Learning Situated-Decision Strategies for Collaborative Navigation

Raj Korpan RKORPAN@GRADCENTER.CUNY.EDU

Department of Computer Science RAJ.M.KORPAN@GMAIL.COM

The Graduate Center, City University of New York

Abstract

A key task in human-robot collaboration is *collaborative navigation*, where an autonomous mobile robot and a human companion travel together to a shared destination. This proposal focuses on three challenges in collaborative navigation: how to tailor a robot's behavior to its experience, how to ensure the human companion's comfort with and trust in the robot's behavior, and how the robot can explain its decision making in natural language. Of course, it is not feasible to hard code each possible situation that the robot may face. Instead, to address the first challenge, the proposed approach learns to generalize the robot's experiences into salient situations, and then learns specialized decision-making strategies for each of those situations. To address the second challenge, this proposal hypothesizes that a cognitive basis for a robot's decision making will result in natural interaction with people. To that end, the proposed work incorporates decision-making rationales from cognitive science and human-robot interaction, a novel voting-based multi-objective path planner inspired by human path-planning behavior, and an original metaheuristic based on human search-and-rescue behavior as a cognitively-based alternative to A* search. Finally, to address the third challenge, this proposal describes an innovative method to explain navigation decisions in natural language.

Experiments for evaluation and human-subject studies are outlined. The potential contributions of this work have broad implications for human-robot collaboration and for machine learning.

Keywords: Autonomous robot navigation, human-robot collaboration, situated cognition, weight learning

Table of Contents

List of Figures

List of Tables

List of Algorithms

1. Introduction

Autonomous robots are artificial, mobile, mechanical agents that are increasingly prevalent in modern society. Eventually they will be expected to complete tasks independently alongside and in collaboration with people (Bauer et al., 2008). These robots will navigate as they perform such tasks in factories, warehouses, offices, and homes (Kruse et al., 2013). In *autonomous robot navigation* a robot moves through an environment from one location to another. In contrast, in *collaborative navigation* an autonomous robot and a person travel together to some destination. Modern robot navigators are not focused on collaborative navigation and their opaque underlying architectures may be difficult to understand. The work proposed here addresses these issues. It draws inspiration from three lines of research: metaheuristics (high-level heuristic techniques), cognitive models (computational simulations of human behavior), and models of human-robot interaction.

The thesis of this proposal is that, in complex and diverse environments, an autonomous robot can improve both its navigation performance and its impact on human collaborators when it recognizes and adapts to situations through a properly balanced, diverse set of transparent rationales. In a *situation*, a robot makes a decision in a given environment given its capabilities, its current sensor readings, and its knowledgebase (Giannopoulos et al., 2014). In traditional path planning, a decision's complexity is defined by its *branching factor*, the number of actions the robot can take in any given state. A situation, however, is richer; it also considers the robot's spatial abilities and decision-making strategies, environmental factors, and other information available in its knowledgebase. For example, a situation may be a place in the environment similar to other previously visited places, such as when the robot faces a corner or stands in a doorway.

Because planning an optimal path in a non-trivial environment is NP-hard, both people and computational approaches for navigation use *heuristics*, efficient strategies that can often solve a problem (Pearl, 1984; Canny, 1988). Human navigators use multiple heuristics and change their strategy across and within tasks (Iaria et al., 2003). Similarly, computational methods, such as metaheuristics, use a heuristic strategy to solve problems and combine or select among different heuristics. Neither heuristic-based computational approaches nor models of human behavior, however, have addressed situation-specific approaches. A *situated-decision strategy* combines heuristics, metaheuristics, and/or hybrid metaheuristics to produce an approximate solution in response to a particular situation. Thus, two novel ideas can be investigated within the context of collaborative navigation: how to identify unique and meaningful situations, and how to identify an appropriate situated-decision strategy.

At a high level, this proposed work will allow a robot to identify its current situation, use a situated-decision strategy to decide which action to take, and communicate this understandably to its human collaborator. This proposed work targets three research goals:

- Construct and implement methods to learn and identify situations. These methods will group the robot's experiences into meaningful, useful clusters, and efficiently classify new experiences.
- Construct, implement, and evaluate situated-decision strategies based on metaheuristics, weight learning, cognitive modeling, and path planning.
- Implement and evaluate natural language generators to explain navigation decisions, and situated-decision strategies.

The potential significance of this proposed work is threefold: it will allow a robot to learn to specialize its behavior based on its context, it will learn to combine a variety of decision-making rationales, and it will improve the robot's ability to communicate with people about its behavior and its perception of the environment. This proposal draws inspiration from human navigation behavior; it allows a robot to represent and reason about space similarly to the way people do, and thereby facilitates human-robot collaboration (Kennedy et al., 2007). The proposed work combines diverse rationales and seeks to balance reactivity with deliberation and to balance exploration with exploitation. Furthermore, enriched communication potentially improves human comfort with and trust in the robot. The resultant system is expected to be more robust to real-world challenges. This work will be evaluated both on computational performance and on human impact.

My preliminary work has formulated natural-language explanations for the robot's navigation decisions and navigation plans. I have contributed to the development of MengeROS, a system that simulates both a crowd of pedestrians and a robot, that will be used to evaluate the proposed work in a variety of test scenarios. Also, I contributed to a novel Bayesian approach for path planning in a crowded environment.

- Korpan, R., Epstein, S. L., Aroor, A., and Dekel, G. (2017). WHY: Natural explanations from a robot navigator. In *Proceedings of AAAI 2017 Fall Symposium on Natural Communication for Human-Robot Collaboration*
- Korpan, R. and Epstein, S. L. (2018). Toward natural explanations for a robot's navigation plans. In *Proceedings of Workshop on Explainable Robotic Systems at HRI 2018*
- Aroor, A., Epstein, S. L., and Korpan, R. (2017). MengeROS: A Crowd Simulation Tool for Autonomous Robot Navigation. In *Proceedings of AAAI Fall Symposium on Artificial Intelligence for Human-Robot Interaction*, pages 123–125. AAAI
- Aroor, A., Epstein, S. L., and Korpan, R. (2018). Online learning for crowd-sensitive path planning. In *Proceedings of the 17th Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18. International Foundation for Autonomous Agents and Multiagent Systems

The remainder of this chapter formalizes the robot's task, collaborative navigation. It begins with fundamental concepts and defines key terms. It then discusses challenges related to the proposed work.

1.1 Definitions

A robot is an *embodied* agent, one that interacts with the environment through its physical body. The robot perceives its environment through *sensors* and acts upon that environment with *actuators* (Russell and Norvig, 2009). Examples of robot sensors include cameras and lasers; an example of a robot actuator is a motor. *Actuator error* occurs when actuators do not execute an action precisely. *Sensor error* occurs when there is noise in the sensors' signal.

Situated cognition is a psychological theory; it posits that knowledge is inseparable from the activity, context, and culture in which it was learned (Brown et al., 1989; Clancey, 1997). Situated cognition argues that a robot's knowledge emerges from its interaction with the environment.

At any instant, a mobile robot's *pose* $\langle x, y, \theta \rangle$ in its 2D environment is its location $\langle x, y \rangle$ and its orientation θ with respect to some allocentric coordinate system. A robot *navigation problem* is $\mathcal{P} = \langle S, I, A, G \rangle$ where

- S is a set of states that represent an instance of the environment. Each *state* $s = \langle x, y, \theta \rangle \in S$ represents a possible robot pose.
- $I \subseteq S$ is a set of *initial states*, poses at which the robot may begin.
- A is a set of possible actions from which the robot can select. Each *action* $a \in A$ is intended to change the robot's pose from state s to another state s'.
- $G(s)$ is a Boolean goal test that returns true if the robot's current location is a *target* (a location $t = \langle x, y \rangle$).

For example, the initial state may be the robot's pose at a charging station or at the entrance to a building. The robot uses its actuators to perform actions (e.g., a forward movement or a turn). If an action fails or is a deliberate pause, s may be the same as s'. (A glossary that summarizes the notation used in this paper begins on page 109, before the references.)

Given a navigation problem $\mathcal{P} = \langle S, I, A, G \rangle$, a *path* p is a finite ordered sequence of interleaved states and actions $\langle s_1,\ a_1,\ s_2,\ a_2,\ s_3,\ \ldots,\ s_{y-1},\ a_{y-1},\ s_y \rangle$. An example of a path is s_1 = initial pose, a_1 = move forward two feet, s_2 = next pose, a_2 = turn left 90°, s_3 = next pose, and a_3 = move forward five feet. The robot seeks a *solution* to $\mathcal{P}$, a finite path $p = \langle s_1,\ a_1,\ s_2,\ \ldots,\ a_{k-1},\ s_k \rangle$ from an initial state $s_1 \in I$ to a goal state s_k, where $G(s_k) = True$.

Step cost is a metric defined on an action a in state s. Examples of step costs include the amount of energy consumed, the time taken, or the distance traveled. Step cost may be uniform across all state-action pairs, or may be defined in the context of the problem. The *path cost* of path p is the sum of the step costs of all the actions along p, for example, the total power consumption or distance traveled.

The *search space* H for a navigation problem $\mathcal{P}$ is the set of all paths that start at an initial state. Given a step cost metric, an *optimal solution* o is a solution in the search space with minimum path cost: $o = \underset{p \in H}{\arg\min}\ PathCost(p)$. An optimal solution may, for example, be the fastest or the shortest path from the initial position to the target. A *satisfactory solution* is a solution that is good enough with respect to some domain-specific criterion, and thus is less likely to require prohibitive computational resources (Poole and Mackworth, 2010). Satisfactory solutions are usually suboptimal, such as a sufficiently short path or a sufficiently fast one. *Search* explores H to find a solution.

A *plan* P is a path that can be proved to be a solution before it is executed. A *waypoint* is a location extracted from a state in a plan. Henceforward, a plan refers to a finite ordered sequence of waypoints, which excludes the orientation for each state and the actions between states. Such a plan represents the locations through which the robot intends to pass during travel on its way to a target t. *Path planning* is the search for a plan that minimizes some domain-specific criterion, such as travel time, travel distance, or resource consumption (Fong et al., 2015).

An autonomous robot navigation system (*robot controller*) can plan ahead to reach its target or make its decisions reactively, one action at a time. A modern robot controller does both. *Deliberation* formulates plans in advance to capitalize on the robot's knowledge, while *reactivity* senses and responds to the robot's environment. A *hybrid* robot controller integrates the flexibility and robustness of reactivity with the foresight of deliberation. The next section discusses challenges related to the proposed work.

1.2 Challenges

In real-world navigation there are infinitely many states, paths can double back on themselves to form cycles, and many paths do not reach a goal state. Thus the search space may contain infinitely many paths that start at a given $s \in I$. Because computation of an optimal plan in such a large search space is intractable, heuristic techniques seek satisfactory ones instead. A particular challenge for heuristics, however, is *premature convergence*, when search halts at a poor quality solution. Henceforward, the approaches discussed in this proposal all seek satisfactory solutions. Another challenge for path planning is that, although a plan is proved to be a solution before it is executed, actuator error may cause a robot to deviate from its plan and follow a different path.

This proposal does not address some important challenges to robot navigation: localization, mapping, obstacle avoidance, and motion control. *Localization* requires an autonomous robot to detect its current pose despite sensor error. *Mapping* requires the robot to construct a metric map of an unknown environment. *Obstacle avoidance* requires the robot to move through its environment without collisions. *Motion control* requires the robot to manipulate its actuators to perform intended actions despite actuator error. This proposal builds upon state-of-the-art solutions to those challenges and focuses instead on challenges specific to collaborative navigation.

A *static* environment does not change while a robot decides on an action. In a *dynamic* environment, obstacles, other robots, people, and the structure of the environment itself can all move or change over time. In a *real-world* environment a robot must contend with these dynamics. A hybrid robot controller reactively repairs or abandons its plan when people move or unanticipated obstacles appear, much the way people experience and move through space (Spiers and Maguire, 2008). As a result, the robot can plan a route to its goal, travel along that route, and manage the unexpected. If the environment changes while the robot executes its plan, a hybrid robot controller should react until it can resume its plan or formulate a new one. In particular, the presence of people in the environment presents unique challenges to collaborative navigation.

Most robot controllers do not consider their impact on people or adapt to human behavior. Instead, they search for satisfactory solutions and treat people the same way they treat other dynamic obstacles. A collaborative robot, however, must also build trust with people, respect social norms, and produce human-like motion (Kruse et al., 2013). In addition, a collaborative navigator must address the physical nature of travel alongside a person, that is, it should maintain a sufficient level of comfort and safety for its companion while it navigates (Rios-Martinez et al., 2015). Another challenge to collaborative navigation is that the robot should adapt to an individual person's behavior. To the best of my knowledge, no single system has yet addressed all of these challenges.

The remainder of this proposal is organized as follows. Chapter 2 provides background and reviews related work on human-inspired and human-aware robot navigation, as well as metaheuristics and weight learning. Chapters 3 and 4 describe the proposed work in more detail. Finally, Chapter 5 summarizes the expected contributions of this proposed work.

2. Related work

Three areas of research are related to the proposed work: systems that take inspiration from human behavior, systems designed to address travel in environments with people, and metareasoning to improve heuristic-based learning. This chapter reviews each of these to provide a context for the proposed work. Each section also discusses the gap that my proposed work addresses.

2.1 Human-inspired robot navigation

Biological and psychological explanations for human navigation can be useful starting points for robot controllers (Werner et al., 1997). *Cognitive science*, the study of the mind and intelligence, includes *spatial cognition*, the study of navigation behavior and spatial problem solving for navigation (Wolbers and Hegarty, 2010). Cognitive scientists construct computational models that seek to explain human behavior (Friedenberg and Silverman, 2011). Because robust human navigation ability is well-studied, this section first reviews results from spatial cognition research. It then describes computational models of human navigation behavior, which provide a strong basis for robot navigators in complex, dynamic environments. My proposed work will draw from the work discussed in this section to learn more sophisticated spatial representations and incorporate peoples' navigation strategies. The related methods described in Chapter 3 and 4 seek to improve navigation performance, and make decision-making rationales more transparent.

A *decision* is a choice among a set of alternatives; *decision making* selects an alternative. *Reasoning* draws a conclusion from information to solve a problem or make a decision (Leighton, 2004). People use heuristics to make fast and frugal decisions, especially when they have limited time, knowledge, and computational power (Gigerenzer et al., 1999). Gigerenzer and colleagues propose three types of heuristics: those that guide the search for alternatives, those that determine when to stop the search, and those that make a decision given the results of the search. They also suggest that people employ *cognitive economy*, that is, they tend to employ the heuristic with the least cognitive cost.

The spatial environments where people navigate are complex and dynamic. Because people rarely have perfect information about the environment, they are unlikely to make optimal decisions. Nonetheless, they travel through challenging environments with good-enough decisions (Conlin, 2009). People use a variety of heuristics to search for paths and to make decisions when confronted with unknown environments and unanticipated obstacles. Most human navigation heuristics are triggered by either external information in the environment or an internal signal.

People choose heuristics with respect to the problem they confront, in other words, human behavior is goal-directed (Aarts and Elliot, 2012). Consider, for example, one person who travels to a hospital emergency room and another who walks through a park to enjoy the scenery around a lake at the center of the park. Both must navigate from their current location to a target location, but each of them employs significantly different strategies to solve that problem. In addition, a person's navigation strategy when she travels through an environment differs from her path-planning strategy and from the way she gives route directions to someone else (Hölscher et al., 2011).

Within a particular navigation problem, a person also may apply different strategies at different points in the task. Scans of brain activity suggest that people shift between different strategies during a navigation task (Iaria et al., 2003). For example, the person in the park may follow a trail at a leisurely pace but later leave the trail to avoid a fallen tree. While on the trail, her strategy could be to walk at a normal pace and stay in the center of the trail. When she leaves the trail to avoid the fallen tree, however, her strategy may change to move more slowly and more cautiously through the brush. People can choose their strategies with respect to their overall goal, their current state, and their environment. In other words, their behavior is situated.

People also acquire spatial knowledge as they navigate, and use it to reason and to build internal representations. One type of spatial knowledge that can be used to localize and plan paths is a *landmark*, an interesting and meaningful area in the environment (Richter and Winter, 2014). As people move through an environment, they also acquire *route knowledge*, an egocentric sequence of locations and landmarks along a path, and *survey knowledge*, the spatial layout of the environment, including relations between locations, from an allocentric perspective (Latini-Corazzini et al., 2010).

