for agent n to be p_n, with associated expected utility $EU(p_n)$. The probability of the event that the agent learns that a subset of cells $F_i \subseteq W_i$ are free and all cells in $B_i = W_i - F_i$ are blocked is $P(F_i) = \prod_{c \in F_i}(1 - \alpha_c) \times \prod_{c \in (W_i - F_i)} \alpha_c$. For convenience, we are referring to this event, given W_i, as F_i.

Given this event F_i, we can modify the graph representation G of the environment to reflect this event by changing the edge weights such that for every edge in G with out-neighbor $v \in W_i$, if $v \in F_i$, then that edge's weight is changed to 0, and to 1 otherwise. From here, we can simply use our Modified Dijkstra's algorithm presented previously to determine an optimal path $p|W_i, F_i$ conditional on the given bundle W_i and event F_i from the source vertex to the destination vertex and the path's associated utility.

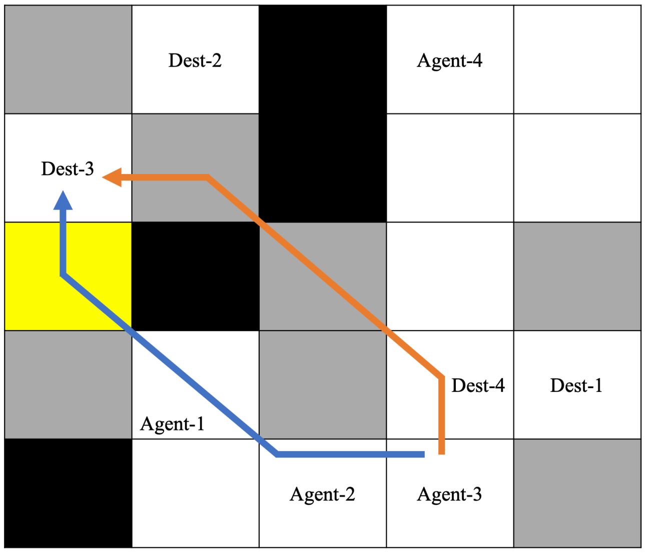
In order to determine the value of information at W_i, it is necessary to consider all possible events $F_i \subseteq W_i$. Thus, agent n's expected value of information on W_i expressed as a function is:

$$V_n(W_i) = \sum_{F_i \subseteq W_i} P(F_i)[EU(p'_n|F_i) - EU(p_n|F_i)] \tag{2.2}$$

where $EU(p'_n|F_i)$ is the expected utility of agent n's optimal path choice when planning with information that event $F_i \subseteq W_i$ occurred for a given W_i and $EU(p_n|F_i)$ is the expected utility of the agent's base path given that event F_i occurs.

Thus, for any $W_i \subseteq W$, the agent can compute the expected value of information for *waypoints* using expected utilities of *paths*.

Figure 2.3: Value of Information Example



Staying with the same example as the last section, Figure 2.3 shows Agent 3's value of information consideration for the highlighted yellow waypoint. Agent 3's base path (in blue) has expected utility of 9. If the waypoint is a free cell, which based on current knowledge will be true with probability $\frac{1}{2}$, then Agent 3 will maintain its base path, which will have a "true" expected utility of $25 - 4 = 21$ given that the highlighted waypoint is a free cell. However, if the waypoint is blocked, the agent's base path would have expected utility -3. If Agent 3 knew that the waypoint was blocked, it would choose the orange alternate path, which has expected utilitiy $\frac{1}{2}(-2) + \frac{1}{4}(-3) + \frac{1}{4}(25 - 4) = 3.5$. Then overall, Agent 3's value of information for the highlighted waypoint is $\frac{1}{2}(21 - 21) + \frac{1}{2}(3.5 - (-3)) = 3.25$.

Moving forward, we will see that it is also useful to consider the value of a waypoint w given that we will be receiving information on some W_0, which we will call the "provisional bundle" since we will be calculating value of other waypoints *provided that* we are currently going to receive information on W_0. For now, we can note that this conditional value to agent n is $V_n(w|W_0) = V_n(w)$ if W_0 is empty. Generally,

$$V_n(w|W_0) = V_n(W_0 \cup \{w\}) - V_n(W_0 - \{w\}) \tag{2.3}$$

If $w \in W_0$, then $V_n(w|W_0)$ captures the value of keeping w in the provisional bundle. Otherwise, $V_n(w|W_0)$ captures the value of adding waypoint w to the provisional bundle.

2.4 Cost of Information

Now, we will consider possible mechanisms by which agents can obtain information. Assuming that no system administrator or oracle is present but other agents are present in the system, it would be possible to query other agents for information about certain locations within the environment, creating a case of distributed sensing that enhances agents' ability to solve the navigation and exploration problems more effectively.[5] Also assume for now that any unknown cell on which an agent steps will be immediately established in the common map as unobstructed, and attempting to step onto a grid location that is obstructed is impossible. For any agent to resolve the occupancy status on a waypoint itself would require incurring some additional cost for no additional reward, meaning that incentive from other agents would be required for the agent, utilizing an expected utility metric to determine optimal paths, to favor a path of assistance over its base path. Therefore, in this section, we wish to determine a function that maps from agent n's c_{init}, c_{dest}, and a given waypoint $w \in W$ to a cost of diverting from its current base path p_n to assist with w before continuing on to its destination. While previously we considered the decision problem from the perspective of agent receiving information, we are now characterizing the decision problem from the perspective of the agent providing information.