A *cognitive map* is a compact, meaningful mental representation of an environment, built by a person as she moves through that environment (Golledge, 1999). Rather than try to remember and reason over all the sensory input from her environment, a person reasons from a cognitive map to reduce her cognitive load. A cognitive map incorporates landmarks, route knowledge, and survey knowledge (Tversky, 1993). Landmarks represent locations in the map, routes represent lines that connect locations in the map, and survey knowledge indicates the spatial relations in the map. Although it has been suggested that cognitive maps use metric distances and angles (Gallistel, 1990), more recent work indicates that cognitive maps have a non-metric, qualitative topological structure (Foo et al., 2005). Other recent work suggests that people use a cognitive graph with labeled metric information (Chrastil and Warren, 2014; Warren et al., 2017). The exact nature of cognitive maps remains an important open problem in spatial cognition (Weisberg and Newcombe, 2016).

People also use external information, such as maps, photos, verbal descriptions, or route directions, as heuristics to form an internal representation of the environment prior to navigation (Pazzaglia and De Beni, 2001). The use of such external information, however, can increase their cognitive load as people try to reconcile the external information with their own perception of the environment. Moreover, people's experience and knowledge from other, similar environments contribute to their internal representation in a new environment. For example, a person who enters an unknown building assumes that her previous experience of navigation in buildings and knowledge about building conventions holds there (e.g., that rooms are accessed from corridors and that elevators facilitate travel to different floors).

Cognitive models seek to simulate observed human behavior with a computational system or algorithm. To produce the desired behavior, these simulations may use logic, rules, determinism, and probability. Cognitive scientists then test the simulation in artificial settings. Early cognitive models simulated representations similar to cognitive maps. The TOUR model for multiple representations in a cognitive map incorporated route knowledge, path integration, and survey knowledge (Kuipers, 1978). TOUR found paths in a simulated, partially observable environment with relation to an external coordinate system. Later work expanded TOUR into the Spatial Semantic Hierarchy (SSH) model (Kuipers, 2000). SSH modeled a cognitive map with hierarchical metric and topological representations. It also incorporated representations of partial knowledge and uncertainty. SSH has been implemented as a robot controller on simulated robots in indoor and outdoor environments and on a physical robot in an office environment (Beeson et al., 2010). The Prototype, Location, and Associative Networks (PLAN) model also represented a cognitive map with a hierarchical structure, but from the perspective of the robot (Chown et al., 1995).

Some cognitive models have used graphs to represent spatial knowledge. A formal, logic-based model incorporated image schemata (recurring mental patterns that structure space) with affordances (what an object or environment enables people to do) into a weighted, labeled directed graph of the state space (Raubal and Worboys, 1999). Another model described a route as a series of directed segments from one place to another, and connected routes to form a graph (Werner et al., 2000). Yet another model used ACT-R, a general cognitive architecture that simulates human memory, information processing, and reasoning (Zhao et al., 2011). It integrated a route-based representation to learn new environments, and a map-based representation to improve a robot controller's ability to follow its learned routes and learn shortcuts.

A cognitive model for heuristic navigation to reach a target addressed a two dimensional simulated environment with static and dynamic obstacles (Gordon and Subramanian, 1997). It decomposed the overall task into two situations (avoid an obstacle or move toward the target) and modeled each action's consequences for each situation. The model heuristically determined which situation the robot currently faced (i.e., whether or not the robot was close to an obstacle) and then selected an action to take for that situation based on its action-consequence model. A follow-up study confirmed that people used a heuristic to recognize when to switch between situations, and introduced a model of how that strategy shifted over time with experience in the environment (Gordon et al., 1998).

To apply spatial cognition to robot navigation, one can learn and use cognitive maps and other internal representations. Early work on a robot controller integrated a grid-based metric map with a topological map, to adapt human-like internal representations of the environment for the robot (Thrun, 1998b). The grid-based map, constructed with an artificial neural network, used Bayesian updating to determine the probability that a grid cell was occupied. The topological map partitioned the grid cells into connected regions at narrow passages, such as doors. Thrun also adapted the human use of landmarks to guide navigation (Thrun, 1998a). His Bayesian approach learned the position of landmarks in the environment, trained an artificial neural network to recognize the learned landmarks, and then used the landmarks for localization.

SemaFORR is a more recent robot controller that uses commonsense qualitative spatial reasoning and incrementally learns a spatial model during travel (Epstein et al., 2015). It relies on *spatial affordances*, abstract representations of the environment, to construct a mental model that is similar to a cognitive map. SemaFORR navigates with a combination of multiple heuristics based on commonsense, path planning, the robot's sensor readings, and its spatial model. This proposal builds upon SemaFORR and its underlying cognitive architecture. SemaFORR is described further in Chapter 3.

Cognitive models have also been used to examine behavior for groups of people. One approach modeled a self-organized process for collective behavior among social beings with four characteristics: positive feedback, negative feedback, random fluctuations, and interactions among individuals (Moussaid et al., 2009). *Positive feedback* encourages individuals to mimic the behavior of nearby individuals, with likelihood proportional to the number individuals already engaged in the behavior. *Negative feedback* counteracts the positive feedback loop and causes individuals to disengage from the group's behavior. A *random fluctuation* causes an individual to engage randomly in a behavior. An *interaction* is a direct or indirect communication among individuals that relays some information.

Other work has sought to explain pedestrian movement with two cognitive models: trail formation and social force. The *trail formation model* describes the process by which pedestrians construct common, heavily-used paths (*pedestrian trails*) in an environment through indirect communication (Helbing et al., 1997). As a pedestrian travels, she is attracted to an existing pedestrian trail in proportion to its closeness and *intensity*, a measure of the frequency of travel through an area by pedestrians. For any given part of the environment, there is a maximum possible intensity, and the intensity level decays over time. A pedestrian increases the intensity associated with some part of the environment by travel through it. The *social force model* describes the motion of a set of pedestrians (Helbing and Molnar, 1995). Each pedestrian has an assigned velocity that is influenced by an attractive force toward her destination and repulsive forces away from other pedestrians. The resultant force determines the pedestrian's movement. These two models have been shown to explain pedestrian travel in the real world.

This section has discussed the internal representations (e.g., cognitive maps) that people create to reduce their cognitive load, and the external cues in the environment that people use to make decisions. It has also described cognitive models that computationally simulate human navigation behavior, and several robot controllers inspired by that behavior. These systems seek to exploit human knowledge and strategies to improve autonomous robot navigation. Only the work by Gordon and Subramanian, however, hard-coded any potential situations, and no approach learned them. Furthermore, most systems use only one navigation strategy. To the best of this author's knowledge no work has addressed or incorporated the psychological results that show people's behavior, knowledge, and learning is situated. This proposal addresses that gap in Chapter 3. The proposed work also incorporates sophisticated spatial representations into its spatial cognitive map and uses those representation to help make decisions, similar to the way people use their cognitive maps during navigation. The next section reviews approaches built to support human-robot interaction.

2.2 Human-aware and socially-aware robot navigation

Human-computer interaction (HCI) studies the design, evaluation, and implementation of computing artifacts that influence and interact with people (Lazar et al., 2017). *Human-robot interaction* (HRI) studies the design, evaluation, and implementation of robotic systems that influence and interact with people (Goodrich and Schultz, 2007). Although HRI draws significantly from HCI, its artifacts are robots, agents embodied in the real world (Scholtz, 2003). HRI research models how interactions between people and robots impact the people, how they impact the robot, and how a change in the behavior of the human or the robot affects the nature of their interactions. This proposal draws inspiration from recent work in HRI. It incorporates models of human social norms when it decides how to navigate, and methods to communicate its decision-making rationales in human-friendly language. Ideally, this will allow a robot to be accepted, trusted, and understood by a human collaborator.

Human-aware robot navigation incorporates three HRI factors into navigation functionality: comfort, naturalness, and sociability (Kruse et al., 2013). Approaches that focus on *comfort* seek to reduce people's stress and annoyance with the robot while they move safely. For example, a comfort-based approach might maintain some specified distance from nearby people within the confines of the environment. *Natural* approaches, on the other hand, try to produce human-like motion. For example, a natural approach might learn and try to imitate typical human-motion trajectories. *Sociability* addresses high-level cultural and societal norms (e.g., pass on the left). Typical human-aware robot navigation focuses on traditional *navigation performance* criteria (e.g., travel to a destination in satisfactory time and distance) but also modifies some aspect of the navigation system (e.g., action selection or path planning) to consider one or more of these HRI factors. Ideally, a robot should be comfortable, natural, and sociable. To maintain acceptable performance, however, most approaches focus on only one of these qualities.

Reinforcement learning is a machine learning paradigm that learns how to act through experience and a reward function. *Inverse reinforcement learning* learns the reward function itself. A recent approach used inverse reinforcement learning to produce human-like path trajectories for adaptive path planning in environments crowded with many moving people (Kim and Pineau, 2016). When deployed on a robotic wheelchair, the approach produced paths similar to human behavior.

One human-aware robot navigation approach learned to predict human trajectories from a human motion dataset, and used those predictions to plan safe and efficient paths in simulation (Unhelkar et al., 2015). Another approach used a heuristic, cognitively-inspired decision-making framework to produce human-like, safe, efficient navigation in a static, simulated environment (Kirsch, 2016). Other work learned a model of human motion, used it to predict how people would interact with the robot, and used those predictions to plan a safe route in a simulated crowded environment (Park et al., 2016).

While human-aware robot navigation primarily focuses on navigation performance, *socially-aware robot navigation* focuses specifically on sociability as its primary goal (Chik et al., 2016). These approaches draw upon research on social awareness, social conventions, and *proxemics*, the study of interpersonal spatial distances between people (Rios-Martinez et al., 2015). In proxemics, personal space is the space a person actively maintains around herself; intrusion into that space causes discomfort (Hayduk, 1978). Empirically, one's personal space is dynamic and situation specific (Hayduk, 1994). Proxemics also examines interpersonal space with respect to an activity, to objects in the environment, and within groups of people. Socially-aware robot navigation incorporates proxemics so that robots follow social norms when people are present.

One proxemics-based approach modeled a social-cost map around people and then planned a path with this map (Talebpour et al., 2016). This approach was evaluated in both simulation and the real world as a robot navigated among static and dynamic people. Another approach modeled human personal and social space and used it to avoid socially-unacceptable paths (Truong and Ngo, 2016). Evaluation both in simulation and with a real robot demonstrated the robot's ability to approach and avoid individuals as well as groups of people. Another recent approach used real-world data from human-human and human-robot interactions to model proxemics probabilistically (Mead and Matarić, 2017). These proxemic models were used by a reactive robot controller and a cost-based path planner to navigate around a person. They produced slightly longer paths that increased average distance from the person.

A recent socially-aware approach classified and used the emotions of nearby people to modify obstacle avoidance dynamically, both in simulation and on a real-world robot (Jiang et al., 2016). Other work used Bayesian inverse reinforcement learning to learn social norms in three large, simulated environments that varied in crowdedness (Okal and Arras, 2016). Another approach learned dynamic cost maps based on observations of human pedestrians, and then used those cost maps to plan paths (Luber et al., 2012). Those paths were more similar to human paths than those from a proxemics-based model.

Socially-aware robot navigation also uses communication to improve people's comfort with a robot. Communication allows people to build a mental model of how the robot perceives and reasons, and thereby helps to establish trust (Kulesza et al., 2013; Bussone et al., 2015). Trust in and understanding of a learning system improved when people received an explanation of why a system behaved one way and not another (Lim et al., 2009). One recent approach grounded perceived objects between the robot and a person to build a mutual mental model, and then generated natural language descriptions from it (Chai et al., 2016). Several other approaches have incorporated semantic mapping to improve robot sociability (Charalampous et al., 2017). These maps allowed the robot to perceive and describe the environment similarly to the way people do.

Another area of research is the production of natural descriptions of a robot navigator's behavior. Previously, only detailed logs of the robot's experience were available to trained researchers (Landsiedel et al., 2017; Scalise et al., 2017). In more recent work, natural language descriptions of a robot's travelled path addressed abstraction, specificity, and locality (Rosenthal et al., 2016; Perera et al., 2016). A similar approach generated path descriptions to improve sentence correctness, completeness, and conciseness (Barrett et al., 2017). Those approaches, however, used a labeled map to generate descriptions and did not explain the robot's reasoning. Other work visually interpreted natural-language navigation commands with a semantic map that showed the robot's resulting action (Oh et al., 2016). This proposal is restricted to question answering in natural language, covered in Chapter 4, without dialogue.

This section has discussed human-aware robot navigation approaches that incorporate comfort, naturalness, and sociability. It also described socially-aware systems that seek to emulate social norms and human proxemics. Although some recent approaches in human-aware navigation have successfully incorporated HRI principles, a significant challenge is the incorporation of all such criteria without a reduction in navigation performance. There is, moreover, little work on the use of cognitive models for human-aware navigation. This proposal addresses that gap in Chapter 3. Although there has been work on human-robot communication to describe a robot's navigation, no work has directly addressed explanations of a robot controller's underlying decision making. The proposed work incorporates methods to explain a robot controller's decision-making rationales in human-friendly language in Chapter 4. The next section reviews approaches that combine and balance heuristic rationales.

2.3 Metaheuristics, weight learning, and voting methods

Within machine learning and artificial intelligence, several areas focus on how to combine multiple rationales to make better decisions. This proposal addresses methods to learn situations that combine acquired knowledge and sensor information, and methods to learn situated-decision strategies that balance various decision-making rationales. My proposed work in Chapter 3, on learning situated-decision strategies, will draw from the related work on weight learning and voting methods described here. It will also implement a novel metaheuristic, inspired by human behavior, as a cognitively-based alternative for path planning. These new methods seek to improve navigation performance, and make the robot's behavior more natural.

A *metaheuristic* is a broadly applicable technique that uses a heuristic strategy to obtain satisfactory solutions (Glover and Kochenberger, 2003). Metaheuristics are typically used when only incomplete or imperfect information is available, when there are limited computational resources, or when the problem is NP-hard. Metaheuristics are not problem-specific or domain-specific; they seek satisfactory solutions through efficient search. These methods are often tailored to avoid premature convergence, with heuristics to mediate the trade-off between exploration and exploitation.

A *single-solution* metaheuristic maintains and improves one candidate solution at a time as it explores the search space. Examples of single-solution metaheuristics include simulated annealing (Kirkpatrick et al., 1983; Černý, 1985) and tabu search (Glover, 1989, 1990). A *population-based* metaheuristic maintains and improves a set of candidate solutions as it explores the search space. Many population-based metaheuristics have been inspired by biological mechanisms and the behavior of organisms (Manikas et al., 2007). One broad category of population-based metaheuristics is *evolutionary algorithms*, which use mechanisms inspired by Darwinian principles, such as survival of the fittest and natural selection, to guide search (Back, 1996). Another category is *swarm intelligence* metaheuristics, which are inspired by the crowd behavior of organisms, such as ants and bees, or the movement of particles (Bonabeau et al., 1999). Some of these methods emulate animals' search for food. An example of a swarm intelligence metaheuristic is ant colony optimization (Dorigo et al., 2006). Both evolutionary algorithms and swarm intelligence incorporate some hill-climbing and randomization.

Although metaheuristics have been used for path planning, they have several disadvantages. There is no guarantee that a satisfactory solution will be found in finite time. They can be computationally demanding and memory intensive, although rapid advances in hardware and its utilization somewhat mitigate this issue. Finally, the parameters of these approaches must be tuned by hand to enable good performance. This remains a significant issue and a major roadblock to their widespread use. To address it, there has been some work toward automated parameter tuning for heuristic algorithms (Hoos, 2011), and very limited work specific to navigation or to metaheuristics (Dobslaw, 2010).

Instead, researchers have sought to combine metaheuristics with each other or with other methods. A *hybrid metaheuristic* combines multiple heuristics and/or metaheuristics with other approaches, such as dynamic programming, machine learning, constraint programming, tree search, or problem relaxation (Blum et al., 2011). Although hybrid metaheuristics draw upon the strengths of their components, it is difficult to compare their performance for robot navigation because they have been tested with different numbers and types of robots in different, mostly simulated, environments. To contend with these issues, this proposal describes a novel population-based metaheuristic for path planning. This new approach is inspired by human behavior and will be evaluated against other metaheuristics and hybrid metaheuristics in carefully designed experiments, as described in Chapters 3 and 4.