With an agent's optimal base path established as p, the agent compares this with the optimal path associated with collecting information, which we call p_W, which is associated with traveling from c_{init} to all waypoints in a bundle W and then continuing on to c_{dest}. Thus, the only difference between these two paths is due to deviating for waypoint traversal, and by comparing $U(p)$ and $U(p_W)$, it will be possible to characterize the expected difference in utility between the two alternatives. The amount for which agent n will need to be compensated to take p_W over p will be

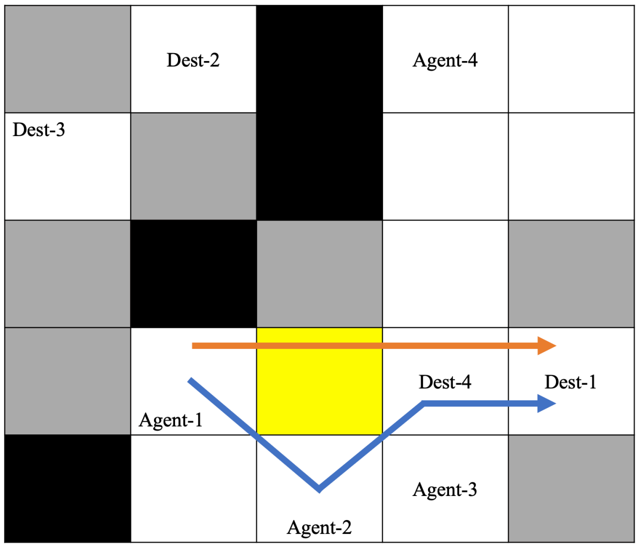
$$C_n(W) = EU_n(p) - EU_n(p_W) \tag{2.4}$$

It is possible to determine the expected utility of traversing between any two grid cells using the modified Dijkstra's algorithm presented earlier. With these utilities established, finding p_W when the order of traversing W is not specified becomes analogous to finding a maximum utility tour of the points c_{init}, c_{dest}, and all points in W. Given that our graph representation of the environment is directed and the expected utility of taking a path in reverse is not equal to that if the path is taken forwards, this is an instance of the Asymmetric Traveling Salesman Problem (ATSP). Polynomial time approximations exist to solve this problem with varying performance guarantees.[2] In order to simplify this problem moving forward, we will constrain ourselves to considering the cost of traversing a bundle in a specific order to collect information.

Assuming then that W is given to the cost function $C_n(W)$ as an ordered set, we can find p_W with the following logic: For a traditional Dijkstra's algorithm that finds the minimum sum-of-edge-costs path p between two vertices, the distance between any two vertices in p is also minimized. If it wasn't, that would imply that we could reduce the length of p by replacing some subpath of p between two vertices x and y with the minimum distance path between x and y. Therefore, finding a minimum distance path p that must pass through some secondary vertex a would be equivalent to combining the minimum distance paths from x to a and then from a to y.

This property of optimal paths computable via a Dijkstra scheme also holds for our expected utility path planning. Therefore, if an agent is given an ordered set of waypoints to traverse in between its initial and destination positions, computing the optimal p_W for a bundle $W = \{w_i\}$ of m waypoints will simply require computing the optimal path between c_{init} and w_1, w_1 and w_2, ..., and w_m and c_{dest}.

Figure 2.4: Cost of Information Example



Consider Figure 2.4 as an example scenario to illustrate the cost of information calculation. Still working with the same environment, we now seek to compute the cost for Agent 1 to collect information at the highlighted waypoint. Agent 1's base path in blue has an expected utility of $25 - 3 = 22$. If the optimal path for Agent 1 to pass through the highlighted waypoint and collect information is the orange path, where this path has utility $\frac{1}{2}(-1) + \frac{1}{2}(25 - 3) = 10.5$, then the cost to Agent 1 to collect information at the highlighted waypoint is $22 - 10.5 = 11.5$.

In certain environments such as this simple one, cooperation may be impossible for some agents due to the fact that all agents are affected by risk and cost in the same way and the placement of waypoints may be too far away from any agent's optimal base path to be feasibly compensated. One way to avoid this is to employ heterogeneous agents, where those less sensitive to risk can provide information to those more sensitive. A classic example is cooperative traversal of an environment by unmanned ground vehicles (UGVs) with the assistance of drones.[6] In our setting, we consider the agents as described already to be UGVs traversing the grid environment. We can relax the constraint of "failure upon stepping on a blocked cell" for some agents while still requiring them to traverse the environment along established graph structure (step-wise, as already described). These agents then operate as drones, and we hypothesize that employing drones who incur less cost for certain risky actions will change the setting dynamics. We also stipulate that drones will have no assigned destination. Therefore, the cost for a drone to collect information at a given w is simply the length of shortest path in the graph from its c_{init} to w, where the graph edge weights are all 1, computable via a traditional minimum distance Dijkstra's algorithm.

3 Information Market

3.1 Problem formulation

The question remaining now is how to determine the criteria for assigning "helper agents" to collect information on bundles. In the toy examples outlined previously, it seems possible to do a simple comparison of value of a bundle with desired compensation for assistance, and if the former is at least as much as the latter, an exchange could occur. Since we're moving beyond the single agent's decision to characterizing coordination of a multi-agent system, we now consider how optimal exchanges should be determined.

Note that so far, each agent in the system has been acting independently in its computation, and in a multi-agent system, this translates to a decentralized control. However, beyond the individual decision problems, we must now take the final step of performing the value-cost comparison and deciding what information should be collected by which agent. Agents can now either take on roles of buyers (who pay to receive information collected by other agents) and sellers ("helper agents" that incur cost to provide information). These additional areas of computation require visibility over the entire system, so we can introduce the role of a "market-maker", who compiles information about waypoint bundle values and corresponding costs for helper agents to assist and then performs the computation necessary to assign helper agents to bundles. A similar design has been utilized in previous system architectures, and we will draw inspiration from this previous work.[8] Note that for the purposes of my evaluation, I ran simulations on completely synthetic grid environments rather than a framework that allows for publish/subscribe commands, such as the ROS Construct.[16] I therefore used centralized control for convenience, though in a decentralized scheme I could have randomly designated an agent to perform the "market-maker" role.

Now that we have expressions for value of information within our system, we would like to be able to ascertain which points to actually acquire. We are working with the following inputs: a set of n agents, each with an initial position and goal position, m individual waypoints available for acquisition, a function for value of information (VOI) at a given waypoint bundle for a given agent, and a function for cost of information (COI) for a given agent and waypoint. We will consider the agents to take on roles of buyers and sellers, and the waypoints are the goods to be purchased. Buyers will wait until sellers collect information before using it to replan new optimal paths. We would like a function that, given these inputs, outputs an assignment of sellers to waypoint traversals such that information is collected in a way that maximizes the expected utility of all agents.

However, it is first necessary to make some slight adjustments to ground the single agent decision problems in the context of the multi-agent system. Note that previously, value was assigned to waypoint bundles purely based on evaluation of the single agent in question. However, since all information is held in common within a multi-agent system and we are currently exploring a cooperative setting whereby navigation is augmented via this distributed sensing and exploration, we now characterize the value of each bundle of waypoints W_i to buyers b as:

$$V(W_i) = \sum_b V_b(W_i) \tag{3.1}$$

We also note that if all agents are allowed to take on roles of buyers and sellers without strict separation and, for example, all agents receive a waypoint assignment, the resulting movement of agents to collect information at those waypoints would alter the optimality of that assignment since the initial positions from which agents recompute optimal paths has changed. To remedy this, given an environment setup, we perform a random assignment of each agent to the role of either buyer or seller. Note that in the case of simulating drone sellers, this corresponds to randomly selecting some of the agents to be drones.

We can formulate our optimization problem as follows:

$$\text{argmax}_{\{w_s \forall s\}} \quad [\sum_b EU(p_b | \bigcup_s w_s) + \sum_s EU_s(p_{w_s})] \tag{3.2}$$

$$\text{s.t.} \quad \bigcup_s w_s \subseteq W \tag{3.3}$$

$$\bigcap_s w_s = \emptyset \tag{3.4}$$

where w_s is the subset of W uniquely assigned to be traversed by seller s, $EU(p_b | \bigcup_s w_s)$ is the expected utility of buyer b's updated best path to its destination given information on the set of waypoints $\bigcup_s w_s$, and $EU_s(p_{w_s})$ denotes the expected utility for seller s to traverse from its initial location, through w_s, and to its destination.

In order to translate this optimization problem into the language of VOI and COI for implementation, we note that $V(W)$ is a measure of the expected gain in utility for buyers given information on the set of waypoints W, and thus a mechanism that maximizes $V(W)$ will maximize expected utility for buyers and also for the entire system. Since the cost remains unique to the individual agent, we call seller s's desired compensation for assisting with waypoint w to be $C_s(w)$. $C_s(w)$ is a measure of the expected utility penalty seller s would personally incur to collect information on w, and thus a mechanism that minimizes $C_s(w)$ will minimize expected utility penalties incurred by sellers during information collection and therefore also maximize expected utility of the system.

Under this lens, which we will employ moving forward, we can rewrite our optimization problem as follows:

$$\text{argmax}_{\{w_i\}} \quad [V(\bigcup_i w_i) - \sum_s C_s(w_i)] \tag{3.5}$$

$$\text{s.t.} \quad \bigcup_i w_i \subseteq W \tag{3.6}$$

$$\bigcap_i w_i = \emptyset \tag{3.7}$$

For the remainder of this paper, when we refer to the "objective function" , we will be referring to the$J = V(\bigcup_i w_i) - \sum_s C_s(w_i)$ maximized in Equation 3.5. Given that the task of assisting with a given bundle W has emerged and that the task assignment is able to be evaluated by comparing value with cost, there is a strong case for employing a market-based approach to this task allocation.