In parallel with the development of metaheuristics, machine learning has developed methods to combine information from multiple sources. Each source is given a *weight*, a number that represents the source's contribution to the total combination. For example, weights may be used to measure path cost with multiple metrics. *Multi-objective* path planning is the search for a plan that optimizes more than one metric. Metaheuristics have been commonly used for multi-objective path planning (Ahmed and Deb, 2013; Mittal and Deb, 2007; Yang et al., 2016; Brintaki and Nikolos, 2005; Geng et al., 2013; Liang and Lee, 2015; Masehian and Sedighizadeh, 2010). These methods compute the step cost as a weighted sum of the metrics, although typically these weights are equal. This proposal suggests an alternative approach to multi-objective path planning in Chapter 3.

Weight learning selects a weight for each information source to satisfy desired criteria (Wettschereck et al., 1997). Weight learning is fundamental to supervised *classification* algorithms, where a labeled dataset is used to build a model (*classifier*) and then that model predicts labels for new, unlabeled data. This proposal addresses the use of weight learning to balance heuristics and metaheuristics.

A particular area of interest here is *ensemble methods*, machine learning algorithms that build a set of classifiers and then combine their predictions to label new data (Polikar, 2006). Ensemble methods have been shown to achieve higher classification accuracy than any individual classifier in the ensemble when the individual classifiers are *accurate* (better than random choice) and *diverse* (make different classification errors) (Kuncheva and Whitaker, 2003). Bagging and boosting, two popular ensemble methods, train individual classifiers on subsets of the data (Dietterich et al., 2000). In particular, *AdaBoost* trains a sequence of individual classifiers, where each subsequent classifier increases its focus on data thus far inaccurately classified by the earlier classifiers (Rokach, 2010).

Given a set of voters that cast a number of votes with respect to a set of *candidates* (i.e., choices), a *voting method* selects the winning candidate (Van Erp et al., 2002). Typically, an ensemble method uses voting to combine the predictions of its individual classifiers and then selects an overall predicted label in a way that best reflects the members of the ensemble. The goal of a voting method is to weigh the voters' choices to select a winning candidate that fairly balances all the voters' opinions. In addition to weight learning, the proposed work will incorporate voting methods to construct situated-decision strategies.

The simplest and most commonly used voting methods are plurality, majority, and run-off voting, described in Table 1. For example, under a plurality vote, an ensemble method would classify a new instance as a member of the class predicted most frequently by its individual classifiers. The drawback to these three approaches is that they may succumb to the *tyranny of the majority*, where the majority overrides the voice of the minority.

In *performance-weighted voting* each voter's vote is weighted by some performance criterion and then the winner is selected with some other voting method. Performance-weighted voting may also be used by an ensemble method to select a

Voting Method	Approach
Plurality	Candidate with most votes is the winner
Majority	Candidate with more than half the votes is the winner
Run-off	Top two candidates in a plurality vote face off in a majority vote

Table 1: How majority-based voting methods select a winner

Voting Method	Approach
Condorcet	Each candidate faces each other candidate in a majority vote. The candidate that wins the most of these one-on-one votes is the overall winner.
Copeland	Each candidate faces each other candidate in a majority vote. The candidate with the most net pairwise wins (one-on-one wins minus one-on-one losses) is the overall winner.
Cup	Sequences the candidates randomly from 1 to c and then holds a series of majority votes between consecutive pairs of candidates. The winner of the previous pair faces the next candidate in the sequence (i.e., $winner(i, i+1)$ faces $i+2$). The winner of the final vote is the overall winner.
Simpson	Each candidate faces each other candidate in a majority vote. Each candidate's performance in these pairwise votes is ordered based on the number of votes received, from highest to lowest. The candidate with the most votes received in its poorest pairwise performance is the overall winner.

Table 2: How preference-based voting methods select a winner

label. For example, each classifier's vote could be weighted by its prediction accuracy, or, for a Bayesian approach, by its posterior probability given the training data. Another criterion for classifiers in ensemble methods is based on the entropy of the classifier's predictions; it gives high weights to classifiers that usually predict a single class. A challenge with performance-weighted voting is that it is difficult to select the correct performance criterion for a given problem.

In an effort to weight voters' opinions more fairly, other voting methods have incorporated characteristics from them, such as preferences, ranking, approval, and scoring. A voter has a *preference* between two candidates when it selects one over the other according to some criterion (Coombs and Avrunin, 1977). A voter can *rank* all the candidates according to its preferences (Flach, 2012). Table 2 describes voting methods that use preferences to select a winning candidate (Rossi et al., 2011). Table 3 describes voting methods that use rankings to select a winner. Table 4 describes

Voting Method	Approach
Single transferable	Candidate with a majority wins. Otherwise, eliminate the candidate with the fewest votes and transfer those votes to those voters' second choice. Repeat until one candidate has a majority and is the overall winner.
Borda	Assign values to the c candidates based on the voters' rankings: a voter's first choice receives a value of $c-1$, its second choice a value of $c-2$, and so on. The candidate with the largest total value is the winner.
Nanson	Eliminate candidates with total Borda values below the average total Borda value until one winner remains.
Baldwin	Eliminate the candidate with the lowest total Borda value until one winner remains.
Coomb	Candidate with a majority wins. Otherwise, the candidate ranked last by the most voters is eliminated, this is repeated until one candidate has a majority and is the overall winner.
Bucklin	Candidate with a majority wins. Otherwise, voters' second choices are treated as additional votes. Repeat the addition of lower-choice votes until a majority candidate wins.
Dodgson	The overall winner is the candidate that wins a Condorcet vote with the fewest interchanges in voters' rankings.

Table 3: How ranking-based voting methods select a winner

voting methods where voters approve or disapprove each candidate rather than create a ranking, and Table 5 describes voting methods that allow voters to indicate their level of approval with a score.

Voting Method	Approach
Approval	Each voter approves between 1 and $c-1$ of c candidates. The candidate with most approvals wins.
Veto rule	Each voter vetoes between 1 and $c-1$ of c candidates. The candidate with fewest vetoes wins.
Inverse plurality	Each voter vetoes one candidate. The candidate with the fewest vetoes wins.

Table 4: How approval-based voting methods select a winner

Voting Method	Approach
Sum range	Each voter gives a score within a given range to each candidate, and the candidate with highest sum of scores wins.
Product range	The candidate with the highest product of its scores wins.
Cumulative scoring	Each voter distributes the same number of points among the candidates. The candidate with the most points wins.

Table 5: How scoring-based voting methods select a winner

When candidates are ordered along a continuum according some criterion, a voter can express its preferences for the candidates based on where each candidate lies on that continuum. A voter has *single-peaked preferences* when its highest preference goes to a single candidate, and its preferences for candidates on either side of the peak decrease in proportion to their distance from the peak. When all voters have single-peaked preferences and are ordered according to their most preferred candidate along that continuum, the *median voter theorem* has shown that the candidate preferred by the voter in the middle of the order is the winner (Congleton, 2004).

In case of ties in any of these voting methods, a problem-specific tie-breaking rule may be applied to select an overall winner; this can introduce non-determinism. Properties of voting methods, described in Table 6, indicate which methods are appropriate for a given problem (Arrow et al., 2010). Ideally, a desired voting method should be monotonic, Condorcet consistent, participatory, Pareto efficient, and independent of irrelevant alternatives, without being dictatorial. This would ensure that all votes are valued equally and fairly. Unfortunately, *Arrow's theorem* is correct: given at least three candidates and a voting method that is Pareto efficient and independent of irrelevant alternatives, that method will also be dictatorial (Arrow et al., 2010). Arrow's theorem holds even when there are ties.

This section has reviewed metaheuristics, weight learning, ensemble methods, and voting methods. Both metaheuristics and hybrid metaheuristics combine heuristics

Property	Definition
Monotonic	A candidate's outcome under a specific voting method is not worse when a voter improves its rank for that candidate (Pérez-Fernández et al., 2017)
Condorcet consistent	A voting method always selects the Condorcet winner when such a candidate exists
Participation	The addition of a voter that prefers candidate A over candidate B does not change the winner from A to B
Pareto efficient	When every voter prefers candidate A over another candidate B, the overall result also prefers A over B
Independent of irrelevant alternatives	A voter's preference between two candidates is not influenced by changes in preference for a third, irrelevant candidate
Dictatorial	One voter alone decides the outcome of the vote

Table 6: Properties of voting methods and their definitions

and have been used to seek or improve solutions for path planning. Ensemble methods combine multiple classifiers to produce a more robust overall classification. Weight learning can improve machine learning accuracy. Voting methods decide which class to predict in ensemble methods. An important gap in the literature, however, is a method that incorporates both weight learning and voting to combine multiple heuristics and metaheuristics. This proposal addresses that gap with situated-decision strategies in Chapter 3. Another gap in the metaheuristic literature is a swarm intelligence metaheuristic based on human behavior. The proposed work addresses this with a novel, population-based metaheuristic for path planning. The next chapter proposes situation-based robot navigation based on the related work in cognitive science, HRI, metaheuristics, weight learning, and voting methods discussed in this chapter.

3. Proposed work: situations

This proposal addresses three major challenges in collaborative navigation: how to learn and identify situations, how to learn and apply situated-decision strategies, and how to generate meaningful natural language that explains these processes. This chapter contextualizes this work with a description of an existing robot controller. It then covers situations and situated-decision strategies, issues that arise from them, along with preliminary work and proposed approaches.

3.1 Context: SemaFORR

The development environment for this proposal is SemaFORR, an autonomous robot controller for navigation. The proposed work, however, is applicable to any robot controller and can be implemented alongside other navigation architectures. SemaFORR is an application of FORR, a cognitive architecture for learning and problem solving (Epstein, 1994).

FORR has been successfully applied in multiple domains, including game playing (Epstein, 2001) and constraint satisfaction (Petrovic and Epstein, 2006). The crux of any FORR-based system is that good decisions in complex domains are best made by a mixture of good reasons. FORR represents each good reason with an *Advisor*, a heuristic procedure that scores candidate actions. Similar to a metaheuristic, FORR-based systems use a heuristic strategy to balance multiple heuristic rationales.

Despite sensor noise and actuator error, a SemaFORR-controlled robot learns to navigate effectively in unfamiliar, dynamic, partially observable environments without a map. Like the cognitive models discussed in Section 2.1, SemaFORR also draws upon spatial cognition and human reasoning. To reach its target, SemaFORR uses local sensor data, learned knowledge, and heuristic reasoning to select one action at a time and to contend with any obstacles. The resultant behavior is satisficing, human-like, and rarely optimal. Although SemaFORR uses sensor data from a laser rangefinder, it could be used with other rangefinders (e.g., sonar, lidar).

SemaFORR makes decisions based on a hierarchical reasoning framework and a spatial model that it learns while it navigates. A *decision state* d records the robot's current sensor data and its pose when it makes a decision. As the robot travels, it records its path to the target as a finite sequence of decision states, plus its final decision state when it arrives within distance ε of its target.

3.1.1 The spatial model

SemaFORR learns a compact, approximate spatial model from experience, one that captures many of the features of a cognitive map. Instead of a metric map, SemaFORR's model is a set of *spatial affordances*, abstract representations that preserve salient details and facilitate movement. SemaFORR learns spatial affordances from local sensor readings and stores them as spatial knowledge. This spatial model is based on three primitive spatial affordances: *regions* (obstruction-free areas), *trails* (usually-shorter paths derived from actually travelled paths), and *conveyors* (frequently visited cells in a grid overlaid on the environment). Figure 1 gives examples of each. SemaFORR's spatial affordances are learned only after each successfully completed task. Like a person's cognitive map, SemaFORR's spatial model is a learned abstraction of spatial knowledge. For example, trails capture route knowledge and high-valued conveyors are landmarks.

A *region* is an obstruction-free area where the robot can move freely. A region is represented as a circle whose center is the robot's location in a decision state and whose radius is the minimum distance sensed from that location to any obstacle. An

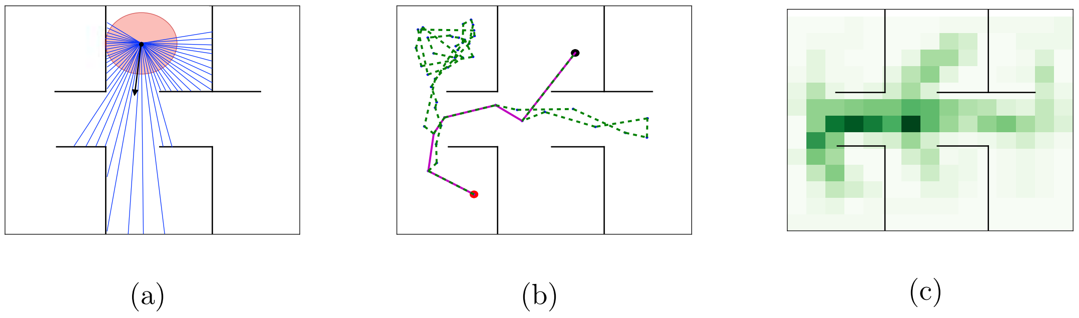


Figure 1: Examples of affordances in a simple environment (a) a region with the robot's pose (black arrow) and laser rangefinders (b) a path (dashed line) and its trail (solid line) (c) conveyors (darker shading denotes higher frequency)

exit is a point on a region's circumference that intersects with a travelled path. Exits are learned as places that allow access to and from a region.

A *trail* refines a path the robot has taken to reach a target. It consists of an ordered list of *trail markers*, decision states selected from the robot's path. The first and last trail markers are the initial and final decision states on the path. The trail-learning algorithm works backward from the end of the path, and creates a new trail marker for the earliest decision state that could have sensed the current trail marker. The resultant trail is usually shorter than the original path and provides a more direct route to the target.

A *conveyor* describes how useful travel has been through a small area. In a grid superimposed on the environment, each cell tallies the frequency with which trails

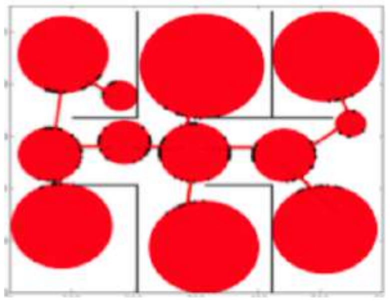


Figure 2: Example of a learned skeleton formed by a set of regions

pass through it. High-count cells in the grid are conveyors. Figure 1(c) denotes higher frequency with darker shading.

SemaFORR's spatial model combines affordances to produce more powerful representations, much the way a person's cognitive map combines landmarks and route knowledge to build survey knowledge. For example, the *skeleton* is a graph that captures global connectivity among regions. Each region is a node in the skeleton and an edge joins two nodes if a path has ever moved between their corresponding regions. Figure 2 shows an example of a learned skeleton.

3.1.2 Decision making

SemaFORR is a hybrid robot controller that uses commonsense qualitative reasoning and its spatial model to make decisions. SemaFORR's Advisors are organized into a three-tier hierarchy, with rule-based decision making in tier 1, path planning in tier 2, and commonsense, qualitative heuristics in tier 3. Table 7 lists SemaFORR's current Advisor rationales by tier. Figure 3 shows how SemaFORR integrates these elements in its architectural design.

Given a decision state and a discrete set of possible actions, a tier-1 or tier-3 Advisor expresses its opinions on possible actions as *comments*. In a *decision cycle*, SemaFORR uses those comments to select an action. Possible action sets are alternately forward moves of various lengths and turns in place that produce various rotations. A move with distance 0 is equivalent to a pause. Thus, in any given decision state, SemaFORR chooses only the intensity level of its next move or turn. The resultant action sequence is expected to move the robot to its target.

Tier 1 invokes its Advisors in a predetermined order; each of them can either mandate an action or veto some subset of actions. If no action is mandated and at least two actions are unvetoed, the remaining actions are forwarded to the next tier.

Tier 1, in order	
VICTORY	Go toward an unobstructed target
ENFORCER	Go toward an unobstructed waypoint
AVOIDWALLS	Do not go within ε_{aw} of an obstacle
NOTOPPOSITE	Do not return to the last orientation
Tier 2	
A*	Minimize distance traveled
CSA*	Avoid crowds
RISK-A*	Avoid risky actions
FLOW-A*	Avoid travel against a crowd's flow
Tier 3	
	Based on commonsense reasoning
BIGSTEP	Take a long step
ELBOWROOM	Get far away from obstacles
EXPLORER	Go to unfamiliar locations
GOAROUND	Turn away from nearby obstacles
GREEDY	Get close to the target
	Based on the spatial model
CONVEY	Go to frequent, distant conveyors
ENTER	Go into the target's region
EXIT	Leave a region without the target
TRAILER	Use a trail segment to approach the target
UNLIKELY	Avoid dead-end regions

Table 7: SemaFORR's Advisors and their rationales.

If a plan does not exist, then tier 2 constructs one and returns the decision cycle to tier 1. SemaFORR currently uses only one planner at a time to produce a plan. Otherwise, the remaining actions are forwarded to tier 3. Each tier-3 Advisor uses its own commonsense rationale to construct its comments on the remaining possible actions.

A comment from a tier-3 Advisor assigns a *strength*, a normalized value in [0,10], to each available action. Strengths near 10 indicate actions that closely conform to the Advisor's rationale; strengths near 0 indicate direct opposition to it. Once all tier-3 Advisors comment, SemaFORR selects the action with the most support using

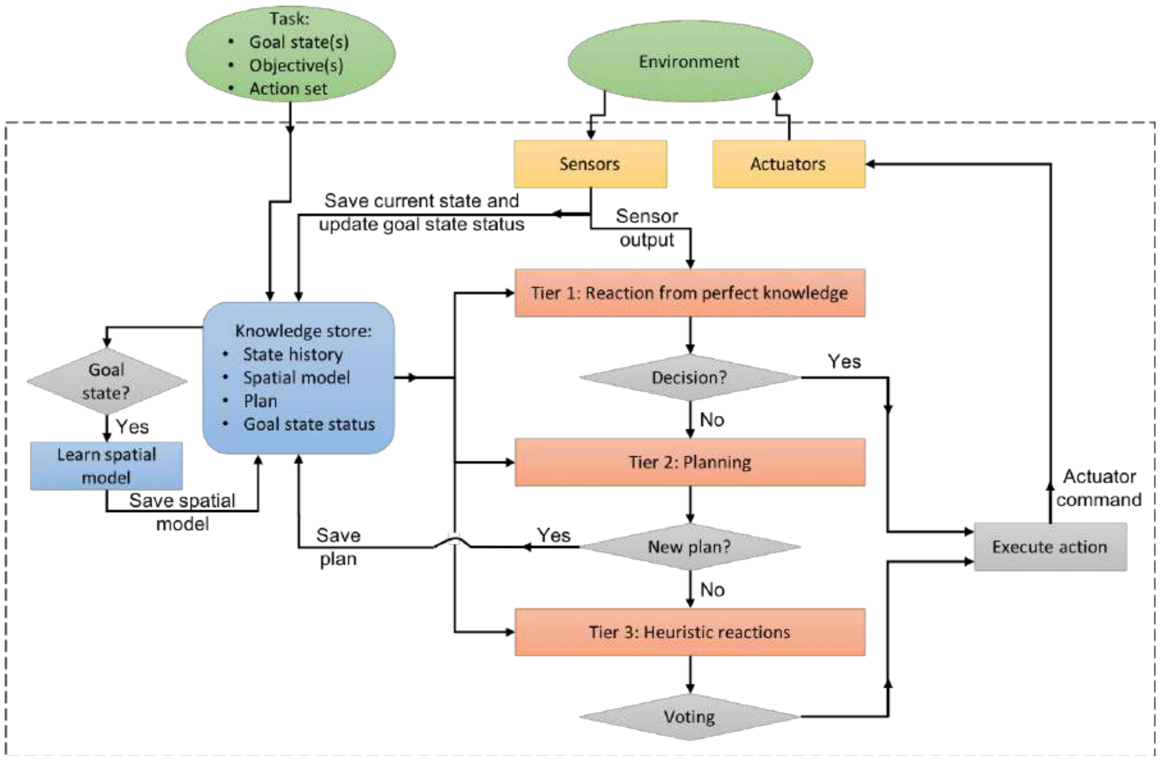


Figure 3: SemaFORR's autonomous robot navigation architecture receives sensor inputs from the environment and acts on that environment with the robot's actuators

sum range voting

$$argmax_{a \in A} \sum_{i=1}^{n} c_{ia}$$

where c_{ia} is the comment strength of Advisor i on action a. Tier 3 introduces non-determinism into action selection because it breaks ties at random.

The remainder of this chapter describes how my proposed work will build upon SemaFORR to produce *Situated-SemaFORR*, a robust system for collaborative navigation. First it discusses approaches to learn and identify meaningful and useful situations that generalize over the robot's experiences. Then it proposes a variety of methods to learn and apply situated-decision strategies. These proposed contributions draw upon cognitive models of human navigation, incorporate principles from human-aware and socially-aware robot navigation, and use weight learning and voting methods to balance rationales. Situations and situated-decision strategies will also facilitate human-friendly explanations of the robot's decisions because of this underlying human-like decision-making process.

3.2 Learn to identify situations

A robot that learns to represent and identify situations can specialize its behavior to address immediate challenges and adapt to human collaborators. This section discusses novel, general (i.e., platform independent) ways to represent situations and methods to evaluate and access learned situations. Situated-SemaFORR will use these representations to facilitate situated-decision strategies and human-robot communication.

While a state s in robot navigation typically represents the robot's pose and a decision state d incorporates the robot's sensor data, a *situated state* x also includes additional features, such as a robot's characteristics, its accessible external information, its learned internal knowledge, its tasks and objectives, and its knowledge about its environment. More formally, let X be a set of situated states that represent instances of the environment. Then, a *situation* $U \subseteq X$ is a subset of similar situated states that can be represented by a single *exemplar* e. Similarity is measured by a problem-specific metric on a class of features that incorporates the robot's situated nature. Situations are not mutually exclusive, nor need a learned set of situations be collectively exhaustive. Table 8 provides examples of features that could define a situation.

There are many potentially meaningful situations. Some reflect the robot's knowledge about its environment: the robot may be *lost* (current state is unknown but environment is known), in an *unknown area* (current state and environment are unknown); otherwise it is in a *familiar* area. Some situations reference other objects in

Features	Potential variables
Individual differences in spatial abilities and strategies	• Amount of experience in the environment on a discrete scale • Available actions in current state on a discrete scale • Available strategies in current state • Computational restrictions (time limit or memory limit) on a discrete scale
External information available to the robot (e.g., instructions, maps, directions)	• Reliability on a discrete scale • Comprehensiveness on a discrete scale • Ease of access on a discrete scale • Understandability on a discrete scale • Value of information on a discrete scale • Agreement with overall environment, current state, or spatial affordances on a discrete scale
Internal information available to the robot (e.g., spatial affordances)	• Reliability on a discrete scale • Comprehensiveness on a discrete scale • Value of information on a discrete scale • Probabilistic cognitive map (likelihood of affordance's accuracy and utility)
Overall objectives	• Minimize travel time, computation time, or travel distance • Maximize comfort of people in the environment
Current task	• Avoid obstacles • Get to the target
Overall environment	• Static or dynamic • Presence of people and other robots on a discrete scale • Fully observable or partially observable
Current state	• Sensor readings (e.g., tightly constrained on all sides and shorter range readings to the robot's left) • Internal model (e.g., inside a region and near a trail) • Number of times previously visited on a discrete scale • Nature of immediate vicinity (spatial arrangement of objects with respect to the robot)

Table 8: Features for a situated state and examples of possible criteria to define a situation

the environment: the robot detects a crowd of moving people or is accompanied by a person. Other situations capture the robot's progress on its task: the robot is *late* (travel time has exceeded some threshold), trapped in some part of the environment (time in a spatial area of the environment has exceeded some threshold), or is moving erratically (path smoothness criterion has been violated).

The problem now becomes how to recognize useful situations online. This requires:

- An efficient and compact data structure to capture all relevant information.
- A method to learn situations from these data structures and to produce an exemplar for a situation.
- An evaluation mechanism to ensure that situations are both meaningful and useful.
- An efficient method to access and detect learned situations.

Situated-SemaFORR addresses these challenges with a variety of approaches to represent, learn, evaluate, and access situations.

3.2.1 Situation representation

Four ways to represent situations will be explored; Figure 4 shows the relationships between them. First, a situated state can be represented as a feature vector $\langle f_1, f_2, \ldots, f_m \rangle$. Individual features will describe the robot's pose, sensor readings, learned knowledge, current task, objective criteria, external or internal information, possible actions, and metaknowledge. Features will change values with different frequencies; some will remain unchanged for long periods of time. Methods for such a data structure will address this issue. They will also quickly encounter the curse of dimensionality.

Dimensionality reduction can reduce the size complexity of a feature vector and identify significant features of situations. One approach is feature selection with metaheuristics, which identifies those features that are most useful with respect to the current problem (Guyon and Elisseeff, 2003). Another approach is feature extraction with principal component analysis, which maps a set of features to a lower dimensional space with orthogonal linear combinations that capture correlations in the data (Abdi and Williams, 2010). Either the full feature vector or its projection onto a lower dimensional space may be used by the proposed situation learning algorithm to produce an exemplar e.

Second, a situation can be represented as a geometric pattern with, for example, points, lines, or polygons. This representation provides an opportunity to exploit geometric properties and spatial relations among the features in a robot's situated state, such as sensor readings, pose, target, and spatial knowledge. Like the way people represent and describe spatial structures, a geometric representation of spatial patterns allows learned situations to be flexible and robust to sensor noise. One such representation could use the distribution and shape of a robot's range sensors to identify distinct spatial structures in the environment. For example, a doorway is a geometric situation that could be represented as the set of situated states in which a robot's range sensors identify closer obstacles on its sides than in front of or behind

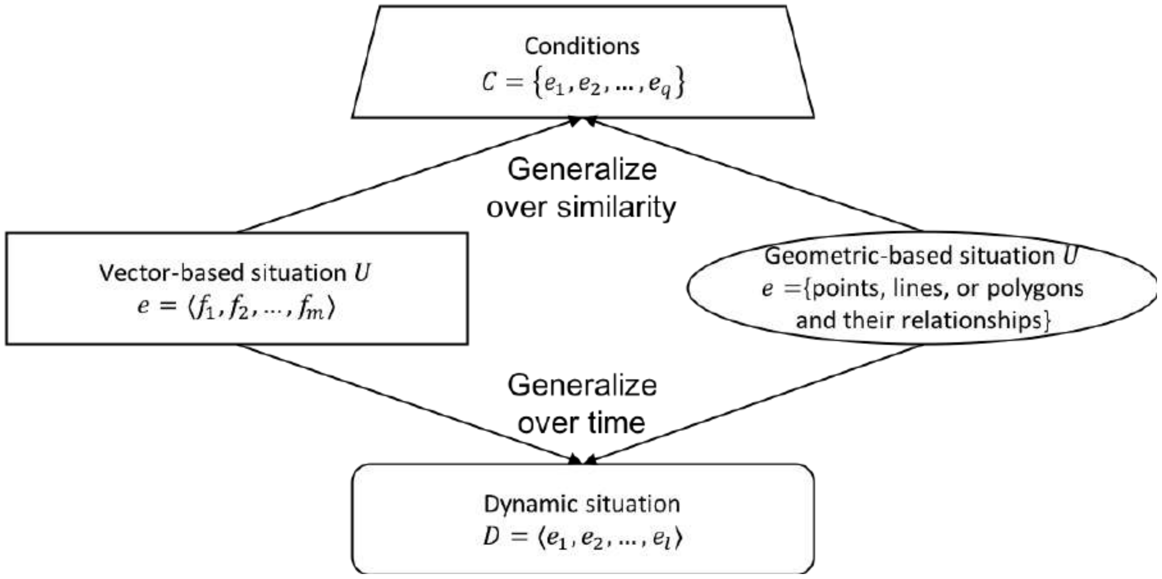

Conditions
$C = \{e_1, e_2, \ldots, e_q\}$
Generalize over similarity
Vector-based situation U
$e = \langle f_1, f_2, \ldots, f_m \rangle$
Geometric-based situation U
e ={points, lines, or polygons and their relationships}
Generalize over time
Dynamic situation
$D = \langle e_1, e_2, \ldots, e_l \rangle$

Figure 4: Hierarchy of situation representations that shows how situations can be generalized by their similarity as conditions or over time as dynamic situations

it. Another potential situation is a traditional corner; where two walls meet, the range sensor's endpoints should trace two lines that make an approximate right angle. This geometric representation allows a situation to be flexible and parameterizable. For example, the doorway situation allows for variety in the width and the depth of the door frame. Similarly, the corner situation allows for some variation in the angle between the obstacles. Ideally, a robot would learn to identify those geometric patterns that are most useful for its task.

Both the vector-based and geometric-based representations for situations symbolize sets of similar situated states. Situations can be further generalized into higher-level representations, either by similarity or over time, as shown in Figure 4. A *dynamic situation* represents dynamic events and patterns across time as a sequence of exemplars in chronological order, $D = \langle e_1, e_2, \ldots, e_l \rangle$. For example, an opening door can be represented as a sequence of situations: door closed, door partially open, and door completely open. A dynamic situation also occurs when an elevator door opens or a person passes by the robot. Exemplars can also be grouped into *conditions* based on their similarities, $C = \{e_1, e_2, \ldots, e_q\}$. For example, the situations where the robot enters a room, exits a room, and travels through a narrow corridor could be grouped together because they share similar characteristics (obstacles close to the robot's sides).

3.2.2 Learning situations

Given representations for situations, the next step is to design an online algorithm that learns situations efficiently from limited experience. Algorithm 1 is pseudocode for a basic, vector-based, situation-learning algorithm. Its input is a set of observed situated states $O = \{x_1, x_2, \ldots, x_g\}$, where $O \subseteq X$. The algorithm identifies the

Algorithm 1: Vector-based situation learning algorithm

```
Input: situated states $x_1, x_2, \ldots, x_g$
Output: exemplars for situations $e_1, e_2, \ldots, e_b$
$Observations \leftarrow x_1, x_2, \ldots, x_g$
$Exemplars \leftarrow \{\}$
$\{U_1, U_2, \ldots, U_b\} \leftarrow group(Observations)$
for $situations\ U_h, h = 1 \ldots b$ do
    $e_h \leftarrow exemplify(U_h)$
    $Exemplars \leftarrow Exemplars \cup \{e_h\}$
end
return $Exemplars$
```

situations, sets of similar states $\{U_1, U_2, \ldots, U_b\} \subseteq O$, and, for each situation U_h, it outputs an exemplar e_h.

Algorithm 1 groups observations into situations and then generates an exemplar for each situation. These two components can be decoupled, with separate approaches for each. A clustering algorithm will group observed states into vector-based situations based on similarity. Several popular approaches will be considered: hierarchical clustering, k-means clustering, expectation maximization clustering, density-based clustering, fuzzy clustering, and subspace clustering (Berkhin et al., 2006). Each clustering algorithm groups data together in different ways that allow for flexibility. For example, rather than a strict partition of the data, fuzzy clustering assigns a likelihood membership for each cluster, which allows for more nuanced, overlapping situations. Other approaches create a strict partition, which forces each situation to be more distinctive. Density-based clustering approaches exclude outliers from clusters, which could allow those outliers to symbolize potential new situations. Situated-SemaFORR will use online adaptations of these clustering algorithms to continuously cluster data as it is collected, so that the robot can learn as it travels (Aggarwal and Reddy, 2013).

The second part of Algorithm 1 identifies an exemplar e_h for a situation $U_h \subseteq O$. One approach would take the centroid of a cluster as its exemplar (e.g., the mean vector or the median vector computed on the values for each feature). Another approach would select one of the observed states as the exemplar, possibly the one closest to the centroid or the *medoid* (a vector that minimizes the Manhattan distance to all the other vectors in the cluster). Another approach could be sparse coding, which finds those representative data points in the set that can be linearly combined to describe all the other points in the dataset (Elhamifar et al., 2012). Sparse coding is different from a linear mapping with eigenvectors because it linearly combines data points, whereas eigenvectors finds linear combinations of features in the data.

Learning geometric-based situations requires a more nuanced approach. Algorithm 2 is pseudocode for a proposed geometric-situation learning algorithm. Its input is a set of potential spatial relations $\{r_1, r_2, \ldots, r_z\}$ among the features $\langle f_1, f_2, \ldots, f_m \rangle$ in the observed situated states, where each spatial relation applies to a list of features. Since there are exponentially many subsets of features and infinitely many possible spatial relations, exhaustive computation is infeasible. Instead, the input to Algorithm 2 is a predefined list of spatial relations of interest. Examples include the lines formed by pairs of the $\langle x, y \rangle$ features (e.g., the robot's position, the target, previous robot positions), the angles between those points, and the polygons formed by three or more of those points. Other spatial relations could consider the robot's pose with respect to its spatial model and the environment (e.g., lines formed by the robot's position and the nearest region's center). Such a representation could potentially learn situations based on the robot's proximity to a region, and a situated-decision strategy could use this information to adapt the robot's behavior.