We will now characterize an algorithm for this approach, as well as those for several other possible approaches to arriving at an optimal solution.

For this paper, we will only allow each seller to be assigned a single waypoint. Single-waypoint assignment will serve as the baseline for further research on creating and traversing more complex bundles. We also do assignment in an offline fashion; that is, we require agents, after computation of base paths, to calculate VOI and COI based on currently available information and produce waypoint assignments to sellers. Sellers will then attempt to collect information along their optimal waypoint traversal paths, after which buyers can recompute their optimal paths and all agents will complete their paths to their destinations, if possible.

3.2 Optimal Assignment

For this algorithm, we seek to determine the optimal allocation of waypoints to sellers for a given system. This approach requires exhaustive exploration of all possible assignments of waypoints to sellers.

We will utilize following variables:

- An ordered set of all seller agents S.
- A set of possible assignment points, $A = W + \{\emptyset\}$, where $\emptyset$ designates a null value, rather than the empty set. Thus, A consists of the set of waypoints W as well as the null value, which if assigned to a seller agent will signal this seller will remain unassigned.
- The set "Perm" consisting of all permutations of selections of $|S|$ elements from A, with replacement. Perm then holds every possible assignment for the ordered set S. We can assume that Perm is determined in advance based on W and A and provided to the optimal assignment algorithm as input.

- Indicator variables x_{sa}^P, where $x_{sa}^P = 1$ if and only if seller s receives assignment a in assignment permutation P. Otherwise, $x_{sa}^P = 0$. Let $X_P = \{x_{sa}^P \forall s \in S, a \in A\}$

- The set Costs $= \{c_{sa} \forall s \in S\}$, where c_{sa} is the cost for seller agent s to collect information at assignment a. These costs are fixed for sellers all assignment algorithms introduced in this paper, we so can assume that they are calculated in advance and provided to the algorithms as inputs. Note that if $a = \emptyset$, then $c_{sa} = 0$.

- The set of all buyer agents B.

- The function $V(W_i)$, which returns the the value of bundle W_i.

Algorithm 2 Optimal Assignment

```
    Input: $S$, $P$, Costs, $B$
    Output: Optimal $X$
 1: procedure Opt-Assign($S$, $P$, Costs, $B$)
 2:     $J \leftarrow 0$
 3:     $X^* \leftarrow$ None
 4:     for $P \in$ Perm do
 5:         $C_T \leftarrow 0$
 6:         $W' = \emptyset$
 7:         for all $x_{sa}^P = 1$ do
 8:             $C_T \leftarrow C_T + c_{sa}$
 9:             if $a \neq \emptyset$ then
10:                 $W' \leftarrow W' \cup \{a\}$
11:         $V_T = V(W')$
12:         if $V_T - C_T > J$ then
13:             $J = V_T - C_T$
14:             $X^* = X_P$
    return $X^*$
```

Algorithm 2 outlines the optimal procedure. Note that although P includes permutations that may assign a waypoint to multiple agents, the optimal assignment algorithm will never return such an assignment since a higher J can be achieved by reassigning all but one of those agents to other waypoints or no waypoint.

Despite finding an optimal assignment, this algorithm requires checking $(|W| + 1)^{|S|}$ permutations, making it impractical for real-world use-cases. Therefore, we now consider several alternative algorithms that seek to achieve the same maximized objective function with less computation.

3.3 Greedy Assignment

For our first alternative algorithm, we consider a greedy approach to assignment; assignment is done iteratively for all unassigned sellers, and at each round the single best seller-waypoint pair is determined and removed from consideration for future rounds.

We will work with the following variables:

- A provisional set of assigned waypoints A, which will be initialized to the empty set.
- A set of active (still unassigned) waypoints W_a, which will be initialized to all waypoints.
- A set of active (still unassigned) seller agents S_a, which will be initialized to all sellers.
- The set of costs $\text{Costs} = \{c_{sw} \forall s \in S\}$, where c_{sw} is the cost for seller agent s to collect information at waypoint w.
- The set of all buyer agents B.
- The function $V(W_i)$, which returns the the value of bundle W_i.
- $v^A_{bw} = V_b(w|A)$, the value of information to buyer $b \in B$ for waypoint w given the provisional assignment bundle A.
- Indicator variables x_{sw}, where $x_{sw} = 1$ if and only if seller s is assigned to waypoint w in the current allocation A. Otherwise, $x_{sw} = 0$. Let $X_A = \{x_{sw} \forall s \in S, w \in W\}$.

Algorithm 3 Greedy Assignment

```
    Input: $S$, $W$, Costs, $B$
    Output: Optimal $X$
 1: procedure GREEDY($S$, $W$, Costs, $B$)
 2:     $A \leftarrow \emptyset$
 3:     $x_{sw} \leftarrow 0, \forall s \in S, w \in W$
 4:     while $|S_a| > 0$ and $|W_a| > 0$ do
 5:         for $b \in B$ do
 6:             for $w \in W$ do
 7:                 $v_{bw}^{A} = V_b(w|A)$
 8:         $w', s' \leftarrow \text{argmax}_{w,s}[\sum_b v_{bw}^{A} - \min_{s \in S_a} c_{sw}]$
 9:         if $w'$ is None then
10:             break
11:         $A \leftarrow A + \{w'\}$
12:         $S_a \leftarrow S_a - \{s'\}$
13:         $W_a \leftarrow W_a - \{w'\}$
14:         $x_{s'w'} = 1$
    return $X_A$
```

Algorithm 3 outlines the greedy procedure. This algorithm requires at most a factor of (min[S, W]BW) while loop iterations to terminate with a final assignment, a significant improvement over the optimal assignment algorithm.

3.4 Iterative Auction Assignment

For this algorithm, we will employ a market-based method, the iterative auction, to determine an assignment of waypoints to sellers. While a termination condition has not occurred, this algorithm allows for more dynamic shifting of the provisional allocation based on buyers' and sellers' best responses to market prices, terminating only once they agree on an allocation or cycling occurs.

We will work with the following variables:

- A provisional set of assigned waypoints A, which will be initialized to the empty set.

- The sets of all waypoints W, supplied waypoints W_S, and demanded waypoints W_D.
- A set of waypoint market prices $\{p_w \forall w \in W\}$, which will all be initialized to 0.
- The function $V(W_i)$, which returns the the value of bundle W_i.
- A constant price update step size δ. This is a parameter we are free to appropriately choose.
- The set of all seller agents S.
- The set of costs $\text{Costs} = \{c_{sa} \forall s \in S\}$, where c_{sw} is the cost for seller agent s to collect information at waypint w.
- The set of all buyer agents B.
- Indicator variables x_{sw}, where $x_{sw} = 1$ if and only if seller s is assigned to waypoint w in the current allocation A. Otherwise, $x_{sw} = 0$. Let $X_A = \{x_{sw} \forall s \in S, w \in W\}$. .

Algorithm 4 outlines the iterative auction procedure. After it terminates with a final A^*, note that if multiple sellers are supplying the same waypoint in A^*, we will select one at random to supply that waypoint.

Due to the additive nature of the price updates, this algorithm is guaranteed to terminate. If some p_w ever falls to 0 or below, w will be guaranteed to be demanded by at least one buyer. If p_w ever meets or exceeds $\max(c_{sw})$, w will be guaranteed to be supplied by at least one seller. Thus, for a given set of inputs, there are finite price combinations possible, and our iterative auction algorithm is guaranteed to terminate either when buyers and sellers agree on an allocation or when a price set cycle occurs.

Algorithm 4 Iterative Auction

```
    Input: $S$, $W$, Costs, $B$, $\delta$
    Output: Optimal $X$
 1: procedure ITER-AUC($S$, $W$, Costs, $B$, $\delta$)
 2:     $A \leftarrow \emptyset$
 3:     $x_{sw} \leftarrow 0, \forall s \in S, w \in W$
 4:     $p_w \leftarrow 0, \forall w \in W$
 5:     while no cycle do
 6:         $W_S \leftarrow \emptyset$
 7:         $W_D \leftarrow \emptyset$
 8:         for $b \in B$ do
 9:             for $w \in W$ do
10:                 $v^A_{bw} = V_b(w|A)$
11:         for $s \in S$ do
12:             $w' \leftarrow \text{argmax}_w\{p_w - c_{sw}\}$
13:             if $p_{w'} - c_{sw'} \geq 0$ then
14:                 $W_S \leftarrow W_S + \{w'\}$
15:         for $w \in W$ do
16:             if $\sum_{b \in B} v^A_{bw} \geq p_w$ then
17:                 $W_D \leftarrow W_D + \{w'\}$
18:         for $w \in W$ do
19:             if $w \in W_S - W_D$ then
20:                 $p_w \leftarrow p_w - \delta$
21:             if $w \in W_D - W_S$ then
22:                 $p_w \leftarrow p_w + \delta$
23:         $A \leftarrow W_D \cap W_S$
24:         if $\{p_w\}$ seen or $W_S = W_D$ then break
    return $X_A$
```

3.5 Iterative Auction - Stabilized

Since the first iterative auction method is prone to much variation in the provisional assignment, we consider utilizing the value of the objective function for the allocation to both stabilize the algorithm and more closely approximate the assignment returned by the optimal allocation.

In addition to those variables used for Iterative Auction 1, we now include the following:

- $J = J(A)$ equalling the value of our objective function given provisional allocation A.
- The number of rounds to run the algorithm, T.

Algorithm 5 outlines the stablized iterative auction procedure. Note that, as before, we will randomly select an assigned agent to assist with any waypoint that has been assigned more than one agent after the algorithm has returned.

The new feature introduced to the method begins on Line 26, where we first check to see if the objective value achieved by the latest allocation is greater than the best found so far and, if so, updating A. This allows the algorithm to make gradual progress toward finding the best allocation.