A geometric-based situation learning algorithm will first convert each of the observed situated states into a set of geometric objects. Next, the algorithm will group

Algorithm 2: Geometric-based situation learning algorithm

```
Input: spatial relations $r_1, r_2, \ldots, r_z$ and situated states $x_1, x_2, \ldots, x_g$
Output: exemplars for situations $e_1, e_2, \ldots, e_b$
$Relations \leftarrow r_1, r_2, \ldots, r_z$
$Observations \leftarrow x_1, x_2, \ldots, x_g$
$Exemplars \leftarrow \{\}$
$GeometricObjects \leftarrow convert(Relations, Observations)$
$\{U_1, U_2, \ldots, U_b\} \leftarrow group(GeometricObjects)$
for $situations\ U_h, h = 1 \ldots b$ do
    $e_h \leftarrow exemplify(U_h)$
    $Exemplars \leftarrow Exemplars \cup \{e_h\}$
end
return $Exemplars$
```

these geometric objects into situations based on their similarity with a clustering algorithm (e.g., the same potential approaches proposed for the vector situations). Finally, the algorithm will produce an exemplar to represent each situation with methods proposed earlier.

For example, consider a situated state that includes the robot's closest and farthest sensor endpoints, $\langle x_{close}, y_{close} \rangle$ and $\langle x_{far}, y_{far} \rangle$ respectively. As in Figure 5, a geometric relation r of interest is the line determined by $\langle x_{close}, y_{close} \rangle$ and $\langle x_{far}, y_{far} \rangle$. Algorithm 2 would calculate the slope and intercept of that line in each situated state x, and then cluster those pairs to group situated states with similar slopes and intercepts together. Finally, it would produce an exemplar for each cluster.

Another approach to learn and represent situations is deep learning. Recent success in image recognition, for example, has relied on deep learning to identify features and reduce the dimensionality of the data. More recently, unsupervised and semi-supervised deep learning architectures have learned appropriate similarity metrics and clustered high-dimensional data (Law et al., 2017; Chen, 2015). A *capsule network* is a recent deep learning approach that encodes spatial relations among different parts

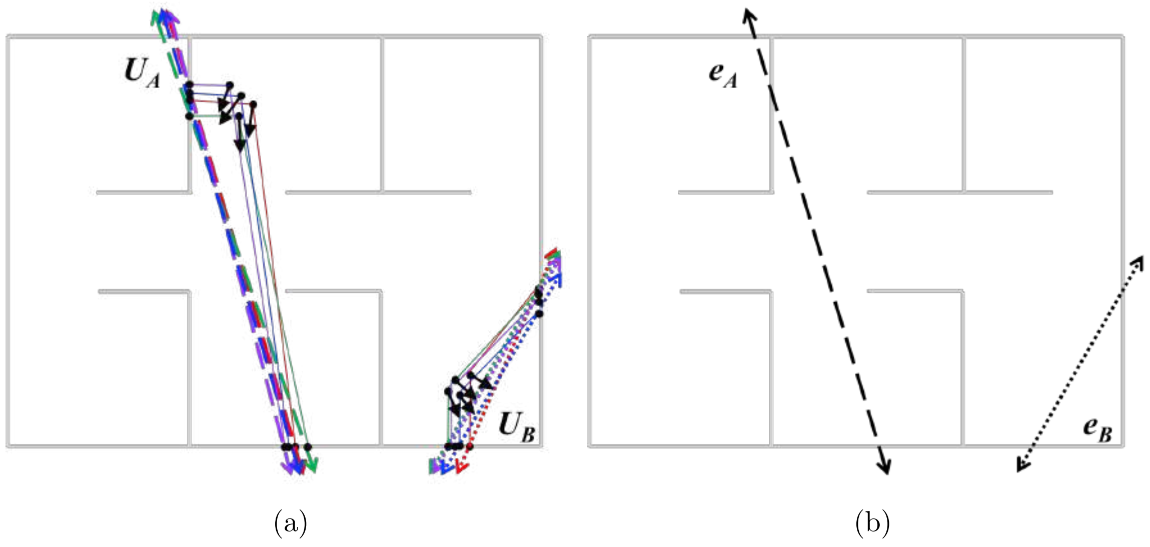


Figure 5: (a) Examples of potential geometric situations U_A (dashed lines) and U_B (dotted lines) determined by $\langle x_{close}, y_{close} \rangle$ and $\langle x_{far}, y_{far} \rangle$. Algorithm 2 groups the lines based on their slope and intercept. The robot's pose at a decision point appears as a black point and arrow. Other than the gray walls invisible to the robot, solid colored lines indicate the closest and farthest obstacles detected by the robot's laser rangefinder from each of the robot's poses. (b) The exemplars e_A and e_B for U_A and U_B that could be produced by Algorithm 2

of an image in a convolutional neural network (Sabour et al., 2017). To learn geometric situations, a capsule networks could identify spatial relations among features in a situated state and then create situations with those learned relations.

Given enough learned situations, clusters of their exemplars will be *conditions*, which group the most similar situations into broader categories. To create meaningful conditions, the learned situations would ideally be varied and numerous. There is no way to know a priori, however, whether situation learning will produce enough situations to identify meaningful conditions. The selection of a clustering algorithm and the settings for its parameters will be varied to attempt to produce a satisfactory set of situations from which to learn conditions. The resultant hierarchy of abstractions based on the conditions and the situations could facilitate hierarchical decision-making with the situated-decision strategies, as well as a more natural description of the robot's learned knowledge for a human collaborator.

Dynamic situations are more difficult to learn because they involve patterns over time. Algorithm 3 is pseudocode to learn dynamic situations. The algorithm accepts a chronological sequence of observed situated states, identifies which situation each belongs to, and builds a corresponding sequential list of their respective exemplars in the same order. If a situated state belongs to more than one situation, then multiple potential sequences must be considered. For example, if a sequence of observed situated states x_1, x_2, x_3, x_4, x_5 were associated with situations U_1, U_3, U_1, U_3, U_2, respectively, Algorithm 3 would produce the sequential list of exemplars e_1, e_3, e_1, e_3, e_2. Next, the algorithm characterizes the exemplar sequence by its most frequent subsequence of length greater than one. In the example, this would be $e_1\ e_3$, a learned dynamic situation. Likely approaches to find the most common sequences in a list of objects are drawn from natural language processing, such as n-gram models (for exact sequences of length n) and q-gram models (for approximate sequences of length q) (Fink, 2008). Another approach could be text compression methods that find sequences of frequent symbols to achieve maximum compression of text data (Wan, 2003).

3.2.3 Situation evaluation and access

Learned situations will be evaluated by two metrics that examine how well an algorithm identifies distinct clusters (Brun et al., 2007). One metric is the sum of the squared distances between each data point and its cluster's centroid. Ideally, this metric should be close to 0, which would indicate high intra-cluster similarity. The other evaluation metric is the Silhouette Coefficient, which compares the mean distance between a data point and all other points in the same cluster with the mean

```
Algorithm 3: Dynamic situation learning algorithm
Input: situated states $x_1, x_2, \ldots, x_g$, situations $U_1, U_2, \ldots, U_b$, exemplars
  $e_1, e_2, \ldots, e_b$, and $\eta$ the minimum length of dynamic situation sequences
Output: $DynamicSituations$
$Observations \leftarrow x_1, x_2, \ldots, x_g$
$Situations \leftarrow U_1, U_2, \ldots, U_b$
$Exemplars \leftarrow e_1, e_2, \ldots, e_b$
$Sequence \leftarrow (\ )$
for $x_\iota, \iota = 1 \ldots g$ do
    identify $x_\iota$'s situation in $Situations$
    append that situation's exemplar to $Sequence$
end
$DynamicSituations \leftarrow$ most common patterns with length $\geq \eta$ in $Sequence$
return $DynamicSituations$
```

distance between that same data point and the nearest-neighbor cluster (Rousseeuw, 1987). This approach addresses not only intra-cluster similarity, but also inter-cluster dissimilarity.

Other cluster-evaluation methods use a gold standard or hand-labeled clusters to evaluate a learned clustering. To that end, several human coders could cluster the observed situated states from a robot's navigation experience to generate a standard. Methods to compare a clustering to a gold standard include precision and recall, mutual information, the Rand index, the Fowlkes-Mallows index (Wagner and Wagner, 2007), and the V-measure (Rosenberg and Hirschberg, 2007).

Situated-SemaFORR must contend with two challenges: where to store its learned knowledge, and how to retrieve data from that knowledge store. Data may be too extensive for storage on-board the robot, but retrieval from off-board storage would be limited by the wireless bandwidth. If a robot limited how often it learned situations, that could detract from navigation decisions and prevent human-like, real-time learning. Although no easy solution exists for the data storage problem, one approach could be to retain only some navigation history on board (e.g., only those data points close to exemplars). Although this could result in the loss of potentially important data, it is more human-like; people tend to remember only the most salient details of their experiences. Wherever the knowledge is stored, Situated-SemaFORR will efficiently retrieve it with standard database retrieval methods.

Another issue related to retrieval is how to determine if the robot's current state is part of an existing situation or if it is potentially novel. The robot must quickly determine its situation so that it can select the appropriate situated-decision strategy. One approach is to measure the distance between the current state and the exemplars for each of the situations. If the distance is below some threshold then it will be considered part of that exemplar's situation, otherwise it would be novel. (A threshold less than half the distance between the two closest exemplars, among them all, guarantees that the current state would never be below the threshold for more than one situation.) Another approach would be to assume that the current state is part of the closest exemplar's situation, and then periodically recluster (e.g., at the end of a task).

3.2.4 Preliminary work

Situated-SemaFORR adapts SemaFORR's architecture to incorporate situation learning, which will be done in parallel with learning the spatial model. Figure 6 shows Situated-SemaFORR's architecture, a revision of Figure 3 that incorporates situation learning. After situations are learned, tier 1 now also receives the current situation based on the robot's current state. Section 3.3 will describe how these learned situations will be used within the context of this revised architecture to improve both navigation behavior and the ability to collaborate effectively with people.

SemaFORR is a package implemented in *ROS*, the state-of-the-art Robot Operating System. This allows SemaFORR to control either a physical robot or a simulated

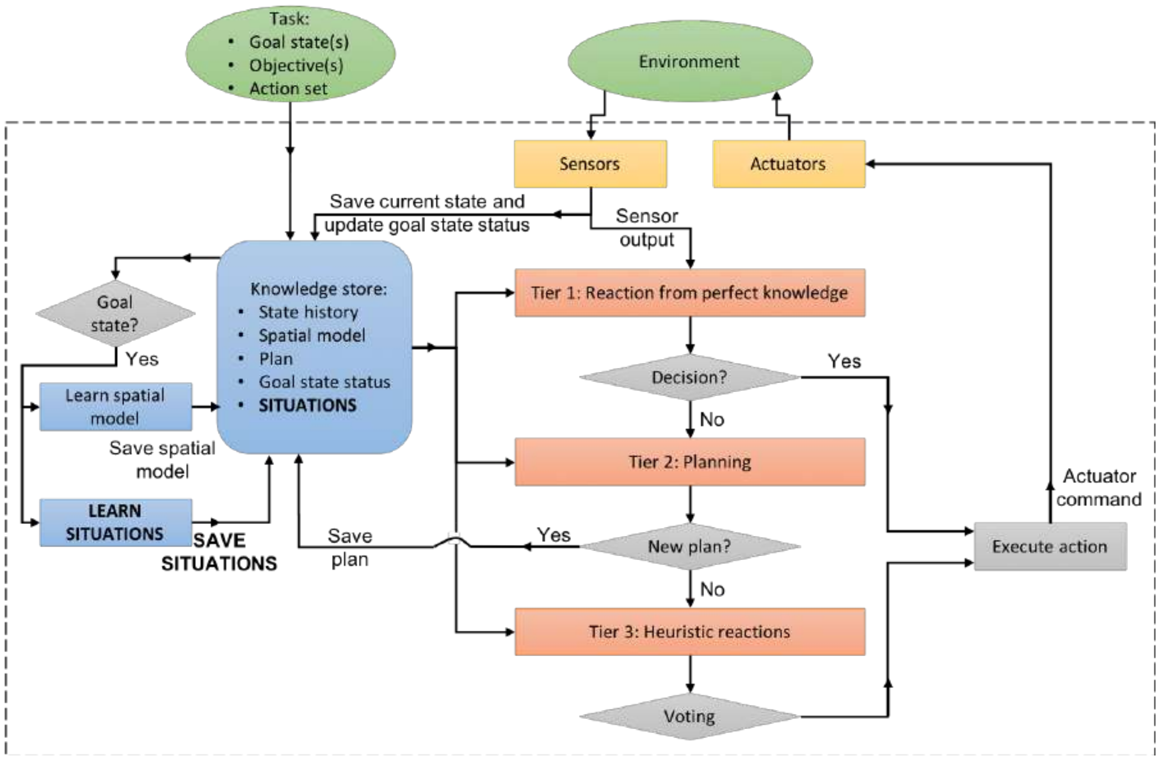


Figure 6: Situated-SemaFORR's architecture with situation learning. Proposed work introduced in this section is indicated in uppercase.

one. My preliminary work included the implementation of trails and several Advisors, data logging in a machine readable format, and bug fixes in the ROS version of SemaFORR. The modular nature of ROS also supports easy integration of new components into existing systems, such as those for Situated-SemaFORR. *MengeROS* is a new tool that simulates both robot navigators and crowds of pedestrians (Aroor et al., 2017). MengeROS is integrated with ROS and it should facilitate rapid evaluation of different dynamic navigation scenarios. Although MengeROS can simulate multiple robots, this proposal specifically focuses on the problem of collaborative navigation for a single robot.

In preparation for situation learning, I have implemented a general framework to capture a situated state's features. It processes the output from the robot's log data and produces a data object that stores values for all of the specified features. The framework is modular; it is easy to add and remove features and to build more sophisticated features that combine existing features. Its output is the vector-based situated states that will be used for situation learning.

This section has formalized situated states, proposed approaches to represent situations, and described methods to identify, evaluate, and access learned situations. These methods draw upon techniques from machine learning (e.g., clustering and deep learning), natural language processing, and cognitive science. The next section considers how to learn and use situated-decision strategies for each learned situation. These strategies will enable Situated-SemaFORR to navigate effectively and collaboratively with people and to explain its decision-making rationales more naturally.

3.3 Learn situated-decision strategies

Currently, SemaFORR and most autonomous robot navigation systems assume that one general strategy is appropriate for all situations. Situated-SemaFORR's hypothesis is that specialization of a robot's behavior for each situation is more human-like, more efficient, and will result in improved human-robot collaborative navigation. Here, situated-decision strategies will combine heuristics, metaheuristics, path planning, and weight learning to find satisfactory solutions to situations. This section describes multiple approaches to learn and use situated-decision strategies. It also discusses how these strategies will be evaluated and how they will evolve over time.

Formally, a *situated-decision strategy* χ selects either an action a for a given situation U, or a plan P for a given dynamic situation D, to advance the robot toward its target. For example, in the corner situation described earlier, a situated-decision strategy could choose to rotate the robot 180° so that it no longer faced into the corner. This approach draws upon *case-based reasoning*, a model of human reasoning that solves new problems by finding and adapting solutions to similar problems previously encountered (Kolodner, 2014). Most case-based reasoning systems rely on a diverse and reliable database of experiences with solutions that must be specified prior to use. In contrast, Situated-SemaFORR will allow the robot to learn both situations and strategies for responses to those situations.

The remainder of this section discusses the components of situated-decision strategies. A variety of new Advisors allow the robot to integrate robust, diverse strategies into its decision-making mechanism. Situated-SemaFORR will also add methods to learn situated-decision strategies with weight learning and voting.

3.3.1 New spatial representations

One way to potentially improve SemaFORR's decision making capability is to add more sophisticated representations to its spatial model. This subsection describes two new spatial affordances: doors and hallways. It also describes barriers, a spatial representation of obstacles in the environment.

A *door* generalizes over the exits of a region. It is represented as an arc along the region's circumference. The door-learning algorithm introduces a door when the length of the arc between two exits is within some small ε_{door}. Once generated, a door incorporates additional exits if they are within ε_{door} of it. A door represents a way into or out of a region. Doors and the door-learning algorithm were developed by Gil Dekel (a Hunter undergraduate student). My preliminary work for this proposal included their incorporation into SemaFORR.

Another potential spatial affordance is a *hallway*, which captures well-travelled routes in some *cardinal direction* (vertical, horizontal, major diagonal, or minor diagonal), and typically connects different parts of the environment. A hallway generalizes over line segments observed by the robot, such as neighboring trail markers on a trail, neighboring situated states in its path, or range sensor end-points from the robot's position. Like other spatial affordances, hallways are learned after each task is successfully completed.

Algorithm 4 is pseudocode to learn hallways. Its input is a set of line segments, such as the learned trails or the robot's path. First, the line segments are grouped by the four cardinal directions. (More directions would capture more nuances in the environment's connectivity, but would also likely learn fewer hallways because the data would be partitioned into more groups.) Next, within each group of line segments, nearby line segments are merged together based on the distance between their respective endpoints and the similarity in their angle with respect to the horizontal axis. This step consolidates information and adjusts for sensor error. Finally, the merged segments are pruned based on their distance δ from other merged segments and the number ω of nearby segments. The parameters δ and ω are tunable thresholds that control how many hallways are produced. For example, a conservative approach would require many segments to be close together to justify the existence of a hallway. Finally, to create hallways, one approach is to obtain the convex hull for each group of nearby segments. Another approach represents a hallway as a rectangle. This approach would maximize coverage of nearby segments inside a rectangle and also minimize the amount of empty space in the rectangle.

Algorithm 4: Hallway learning algorithm

```
Input: trails or paths, $\delta$ and $\omega$
Output: $Hallways$
$LineSegments \leftarrow$ input line segments
$CardinalDirections \leftarrow Partition(LineSegments)$
$MergedSegments \leftarrow Merge(CardinalDirections)$
$Hallways \leftarrow Prune(MergedSegments, \delta, \omega)$
return $Hallways$
```

Algorithm 5: Barrier learning algorithm

```
Input: decision state history
Output: $Barriers$
$SensorHistory$ ← decision state history
$Barriers$ ← { }
for each situated state ∈ $SensorHistory$ do
    Construct line segments from each pair of adjacent sensor endpoints
    Repeatedly merge adjacent line segments with similar slopes
    if a line segment is near an existing barrier and has a similar slope then
        Merge the line segment with the barrier
    $Barriers$ ← remaining line segments
end
return $Barriers$
```

Another new spatial representation will be a *barrier*; it generalizes over range readings to symbolize long stretches of obstacles as lines. Without a map, SemaFORR relies purely on its sensors to detect obstacles in the environment. This spatial representation abstracts the sensed endpoints from a rangefinder to produce lines. For example, our real-world robot (Fetch Robotics' Freight) detects obstacles within 25 meters from the robot along a 220° arc. Learned barriers approximate the locations of obstacles in the environment. Barriers are not meant to capture all spatial details of the obstacles in the environment or to build a map, but rather to abstract observed sensor readings into useful representations that facilitate the robot's movement.

Algorithm 5 is pseudocode to learn barriers. First, it finds the slope of the line segment formed by each pair of adjacent sensor endpoints. Although SemaFORR accepts any number of range sensor readings in a parameterized arc, it currently simulates Freight's 660 range sensors, which would yield 659 line segments with their slopes. Adjacent line segments are merged when the difference in their slopes is below some ε_{slope}. The merged line segment is found by least-squares regression on the points that make up the merging segments. This merger process is repeated until no adjacent line segments meet the merger criterion. Next, each remaining line segment is compared to the previously learned barriers and is merged if the slope is similar and the distance between the line segments is below some $\varepsilon_{barrier}$. The final list of barriers is saved and the process is repeated with a new decision state.

In the worst case, Algorithm 5 is quadratic in the number of sensor endpoints because for ν endpoints, there will be a maximum of $(\nu-2)(\nu-1)/2$ comparisons between adjacent line segments. Additionally, each line segment is compared against the ξ previously learned barriers, which in the worst case is $\xi = (\nu-1) * |SensorHistory|$. So the overall worst-case complexity of the algorithm is $O(|SensorHistory| * \nu^2)$. To reduce this complexity, the algorithm could skip some sensor endpoints or do batch learning.

3.3.2 New tier-3 Advisors

This subsection describes potential tier-3 Advisors based on a variety of rationales. These Advisors draw from commonsense reasoning, new spatial affordances, models of pedestrian movement, proxemics, cognitive science, and a human-subject study. Table 9 summarizes them, along with their basis and rationale.

Several new tier-3 Advisors will use the proposed spatial affordances to comment on actions. As part of my preliminary work, I created three door-based Advisors: ACCESS, ENTERDOOR, and EXITDOOR. ACCESS supports actions toward regions with many doors because they potentially represent places in the environment with high connectivity. ENTERDOOR supports actions that go into a region through a door; EXITDOOR supports actions that leave a region through a door.

New tier-3 Advisors will utilize the learned hallways. For example, FOLLOW will support actions that take the robot farthest down the hallway nearest to the target, in the direction that brings it closest to the target. Hallways can also be connected in

Advisor	Rationale
	Based on commonsense reasoning
CURIOSITY	Go to never visited locations
LEARNSPATIALMODEL	Go to locations without learned spatial affordances
	Based on the spatial model
ACCESS	Go to a region with many doors
ENTERDOOR	Go into the target's region through a door
EXITDOOR	Leave a region without the target through a door
FOLLOW	Use a hallway to approach the target
CROSSROADS	Go to a highly connected hallway
STAY	Stay within a hallway
KEEPBACK	Stay far away from barriers
PARALLEL	Move along a barrier
GETAROUND	Go around a barrier toward the target
	Based on the crowd models
CROWDAVOID	Go to low-density grid cells
FINDTHECROWD	Go to locations with uncertain crowd density
RISKAVOID	Go to low-risk grid cells
FINDTHERISK	Go to locations with uncertain risk
FLOWAVOID	Go along with the crowd flow
FINDTHEFLOW	Go to locations with uncertain crowd flow
	Based on proxemics
INTERPERSONAL	Do not violate personal or intimate space
FORMATION	Do not violate F-formations
FRONT	Do not directly confront a person
REAR	Maintain distance behind a person
SIDE	Do not cross perpendicularly in front a person
VISIBLE	Stay in sight of nearby people
WAIT	Wait for person to pass
	Based on cognitive science
ENFILADE	Move toward a recent position
THIGMOTAXIS	Stay near a large object or barrier
VISUALSCAN	Turn in place to examine the environment
LEASTANGLE	Leave a region or a hallway in the target's direction
	Based on a human-subject study
HUMANPREDICTOR	Support actions predicted by human trajectories
HUMANREWARD	Support actions based on a learned reward function
HUMANDESCRIPTION	Support actions based on input strategies

Table 9: Proposed tier-3 Advisors and their rationales

a skeleton-like graph. CROSSROADS will support actions that take the robot closer to hallways with more connectivity in that graph. STAY will support actions that keep the robot within a hallway, because the hallway is a well-traversed area.

Other new tier-3 Advisors will reference barriers to comment on potential actions. KEEPBACK will support actions that avoid learned barriers. PARALLEL will support actions that move along learned barriers. GETAROUND will support actions that circumnavigate a barrier if the target is on the other side. (GETAROUND is similar to an Advisor implemented in *Ariadne*, a FORR-based system for navigation in simplistic grid environments, that circumnavigated obstructions (Epstein, 1998).)

In addition to these new spatial model-based Advisors, two new Advisors will be created to encourage Situated-SemaFORR to explore and actively learn about its environment. The rationale of LEARNSPATIALMODEL is that a spatial model that covers more of the environment will better facilitate travel. It will support actions toward parts of the environment without known regions and with low conveyor counts. CURIOSITY will support actions that take the robot toward hitherto unvisited parts of the environment. CURIOSITY considers the robot's previously visited locations from its entire history, whereas EXPLORER only considers those locations visited during the robot's current task. These two exploration-based Advisors will contend with Advisors that greedily exploit the robot's knowledge, including GREEDY and those Advisors based on the spatial model.

Recent work on SemaFORR incorporated a path planning algorithm, *CSA**, that learns to avoid crowded areas of the environment (Aroor and Epstein, 2017). The *crowd model* is represented as a grid overlaid on the environment, where each cell records the crowd density observed by the robot. Once learned, this crowd model can also be exploited by new tier-3 Advisors to vote on actions that support crowd avoidance. Just as CONVEY attracts the robot toward high-valued conveyor-grid cells, CrowdAvoid will attract it toward low-density crowd-grid cells. FindTheCrowd's rationale is to reduce uncertainty in the crowd model's density estimate, so it supports actions that go toward grid cells where the robot has recorded crowd observations less often or where the observations have high variance.

I also contributed to recent work that learns a *risk model*, whose grid cells record how often the robot came close to a pedestrian (Aroor et al., 2018). Other recent work developed a *flow model*, which records, for each grid cell, how often obstacles move in eight directions. Similar Advisors will be based on these two models: RiskAvoid, FlowAvoid, FindTheRisk, and FindTheFlow. For example, FlowAvoid will support actions that move along with the crowd and avoid those opposite to the crowd's direction. Although these new Advisors support crowd avoidance, they could be easily modified to seek the crowd if the robot's task calls for it (e.g., a museum robot guide that seeks out people).

Additional new tier-3 Advisors will draw upon proxemics to incorporate human social norms into Situated-SemaFORR's decision-making rationales. Because this proposal addresses collaborative navigation, robot-robot interaction and social norms between robots are not considered. Interpersonal is inspired by Hall's four interpersonal distances (intimate, personal, social, and public) to penalize actions that come too close to a detected person (Hall, 1963). It will support actions that remain in the social and public space and oppose actions that enter or stay in a person's intimate or personal space. It has been suggested that a logarithmic penalty function is more appropriate than a linear function (Henkel et al., 2014).

Another model from proxemics is the F-formation, which describes how two or more people arrange themselves spatially during an interaction (Kendon, 1990). These formations include lines, circles, and other patterns that depend upon the specific joint activity of a group of people. A person *faces* someone or something when her orientation θ from her pose positions her face toward it. FORMATION will oppose actions that would move the robot through a detected F-formation (e.g., cross between two people who face each other). This requires a pose detection algorithm to identify the poses of people nearby based on the robot's percepts. The arrangement of those poses would be matched against a database of known F-formations. (Although most human pose estimates rely on cameras or 3D laser range scans, we could adopt an existing 2D laser rangefinder approach with a Kalman filter (Svenstrup et al., 2009).)

Several other Advisors incorporate social norms from HRI and proxemics based on peoples' detected poses. When a person faces the robot, FRONT will oppose actions that directly approach her. When the robot is behind someone who faces away from it, REAR will support actions that maintain some distance from that person. SIDE will oppose actions that cross perpendicularly in front of a person. The robot is *visible* to a person when it is in that person's field of view relative to her orientation. VISIBLE will support actions that offer visibility to the most nearby people. WAIT will support actions that allow nearby people to pass by the robot first.

Other potential Advisors incorporate human strategies documented in the cognitive science literature. One study found that subjects followed three heuristic search strategies: enfilading, thigmotaxis, and visual scan (Kallai et al., 2005). *Enfilading* is movement that oscillates in a small area of the environment. *Thigmotaxis* is movement that stays near a large object or on the periphery of an open area during navigation as a safety mechanism in an unfamiliar environment. *Visual scan* is movement that remains in a fixed position and turns in place to examine the environment. New Advisors with these strategies as rationales will encourage the robot to examine unknown environments. ENFILADE will support actions toward the robot's recent positions. THIGMOTAXIS will support actions that keep the robot close to a nearby barrier or obstacle. VISUALSCAN will support actions that encourage the robot to rotate in place toward orientations the robot has not faced when in that part of the environment but minimize overlap with the robot's current field-of-view. For example, since our robot's laser rangefinder detects along a 220° arc, VISUALSCAN will support rotation to an orientation where the new arc least overlaps with the current arc.

The *least-angle strategy* is a decision-making heuristic in which a person selects the direction most in line with the target's direction at an intersection in an unknown environment (Hochmair and Frank, 2000; Dalton, 2003). LEASTANGLE will use the skeleton to support actions that leave a region or a hallway to a connected region or hallway in the direction most in line with the target. LEASTANGLE considers the possible directions for movement at an intersection in the skeleton that captures broader connectivity in the environment. This is more sensitive than GREEDY, which always picks the movement most in line with the target but without regard for obstacles.

An alternative approach to generate new Advisors would be to conduct a human subject study to examine what strategies people use during navigation and then encode those strategies as Advisors. This approach will be used if the Advisors proposed thus far are unsuccessful. One could either infer human strategies from behavior during navigation or ask subjects to describe them. In the first case, subjects would navigate to several targets in a simulated environment and their navigation behavior would be analyzed for common strategies. A model could be trained on the subjects' trajectories to predict the most likely action given a situated state. The Advisor HUMANPREDICTOR would support actions based on their predicted likelihood. Alternatively, human behavior could be modeled as a Markov decision process, and inverse reinforcement learning could learn a reward function for it. The Advisor HUMANREWARD would support actions that maximize this reward function.

After a set of navigation tasks, subjects could be asked to describe their strategies in natural language to answer questions such as “Why did you do that?”, “Why did you go that way?”, “What were you thinking when you did this?”, and “Why didn’t you do something else?” To ensure inter-coder reliability, multiple coders would examine responses to these questions. The resulting categories could identify common strategies and when they were used. Finally, any non-redundant strategies would be encoded as new Advisors.

There are two potential issues that must be addressed here: scalability and parameter tuning. Currently, SemaFORR has 10 Advisors in its tier 3, and the proposed work could incorporate as many as 30 more. Although this would quadruple the number of Advisors, the architecture is expected to scale and still make decisions in real-time. For example, a previous FORR-based system that solved constraint satisfaction problems had 42 tier-3 Advisors and scaled effectively as Advisors were added (Petrovic et al., 2007). The second issue is the number of potential parameters that will need to be tuned for all the proposed tier-3 Advisors. This will be addressed empirically to ensure good performance, but also will draw from past literature where appropriate (e.g., proxemics research for the proxemics-based Advisors).

This subsection has described new approaches for tier-3 Advisors based on a variety of rationales. Because it incorporates commonsense, an enhanced spatial model, proxemics, cognitive science, and crowd avoidance strategies, this proposal diversifies SemaFORR’s reasons for taking an action to accomodate collaborative navigation. This diverse, rich set of Advisors allows a robot to further specialize its behavior and adapt to nearby people during a collaborative navigation task. The next subsection reviews new proposed tier-2 Advisors for path planning.

3.3.3 New A* planners

This subsection describes how Situated-SemaFORR will modify SemaFORR's tier 2. It first describes a new method to select a plan that satisfices multiple path-planning criteria. Then, it outlines several new planners based on rationales from human navigation.

*A** is a traditional search algorithm to find a plan that performs well with respect to some objective $\mathcal{O}$, such as "minimize distance traveled" or "avoid crowds" (Korf, 2014). A* relies on a graph whose nodes represent possible locations in a static environment. An edge between two nodes in the graph indicates that the robot can move from one location to the other. Each edge has a weight that represents the anticipated cost of that action from $\mathcal{O}$'s perspective. As A* seeks a plan through that graph, it estimates $f(s)$, the estimated total cost of a partial plan up to node s, as $g(s)$, the total of the edge costs from the start node to s, plus $h(s)$, a heuristic that estimates the cost from s to the target's node t. A* is optimal when its heuristic is *admissible*, that is, it never overestimates the actual cost to the target node.

SemaFORR's tier 2 is a set of A* path planners, each of which uses a different objective to weight the A* graph. For example, CSA* incorporates both crowd-density data and distance into *crowd-sensitive* weights on the edges of its graph and then applies A*. Thus the objective of a CSA* plan is to both avoid crowds and minimize travel distance. SemaFORR currently uses only one planner in Table 7 at a time. Although they have different objectives, they all use the Euclidean distance between s and t for $h(s)$.

In contrast, Situated-SemaFORR will have a set of planners that use their objectives to score one another's plans. Each planner's objective function will assign a *score*, a normalized value in [0,10], to plans from the other planners. A score near 10

Minimize path length	Minimize effort
Minimize travel time	Longest leg first
Minimize the number of turns	Shortest leg first
Maximize the number of turns	First noticed path
Minimize the number of curved segments	Novel path (different from previously travelled)
Maximize the number of curved segments	Avoid congestion
Minimize the number of segments in the chosen path	Avoid detours
Minimize the number of non-orthogonal intersections	Minimize negative externalities (e.g., pollution)
Minimize actual or perceived cost	Maximize aesthetics

Table 10: Human path-selection heuristics (Golledge, 1999, p. 31)

indicates that a plan closely conforms to a planner's objective; scores near 0 indicate opposition to it. Once all tier-2 plans have been scored, Situated-SemaFORR will select the plan with the most support among all planners using sum range voting

$$argmax_{P \in \Psi} \sum_{\pi=1}^{\Pi} \Theta_{\pi}(P)$$

where $\Theta_{\pi}(P)$ is the score objective π assigns to plan P. Ties will be broken at random.