Algorithm 5 Iterative Auction Stable

```
    **Input:** $S$, $W$, Costs, $B$, $\delta$, $T$
    **Output:** Optimal $X$
 1: procedure ITER-AUC-STABLE($S$, $W$, Costs, $B$, $\delta$, $T$)
 2:     $A \leftarrow \emptyset$
 3:     $x_{sw} \leftarrow 0, \forall s \in S, w \in W$
 4:     $p_w \leftarrow 0, \forall w \in W$
 5:     $J \leftarrow J(A)$
 6:     $J_{\max} = J$
 7:     $t = 0$
 8:     while $t < T$ do
 9:         $W_S \leftarrow \emptyset$
10:         $W_D \leftarrow \emptyset$
11:         for $b \in B$ do
12:             for $w \in W$ do
13:                 $v_{bw}^A = V_b(w|A)$
14:         for $s \in S$ do
15:             $w' \leftarrow \mathrm{argmax}_w \{p_w - c_{sw}\}$
16:             if $p_{w'} - c_{sw'} \geq 0$ then
17:                 $W_S \leftarrow W_S + \{w'\}$
18:         for $w \in W$ do
19:             if $\sum_{b \in B} v_{bw}^A \geq p_w$ then
20:                 $W_D \leftarrow W_D + \{w'\}$
21:         for $w \in W$ do
22:             if $w \in W_S - W_D$ then
23:                 $p_w \leftarrow p_w - \delta$
24:             if $w \in W_D - W_S$ then
25:                 $p_w \leftarrow p_w + \delta$
26:         $A' \leftarrow W_D \cap W_S$
27:         $J = J(A')$
28:         if $J > J_{\max}$ then
29:             $A \leftarrow A'$
30:             $J \leftarrow J_{\max}$
31:         if $W_S = W_D$ then break
32:         $t \leftarrow t + 1$
    return $X_A$
```

4 Empirical Evaluation

The objective of this empirical evaluation is to determine the impact our assignment algorithms have on the total utility outcome of multi-agent systems. From generating abstract, centralized simulation environments, we compare the performance of our four information exchange settings – greedy assignment, iterative auction, stabilized iterative auction, and optimal assignment – with that of the setting in which agents do not exchange information.

4.1 Dataset

We work with randomly generated 5×5 grid environments where the occupancy status of each cell is set to be free/unknown/blocked with $40\%/30\%/30\%$ chance, respectively. All unknown cells are associated with the same probability of being blocked, and we test settings where $\alpha_w = 0.5$. 4 agents are placed on the grid and randomly divided into sellers and buyers (requiring at least one buyer). Each agent is initially positioned on a starting cell and assigned a destination cell, with reward at each destination set to 25.

We consider two different "team compositions":

1. **All-UGV:** All agents will fail upon attempting to step on blocked cells, remaining at the position immediately preceeding failure. We can imagine this setting to consist entirely of risk-prone UGVs. Note that this means if sellers with assignments fail before finishing information collection, they will communicate this so that buyers do no unnecessarily delay their path re-computation.

2. **Drone-Sellers:** Seller agents are able to traverse blocked cells without failing, and do not have assigned destinations. We can imagine this setting to consist of UGV buyers and "drone" sellers, who are employed only to move between waypoints, resolving the occupancy status of any other unknown points along their paths. In computing their cost of collecting information, they are only concerned with the minimum step-cost path from their current position to a given waypoint.

We generated environments that meet these conditions for both team compositions, and then operated the assignment settings according to the following procedure:

1. After all non-drone agents compute base paths, each setting produces assignments based on its corresponding algorithm.

2. Assigned helper agents determine an optimal path to collect information and begin to complete traversal. Drone sellers move to their assigned waypoints only, whereas UGV sellers move to the waypoint and then to their destinations, if possible.

3. After all assigned waypoints have their information collected or their assigned sellers communicate failure before collection is completed, buyers recompute optimal paths to their destinations with all information collected in the previous step. This includes information on waypoints traversed by sellers other than the explicitly assigned waypoints.

4. Buyers and sellers traverse the remainder of their paths until they reach their destinations or fail.

Note that there are two phases of path recomputation, once for assigned helper agents to determine waypoint traversal paths and once for buyers to recompute optimal paths with new information. Agents are bound to the traversal of paths until permitted to recompute. In the no-assignment setting, agents simply traverse their base paths until destinations are reached.

The code for this evaluation and all algorithms cited in this paper can be found in this GitHub repository.

4.2 Setup

We test five settings for multi-agent navigation with expected utility path-planning, one no-assignment setting and four assignment settings:

1. **No Assign**

2. **Greedy Assign** information exchange

3. **Iterative Auction** information exchange

4. **Iterative Auction-S** (stabilized) information exchange

5. **Optimal Assign** information exchange

Note that for the purposes of evaluation, we utilized Algorithm 6, a simpler version of pathfinding Algorithm 1, which utilizes a priority queue structure. Due to this, it is less optimal than Algorithm 1 but has a faster runtime.