A cognitively-based robot navigator should incorporate and balance a variety of path-selection heuristics. People use many different objectives to choose paths, and the objective they select has been shown to depend upon their motivation (Golledge, 1999). Table 10, adapted from Golledge, lists some of these objectives. My proposed voting-based approach selects a plan that maximally supports multiple objectives.

A new tier-2 planner could use A* to plan based on some of these human objectives. Table 11 summarizes the proposed tier-2 planners along with their objectives. (Some of the human path-selection objectives in Table 10 are not appropriate for collaborative navigation; they have been excluded here.) For example, NOVEL is a path planner whose objective is to take novel paths and avoid travel near previously visited locations. TIME estimates travel time based on the robot's speed, distance to be traveled, and constraints from potential obstacles, such as the crowd.

Advisor	Objective
Time	Minimize travel time
Turn	Minimize number of turns
Smooth	Minimize the angle between adjacent path segments
Novel	Avoid travel near previously travelled paths
Spatial	Minimize travel outside learned regions or hallways
Trail	Follow along learned trails toward the target
Barrier	Maximize travel along barriers
Conveyor	Maximize travel through high-valued conveyors

Table 11: Proposed tier-2 planners and their objectives

Inspired by cognitive science, an alternative approach would be to search in a modified A* graph. The fine-to-coarse planning heuristic was observed in a study with human subjects in a virtual environment (Wiener et al., 2004). The subjects made a detailed plan for their immediate vicinity, but an abstract one for distant locations. To apply this heuristic, Situated-SemaFORR could remove nodes in the A* graph so that, instead of nodes spaced evenly throughout Euclidean space, a node's proximity to nearby nodes would depend upon its distance from the robot. Nodes near the robot would be near each other and nodes farther from the robot would also be farther from each other. Figure 7 shows an example of how this heuristic could be applied in a graph. Plans from such a graph are more fine-grained near the robot and more "abstract" farther from it.

Other search algorithms could also be used as planners. Examples include D* search (Stentz, 1994), D* Lite (Koenig and Likhachev, 2002), LPA* (Koenig et al., 2004), and MPGAA* (Hernández et al., 2015). The drawback to all of these, and to A* as well, is that they require a heuristic to estimate the cost from a node to the target. When the robot's objective is to minimize path length then Euclidean distance is an obvious admissible heuristic. When the robot has a different objective, however, selection of an admissible heuristic is more difficult. SemaFORR's current

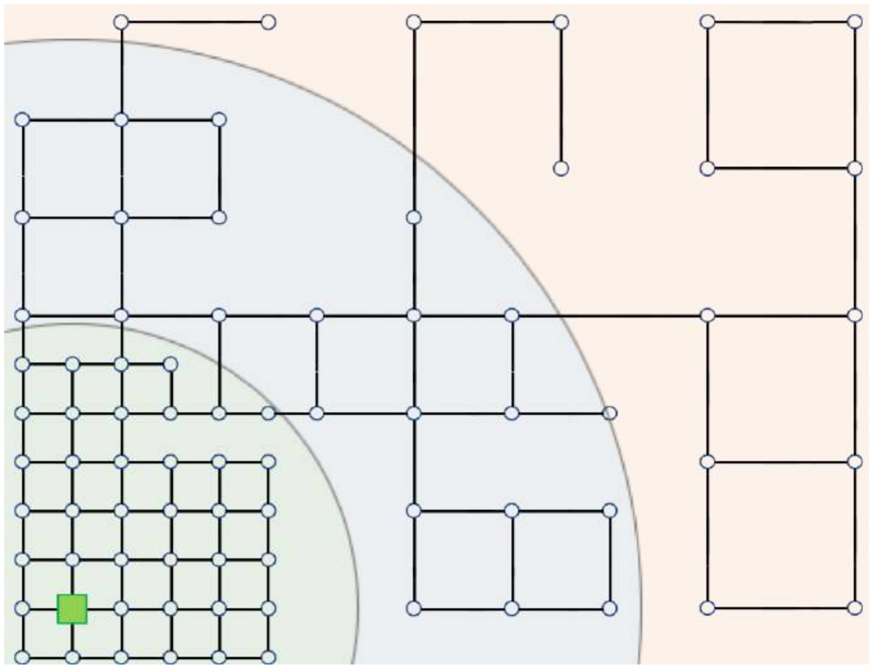


Figure 7: An example of the fine-to-coarse heuristic in a graph, where the robot's current node is the green square, and there are three levels of granularity in the graph based on distance from the robot

planners' heuristics are admissible because they treat the non-distance-based cost to the target node as 0, which is always an underestimate of the true cost to the goal. For example, CSA* combines distance and crowdedness in its edge costs for $g(s)$, but its heuristic $h(s)$ only considers Euclidean distance and sets the heuristic cost estimate from the crowd to 0. When the heuristic is always 0, however, A* search is equivalent to breadth-first search, which is costly. The next subsection describes a new proposed search algorithm, inspired by human search-and-rescue techniques, that could be used instead.

3.3.4 Search-and-rescue planning algorithm

Chapter 2 argued that a human-aware robot navigator should be natural and sociable, and that such behavior is facilitated by the use of human rationales for navigation.

Although this proposal incorporates human rationales to select an action and human objectives for path planning in a cognitive architecture, SemaFORR has thus far only planned with versions of A*, which is not based on human behavior. The *Search-and-rescue planning algorithm* (SARPA) is a cognitively-based alternative.

SARPA is a new proposed population-based metaheuristic that explores a search space similarly to the way people search an environment. It both incorporates models of pedestrian movement and exploits learned knowledge the way people do during search-and-rescue missions. Swarm-based metaheuristics (e.g., artificial bee colony (Karaboga and Basturk, 2007), particle swarm optimization (Kennedy and Eberhart, 1995)) draw from animal behavior, evolution, and other natural processes. SARPA, while similar to them is, to the best of this author's knowledge, the first search algorithm that also draws on human search behavior in two-dimensional space.

Without many simplifying assumptions, it is difficult to prove that a metaheuristic is guaranteed to converge to a global optimum or even a local optimum in a satisfactory amount of time. Instead, metaheuristics' convergence is usually evaluated empirically. Thus, SARPA could be empirically evaluated on standard search tasks. Its performance could be compared with other state-of-the-art metaheuristic and hybrid metaheuristic approaches (Boussaïd et al., 2013; Blum et al., 2011). Although it will not be provably optimal (as A* is), SARPA avoids the need to find an admissible heuristic for objectives other than shortest path.

SARPA uses *searchers*, artificial agents, to explore the same objective-oriented graph used by A*, but it is inspired by real-world search-and-rescue missions. Human behavior on these missions represents a manifestation of people's underlying cognitive processes of problem solving and decision making. Typically in search and rescue, people are divided into multiple teams, each of which uses a different strategy to search for a missing person (Wilfong, 2004). People use helicopters to identify potential areas for on-the-ground teams to explore. A hasty team is a rapidly-deployed team that goes as quickly as possible to the locations in the search area where the missing person is most likely to be. Search-and-rescue units also use dogs to track and follow the scent of the missing person, and may eventually resort to a grid-search team that moves methodically along a predefined search pattern to comb an area exhaustively. Despite the considerable time and effort it entails, a grid team often finds valuable clues about the missing person. Although SARPA is not an exact replica of real-world search-and-rescue missions (particularly because the target's location is known to the searchers), these inspirations form SARPA's cognitive basis.

SARPA partitions its searchers into four teams inspired by real-world scenarios: helicopter, hasty, dog, and grid. These teams explore a search space to find a satisfactory *complete plan*, a plan in the graph from a start node to a target node. A *partial plan* represents a plan in the graph that does not arrive at the target. The *quality* of a complete plan is measured by the planner's objective $\mathcal{O}$ as the total cost of the edges in the plan. SARPA seeks a satisfactory complete plan both with *local search* in the vicinity of a node, and with *global search* through the entire search space. The dog and hasty teams do hill-climbing local search, the grid team does exhaustive local search, and the helicopter team does global search.

SARPA will proceed in stages. First the helicopter team will explore randomly, and then the hasty team will greedily search areas found by the helicopter team. Next, the grid team will search exhaustively based on predefined search patterns, and finally the dog team will search greedily within areas found by the grid team. Each team retains the best complete plan that it finds, and this sequence is repeated until a satisfactory plan emerges or a time limit is exhausted.

The input to SARPA is an objective-oriented graph of the environment, the robot's location, the target's location, and values for its parameters. Table 12 lists the full

General	Search patterns
• Number of individuals in each team • Overall search time limit • Helicopter search time limit • Hasty search time limit • Grid search time limit • Dog-team search time limit	• Track space • Search area boundary • Sector radius • Sector angular displacement

Collective behaviors	Movement behaviors
• Positive feedback bias to visit node • Random fluctuation likelihood • Negative feedback iteration threshold • Negative feedback similarity threshold • Negative feedback threshold for number of similar partial plans	• Count maximum • Count decay rate • Count bias to visit node • Count node avoidance bias • Social force velocity

Table 12: Parameters for SARPA

suite of SARPA's parameters, most of which specify how searchers move through the graph. These will be set empirically to ensure quick convergence and maximize resource usage. (In general, a drawback of metaheuristics is that parameters must be hand-tuned to achieve satisfactory performance. SARPA could also include a method for automated parameter tuning to avoid hand-tuned parameters.)

Algorithm 6 is pseudocode for SARPA. Each searcher performs its designated role based on its team. First, each searcher in the helicopter team constructs a partial plan. A helicopter searcher initially begins its partial plan at the start node and randomly selects an edge from it. The likelihood of edge selection is subject to positive feedback, discussed below. The new node on that edge is appended to the partial plan as a waypoint and the process repeats from there. To prevent cycles, a helicopter plan never returns to a previously incorporated node. A helicopter searcher continues until a time limit is exhausted, the last waypoint is the target, or there are no unused edges from the last waypoint. As a helicopter searcher explores, it saves in memory each visited node and the partial plan that reached it. Then, on subsequent

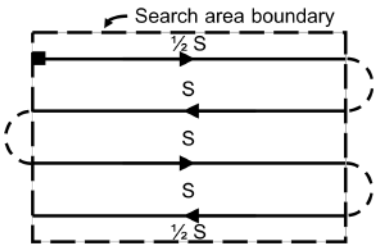
Algorithm 6: Search-and-rescue planning algorithm

```
Input: graph of the environment, robot's and target's locations, and
  parameter values
Output: Best, a satisfactory plan
repeat until Best is satisfactory or search time is exhausted
    for each helicopter searcher do
        Construct a random partial plan until stopping condition is satisfied
        Save visited nodes and partial plans
    end
    $P_H \leftarrow$ the best quality helicopter complete plan
    for each hasty searcher do
        Select a plan from among the helicopter teams' plans
        Modify that plan until stopping condition is satisfied
    end
    $P_A \leftarrow$ the best quality hasty complete plan
    for each grid searcher do
        Follow a random search pattern until stopping condition is satisfied
        Save visited nodes and partial plans
    end
    $P_G \leftarrow$ the best quality grid complete plan
    for each dog-team searcher do
        Select a plan from among the grid teams' plans
        Modify that plan until stopping condition is satisfied
    end
    $P_D \leftarrow$ the best quality dog team complete plan
    Update counters based on the hasty and dog teams
    Best $\leftarrow$ complete plan with maximum quality among $(P_H, P_A, P_G, P_D)$
return Best
```

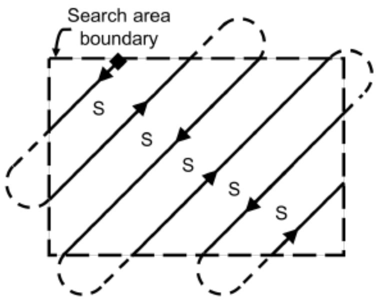
iterations, a helicopter searcher randomly selects among any of its explored nodes and randomly constructs a new plan from there. The best-quality complete helicopter plan is saved as P_H. For example, a helicopter searcher may construct a partial plan with 5 waypoints $P_1 = \langle s, w_1, w_2, w_3, w_4 \rangle$ on its first iteration. On the next iteration, it randomly selects w_3 from P_1 and then combines the partial plan up to w_3 and a new expansion from w_3 to produce a new partial plan $P_2 = \langle s, w_1, w_2, w_3, w_6, w_7, w_8, w_9 \rangle$.

Next, SARPA calls in the hasty team. If any helicopter plans are complete, then each hasty searcher selects among them, with selection likelihood in proportion to the complete plan's quality. Otherwise, each hasty searcher selects a partial plan at random from the helicopter team. Then, each hasty searcher greedily modifies its selected plan by randomly selecting one of three processes: extend, prune, or change. To *extend* a partial plan, a searcher examines all the edges from its last waypoint and adds the waypoint associated with the lowest cost edge. To *prune* a partial plan, a searcher removes its last waypoints until there is some node with an alternative edge. A searcher cannot extend to a pruned waypoint in the same iteration. Finally, to *change* a partial plan, a searcher randomly selects two of its waypoints, constructs a new sequence of waypoints that connects them, and replaces the original plan segment with the new one if it has lower total edge cost. (If a plan is complete, a hasty searcher can only change it.) A hasty searcher continues to extend, prune, or change until it reaches a time limit or the last waypoint is the target. The best-quality complete hasty plan is saved as P_A.

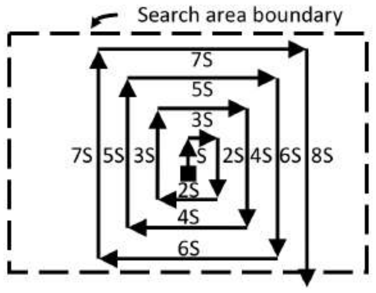
Next, SARPA calls in the grid team. Each grid searcher uses one of five search patterns inspired by real search-and-rescue strategies: parallel track, creeping line, expanding square, sector, and contour (National Search and Rescue Council, 2017). The *track space* is the distance between parts of the environment that are searched. The *search area boundary* is the limits of the area being searched. The *parallel track* search pattern, shown in Figure 8(a), explores along horizontally-aligned parallel lines that are track space apart within the search area boundary. Shown in Figure 8(b), *creeping line* is similar to the parallel track but searches along lines aligned with the minor diagonal within the search area boundary. The *expanding square* search pattern in Figure 8(c) explores in squares that gradually enlarge from a central location, with track space between each square and the maximum expansion limited by the search



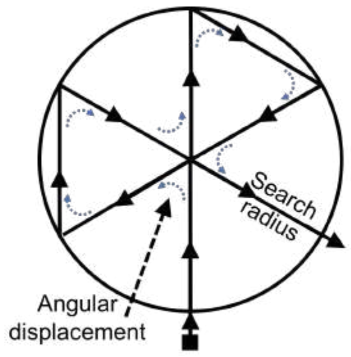
(a) Parallel track



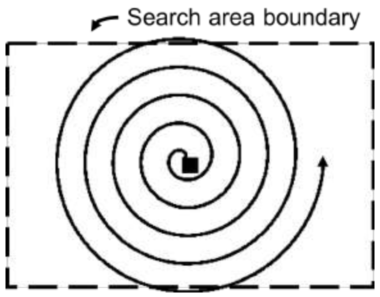
(b) Creeping line



(c) Expanding square



(d) Sector



(e) Contour

Figure 8: Examples of SARPA's search patterns modelled from (National Search and Rescue Council, 2017, pp. 150–158). S is the track space

area boundary. The *sector* search pattern, shown in Figure 8(d), explores within a circular area along segments. The size of the circle is determined by a radius and the segment sizes are specified by an angular displacement. Finally, Figure 8(e) shows the *contour* search pattern, which starts at a point and explores in a spiral up to the search area boundary.