Algorithm 6 Empirical Simulations Pathfinding Algorithm

```
    Input: Graph $G$ with vertices $V$, source node $s$, destination node $d$, and reward $R$
    Output: Max expected utility $U(d)$ and corresponding path $P^*$
 1: procedure MOD-DIJKSTRA-S(G,V,s,d,R)
 2:     for $x \in V - \{c\}$ do
 3:         $U(x) \leftarrow -\infty$
 4:         $prev[x] \leftarrow$ None
 5:     $U(c) \leftarrow R$
 6:     $A \leftarrow \{\}$
 7:     while $A \neq V$ do
 8:         pick x not in A with largest U(x)
 9:         $A' \leftarrow A \cup \{x\}$
10:         for out-neighbors y of x do
11:             if $U(y) > U(x|$“xy” next edge$)$ then
12:                 $U(y) \leftarrow U(x|$“xy” next edge$)$
13:                 $prev[y] \leftarrow x$
    return U(d), getPath(prev[],s,d)
```

Over several thousand simulation runs for different environment setups, we tracked the average expected value of the objective function, average total utility, average total cost to all agents, average percent of non-drone agents successfully reaching destinations.

4.3 Results

Simulation results are summarized in Table 4.1. Since average total utility achieved in each setting characterizes overall performance of each approach, we note that regardless of the team composi-

Table 4.1: Simulation Results

	No Assign	Greedy Assign	Iterative Auction	Iterative Auction-S	Optimal Assign
			All-UGV Team		
Average Expected J	–	1.2067 ± .0117	1.0699 ± .0108	1.1729 ± .0114	1.1742 ± .0367
Average Total Utility	70.8714 ± .1089 –	72.8298 ± .0998 (+2.76%)	72.7024 ± .1009 (+2.58%)	72.9175 ± .0997 (+2.88%)	72.5379 ± .3231 (+2.35%)
Average Total Cost	8.9241 ± .0120	9.1358 ± .0124	9.1711 ± .0126	9.1455 ± .0124	9.1075 ± .0398
Average Success %	79.79% ± .11%	81.96% ± .10%	81.87% ± .11%	82.06% ± .10%	81.64% ± .33%
			Drone-Sellers Team		
Average Expected J	–	1.6426 ± .0126	1.3583 ± .0115	1.4611 ± .0121	1.6647 ± .0402
Average Total Utility	35.4185 ± .0951 –	37.5619 ± .0922 (+6.05%)	37.3231 ± .0928 (+5.37%)	37.3850 ± .0929 (+5.55%)	37.3228 ± .2932 (+5.37%)
Average Total Cost	4.4782 ± .0115	5.4889 ± .0142	5.4226 ± .0143	5.4292 ± .0142	5.4856 ± .0448
Average Success %	79.59% ± .14%	86.15% ± .12%	85.39% ± .13%	85.49% ± .13%	85.69% ± .41%

Notes: Results provided for environments where all unknown cells are initialized with 0.5 probability of being blocked. Settings "No Assign" through "Iterative Auction-S" are compiled from 50,000 simulation runs, and optimal assignment from 5,000 runs, all using the same random seed. "Average Expected J" denotes the average value of the objective function in assignment settings. The first 4 rows of results correspond to the settings with teams of All-UGV's, and the last 4 rows of results to settings with teams whose seller agents are drones. Standard error is reported along with mean values. For the "Average Total Utility" rows, the percent difference between the empirical mean for a given setting (column) as compared with the "No Assign" setting in the same row is provided in parentheses. For example, with an All-UGV Team, the "Greedy Assign" setting resulted in a 2.76% larger average total utility compared to the "No Assign" setting.

tion, each of the assignment settings, on average, exhibited a statistically significant improvement in total utility of the outcome, based on the confidence intervals achieved. "Iterative Auction-S" also exhibited a statistically significant out-performance of "Iterative Auction". How "Greedy Assign" and "Iterative Auction-S" perform in relation to each other and when compared to "Optimal Assign" is less clear, though, and leaves room for further work to characterize the competitiveness of each algorithm in relation to optimal assignment, as well as any other performance guarantees. It is also important to note that while optimal assignment seems to under-perform slightly, this setting was only simulated for 5000 runs, whereas the others are reported for 50000 runs. Optimal assignment's runtime prohibits it from being employed efficiently in even slightly larger settings, especially for higher densities of unknown points. Therefore, "Optimal Assign's" results are reported only to establish a significant difference in average total utility compared to "No Assign".

To explain the differences in average total utility, we note that for all assignment settings, although the average total cost incurred by all agents (via step-wise motion of all agents until termination) was higher than for "No Assign", this is compensated for by the higher rate of success in non-drone agents, for both team compositions. This confirms our hypothesis that, especially in environment setups like the one investigated here where reaching the goal is weighted heavily in the utility formulation, incurring some additional cost in order to determine shorter, less risky paths results in better system performance.

We also note that systems with drone sellers exhibit an even larger improvement in performance in assignment settings. Although the drone seller teams have lower average total (all buyers and sellers) cost, the simulations actually showed that the increase in this cost in assignment settings compared to no assignment was actually larger compared to the additional cost incurred in assignment settings with all-UGV teams. Therefore, the improvement in performance of the drone-seller teams over the all-UGV teams is entirely explained by the improvement in the success rate of non-drone agents. When we compare the Average Expected J between the two team compositions, we see that drone-seller teams allowed for a statistically significant improvement in this metric. Thus, as hypothesized, employing drone sellers allows for more profitable exchanges of information due to the drones lower risk aversion and better specialization for collecting information under uncertainty.

5
Conclusion

Creating multi-agent systems robust to uncertainty and failure risk is key to moving the field forward and tackling real-world applications. Inspired by the challenges present when deploying multiple robots to navigate an environment with an incomplete occupancy map, we created an expected utility path-planning heuristic, determined the prices at which information exchange can occur between agents, and created a suite of algorithm options to test for performance improvements. The results show significant promise for two team makeups (differentiated on the basis of risk aversion of agents) using the assignment algorithms to profit from collection of additional information. These algorithms ultimately improve the global outcome of the system over a no-assignment baseline.

Further work can test the algorithms robustness to additional environment setups, and specifically see how these algorithms scale or would perform in more realistic domains. Given the success of the expected utility path planning, an implementation in ROS could be fruitful and allow for a truly decentralized market setup. Information remains key to developing robust autonomous coordination, so continuing to examine the ways we can exploit multi-agent systems to reduce uncertainty will continue to yield diverse and fruitful research insights.

References

[1] Arvanitakis, I., Tzes, A., & Giannousakis, K. (2017). Mobile robot navigation under pose uncertainty in unknown environments**this work has received funding from the european union horizon 2020 research and innovation programme under the grant agreement no. 644128, aeroworks. *IFAC-PapersOnLine*, 50(1), 12710–12714. 20th IFAC World Congress.

[2] Asadpour, A., Goemans, M., Mądry, A., oveis gharan, S., & Saberi, A. (2017). An o (log n /log log n)-approximation algorithm for the asymmetric traveling salesman problem. *Operations Research*, 65.

[3] Borradaile, G. (2021-03-20). Dijkstra's algorithm: Correctness by induction.

[4] Botelho, S. & Alami, R. (2000). : (pp. 55–68).

[5] Cai, A., Fukuda, T., Arai, F., & Ishihara, H. (1996). Cooperative path planning and navigation based on distributed sensing. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 3 (pp. 2079–2084 vol.3).

[6] Cantelli, L., Presti, M., Mangiameli, M., Melita, C., & Muscato, G. (2013). Autonomous cooperation between uav and ugv to improve navigation and environmental monitoring in rough environments.

[7] Dias, M. & Stentz, A. (2004). Traderbots: a new paradigm for robust and efficient multirobot coordination in dynamic environments.

[8] Gerkey, B. P. & Mataric, M. (2002). Sold!: auction methods for multirobot coordination. *IEEE Trans. Robotics Autom.*, 18, 758–768.

[9] Gerkey, B. P. & Matarić, M. J. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9), 939–954.

[10] Howard, R. A. (1966). Information value theory. *IEEE Transactions on Systems Science and Cybernetics*, 2(1), 22–26.

[11] Lazanas, A. & Latombe, J.-C. (1995). Motion planning with uncertainty: a landmark approach. *Artificial Intelligence*, 76(1), 287–317. Planning and Scheduling.

[12] Likhachev, M. & Stentz, A. (2009). Probabilistic planning with clear preferences on missing information. *Artificial Intelligence*, 173(5), 696–721. Advances in Automated Plan Generation.

[13] Parker, L. (1994). : (pp. 776 – 783 vol.2).

[14] Parker, L. (2008). *Multiple Mobile Robot Systems*, (pp. 921–941).

[15] Parkes, D. & Ungar, L. (2000). : (pp. 74–81).

[16] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., & Ng, A. (2009). Ros: an open-source robot operating system. volume 3.

[17] Rassenti, S. J., Smith, V. L., & Bulfin, R. L. (1982). A combinatorial auction mechanism for airport time slot allocation. *The Bell Journal of Economics*, 13(2), 402–417.

[18] Simmons, R. & Koenig, S. (1995). Probabilistic robot navigation in partially observable environments. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95 (pp. 1080–1087). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

[19] Smith (1980). The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12), 1104–1113.

[20] Stipes, J., Hawthorne, R., Scheidt, D., & Pacifico, D. (2006). Cooperative localization and mapping. In *2006 IEEE International Conference on Networking, Sensing and Control* (pp. 596–601).

[21] Takeda, H. & Latombe, J. . (1992). Sensory uncertainty field for mobile robot navigation. In *Proceedings 1992 IEEE International Conference on Robotics and Automation* (pp. 2465–2472 vol.3).

[22] Vries, S. D. & Vohra, R. (2000). Combinatorial auctions: A survey.

[23] Werger, B. & Mataric, M. (2000). Broadcast of local eligibility: Behavior-based control for strongly cooperative robot teams.

[24] Zilberstein, S. & Lesser, V. (2003). Intelligent information gathering using decision models.