On each iteration, each grid searcher randomly selects one of the five search patterns in Figure 8, and applies it within the preset search area boundary. Initially, each grid searcher begins at the start node, but on subsequent iterations, grid search begins at the node it ended at in the last iteration. To apply a search pattern, a grid searcher extends its partial plan in the graph in the direction indicated by the search pattern. For example, Figure 9 shows a grid searcher's partial plan in a graph

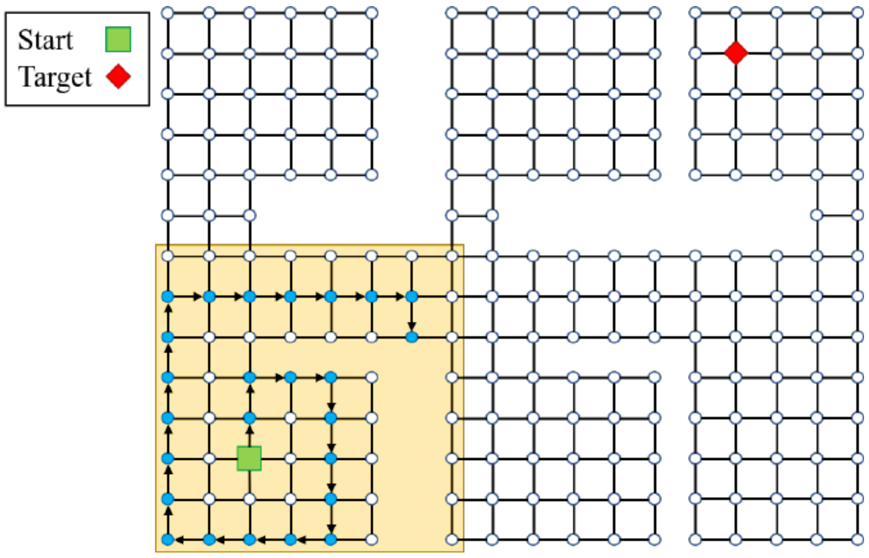


Figure 9: An example of the expanding square search pattern in a graph with a track space of two and where the search area boundary is an 8 by 8 square

as it follows an expanding square pattern. A grid search stops when the searcher has completed its search pattern, arrives at a node where no edge can continue the pattern, or a time limit is exhausted. A grid searcher saves in memory each visited node and the partial plan that reached it. The best-quality complete grid plan is saved as P_G.

Finally, SARPA calls in the dog team. If any grid plans are complete, then each dog-team searcher selects among them, with selection likelihood in proportion to the plan's quality. Otherwise, each dog-team searcher selects a partial plan at random from the grid team, and each dog-team searcher randomly either extends, prunes, or changes its partial plan until it exhausts its time limit or the last waypoint is the target. (If a plan is complete, a dog-team searcher can only change it.) The best-quality complete dog-team plan is saved as P_D. After all four search teams have finished, *Best* is the complete plan with the best quality among P_H, P_A, P_G and P_D. SARPA repeats this process until *Best* is satisfactory or search time is exhausted.

Recall from Section 2.1 that groups of social beings display several collective behaviors: positive feedback, negative feedback, random fluctuations, and interactions among individuals. SARPA applies these four characteristics to individual searchers. Here, positive feedback encourages searchers to explore near members of the same team in an effort to exploit knowledge from other searchers. Specifically, SARPA searchers in the helicopter team will be more likely to visit a node if other helicopter searchers have visited it. Similarly, hasty searchers will be more likely to visit a node visited by other hasty searchers, as will dog-team searchers for a node visited by other dog-team searchers.

Negative feedback in SARPA begins either when a searcher has had similar partial plans for several iterations or when the number of nearby members of the same team reaches a specified threshold. Like tabu search (Glover, 1989, 1990), negative feedback temporarily forbids search to return to recent partial plans. This limits exploitation in an effort to increase exploration. Helicopter, hasty, and dog-team searchers all record the number of iterations they have devoted to partial plans with a high degrees of similarity, where similarity is measured by the percentage of common waypoints. They abandon a partial plan when it passes some threshold and are forced to explore a new partial plan with low similarity to the one just abandoned. They also monitor the number of searchers from their own team with similar partial plans. When the number of searchers with similar partial plans surpasses a threshold, a searcher must abandon its partial plan. In both cases, a searcher is prevented from returning to its abandoned partial plan for a specified number of iterations.

With some small likelihood, random fluctuation explores a worse complete plan. Specifically, the hasty and dog teams may occasionally select a complete plan of lower quality, and a grid searcher may deviate from its search pattern. Random fluctuations provide a way to escape from local optima. SARPA also includes interactions among individuals through the information shared between the two pairs of teams (the helicopter team with the hasty team, and the grid team with the dog team).

SARPA draws inspiration from the two cognitive models of pedestrian movement discussed in Section 2.1: trail formation and social force. To adapt the trail formation model to SARPA, each node in the graph tallies the times it is visited by the hasty team and the dog team. These counters have a maximum value and decay over time. Hasty and dog-team searchers will be more likely to visit a node with a higher count. Analagously, a grid searcher is more likely to deviate from the next node in its search pattern when it has a high count. Thus, these counters encourage exploitation by the hasty and dog teams and exploration by the grid team.

SARPA also integrates the social force model in the way searchers' explore. Members of the same team repel one another. For example, consider a searcher building its partial plan, and about to add another node. If another searcher from the same team is at a nearby node, then the likelihood that first searcher will select an edge that moves it farther from second searcher would be increased, based on the social force velocity parameter. This represents repulsive forces among pedestrians in the social force model. It is counterbalanced by the hasty and dog-team searchers' attractive force towards better quality complete plans, grid searchers' intention to follow the search pattern, and helicopter searchers' positive feedback. These repulsive forces incorporate more exploration into SARPA.

In summary, SARPA is a novel swarm-based metaheuristic that draws from search-and-rescue techniques and models of human-group behavior to explore a search space. Each of its four teams of searchers has its own search strategy, and balances exploration and exploitation. SARPA seeks a satisfactory plan to the target, and will be evaluated empirically against other metaheuristics. SARPA could also be extended to run its two pairs of teams in parallel or to apply bidirectional search.

The last two subsections proposed tier-2 path planners for Situated-SemaFORR that draw from human planning objectives. Two search algorithms were proposed to use with these novel path planners (A* and SARPA), and the fine-to-coarse heuristic could be applied to modify the graph for them. These proposed methods are intended to improve navigation performance, and to incorporate human-like reasoning that may facilitate natural human-robot interaction. The next subsection combines tier-3 Advisors with weight learning and voting to produce situated-decision strategies.

3.3.5 Apply weight learning and voting methods

A situated-decision strategy χ either selects an action χ_h^a for a given situation U_h or produces a plan χ_h^P for a given dynamic situation D_h. This subsection reviews multiple approaches to learn situated-decision strategies for tier 3, which specialize the robot's behavior based on its (learned) situation. A situated-decision strategy is learned only from successfully completed tasks.

The most basic approach for χ_h^a is to select the action most commonly taken among the states in U_h. For example, if 30 states are clustered in U_h, and in 20 of them the selected action was "move forward 1 meter", then χ_h^a selects that action whenever the robot is in U_h. This approach is fast, efficient, and will always produce some χ_h^a. This approach can also be used online to update the χ_h^a after each decision. This method, however, may not be optimal or even satisfactory; it simply repeats the robot's past behavior, whether or not that is worthwhile. Indeed, the first action taken in a situation may have been a random choice. A better situated-decision strategy would incorporate the robot's objective to measure how well the strategy performs.

One way to address this is to learn situation-specific weights for the Advisors. Recall that c_{ia} is the comment strength of Advisor i on action a. Now, for a situation

U_h, each tier-3 Advisor is given a weight $\alpha_{hi} \in [0,1]$ based on some criterion and then an action is selected with weighted sum range voting

$$argmax_{a \in A} \sum_{i=1}^{n} \alpha_{hi} * c_{ia}.$$

Ties will be broken at random.

There are several ways to learn the appropriate weights for a given situation. One approach draws from *Relative Support Weight Learning* (RSWL), a method for weight learning that was developed for a FORR-based system for constraint solving (Petrovic and Epstein, 2006). RSWL first computed an Advisor's *relative support* for an action as the normalized difference between that Advisor's value for that action and the average value it assigned to all available actions. An Advisor is considered to support an action if the relative support is positive, and oppose an action if it is negative. RSWL determined whether an Advisor supported or opposed the action that was best suited to solve the constraint problem and appropriately rewarded or penalized that Advisor's weight. RSWL can be adapted to SemaFORR to learn situated-decision strategies. Here, a tier-3 Advisor's relative support ρ_{ia} for an action a is $\rho_{ia} = (c_{ia} - \bar{c_i})/\sigma_i$ where $\bar{c_i}$ is the mean of Advisor i's comment strengths and σ_i is their standard deviation. Advisor i supports action a if $\rho_{ia} > 0$ and opposes it if $\rho_{ia} < 0$.

RSWL is a form of reinforcement learning, and thus it requires knowledge about which actions to reinforce. One way to determine the "best" action $\beta \in A$ for a situation is to select the action that would have brought the robot closest to the nearest trail marker from the trail produced after that task, because a trail is usually shorter than the original path and provides a more direct, albeit likely imperfect, route to the target. Another way to identify β is to select the action that would have brought the robot closest to the nearest waypoint in the direction of the target in the robot's plan during that task. Then, given β, Advisors' weights are increased if they supported β and decreased if they opposed it. All Advisors would have the same initial weight. This process is repeated for all the states in a situation. The final learned weights would then be used with weighted sum range voting to determine χ_h^a for that situation.

Instead of RSWL, a strategy could find the set of weights for the Advisors that most often results in the Advisors selecting β for a given situation. Given a set of weights, the action a' that would be selected can be computed for each state in a situation with weighted sum range voting. This strategy seeks the set of weights that maximizes the number of states in U_h that would produce $a' = \beta$. For example, if 30 states are clustered in U_h, and β was "move forward 2 meters", then this strategy learns the set of weights on the Advisors that would have resulted in the robot selecting that action in as many of the 30 states as possible. A metaheuristic (e.g., simulated annealing or a genetic algorithm) could be used to find the sets of weights for this strategy. The final learned weights would then be used with weighted sum range voting as the χ_h^a for a situation. Regardless of how the weights are learned, the set of weights learned for one situation could be used as the initial weights for other situations in the same condition.

Tier 3 uses sum range voting to select an action. Recall, however, the many voting methods in Section 2.3. Since none is known to be the best, one possible approach is to employ an ensemble of voting methods and select the action that wins the most of them. Each voting method selects a winner, and then the action with the most wins is selected as the overall winner. Any ties are broken at random. To the best of this author's knowledge no work thus far has employed such an ensemble of voting methods.

Both sum range and product range voting are readily applied to Advisors' strengths. Other voting methods, however, require some additional computation. To apply plurality and majority, each Advisor votes only for the candidate to which it gave the highest strength. With this formulation, the methods in Tables 1 and 2 can be applied. Ranking-based voting methods (Table 3) will rank candidates by their strength in descending order and then use that ranking to select a winner. With some additional computation, the remaining voting methods (Tables 4 and 5) can also be used here. In all, there are 20 different voting methods that can be used to decide which candidate to select. An ensemble of voting methods would apply them all and select the winning candidate as the one selected by the most voting methods. This meta-voting approach could balance the different fairness criteria associated with the voting methods.

Weight learning for Advisors may be brittle because their comment strengths are based on the same sensor inputs and spatial model. Two approaches could nonetheless apply voting to learn Advisor weights. The first would learn which voting method selects the desired action β for a given situation; the one that most frequently selects β for a situation is used as the voting method for that situation. Thus the χ_h^a uses different voting methods for different situations. The second approach assigns a weight to each voting method so that the overall winner is selected as a weighted sum of the winners from each voting method. The weights could be learned by a metaheuristic to select β for a given situation, or they could be set by RSWL. For example, in an RSWL approach, the weights for a voting method would be increased if it selected the winner as β, and decreased if it did not.

An alternative approach to algorithmic weight learning and voting methods is to weight the Advisors so that the robot is more human-like. One way is to bias the Advisor weights to elicit certain kinds of human behavior (e.g., aggressive, cautious, or inquisitive). Another is to calculate weights from human behavior. Such a study would observe subjects as they navigated to several targets in a simulated environment, and then use a metaheuristic to learn weights for the Advisors that most closely mirror the subjects' behavior in a situation. The resultant weight scheme would be human-like. A second approach could show human subjects a situation, ask them to select the action the robot should take in that situation, and assign Advisor weights that would replicate human decisions. A third approach would be to show subjects a robot as it navigates with different weight schemes, ask them to select the one that looks most natural and transparent, and use those weights for the Advisors.

The situated-decision strategies discussed thus far select a single action for a given vector or geometric situation. For a dynamic situation D, a situated-decision strategy χ_h^P should select a sequence of actions, rather than a single action. For the exemplars in the dynamic situation $(e_1, e_2, \ldots, e_l) \in D$, one approach is to combine their respective strategies, $\chi_h^P = (\chi_1^a, \chi_2^a, \ldots, \chi_l^a)$. Then, whenever the robot is in the first state in the dynamic situation, χ_h^P dictates that it follows that sequence of actions. Another approach is to find the sequence of actions that most quickly brings the robot from e_1 to e_l in D and prescribe that sequence as the χ_h^P for that dynamic situation.

This subsection has identified several potential approaches to learn situated-decision strategies. It has also described potential human-subject studies to examine how people balance various decision-making rationales during navigation. Finally, it discussed methods that produce a situated-decision strategy for a dynamic situation. These methods should allow Situated-SemaFORR to balance its Advisors in a more intuitive and natural way.

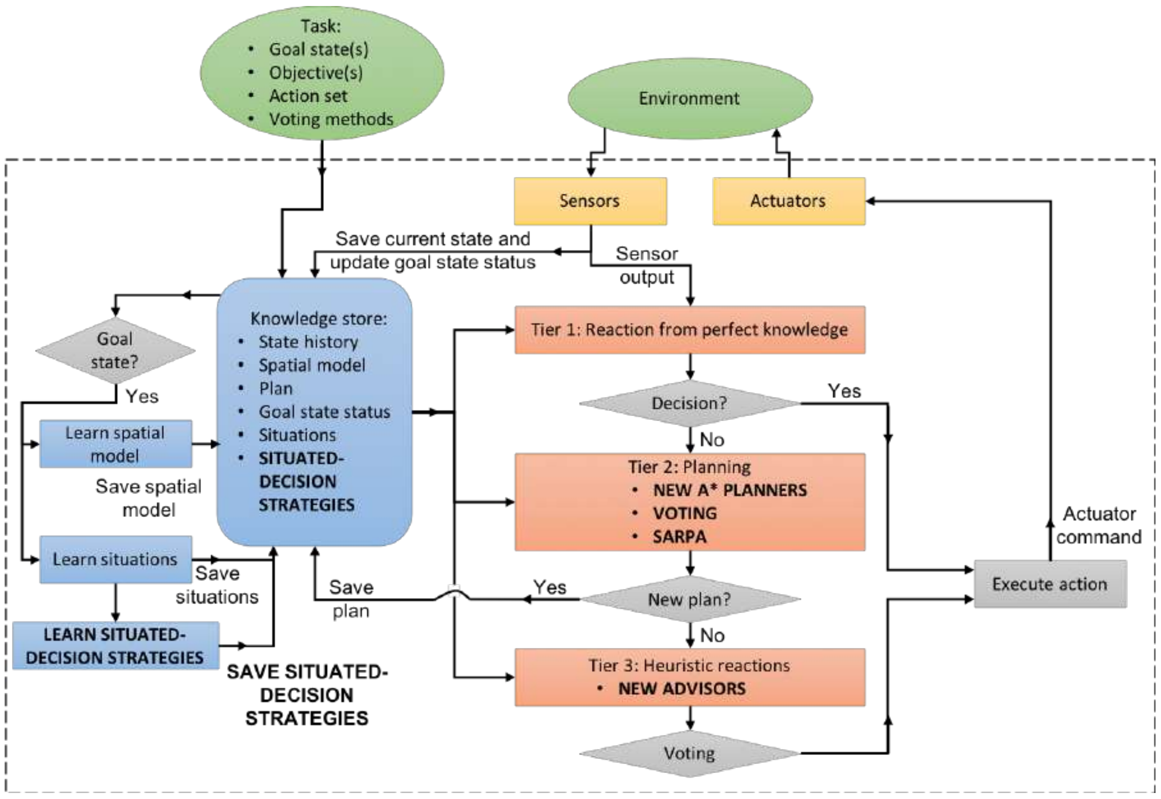


Figure 10: Situated-SemaFORR's architecture with situated-decision strategies. Proposed work introduced in this section is indicated in uppercase.

3.3.6 Preliminary work

After each task, situations will be learned and then situated-decision strategies will be learned. Figure 10 shows the architecture with the situated-decision strategies incorporated. After situated-decision strategies are learned, the appropriate weights or voting methods are applied based on the robot's current situation before a decision is made.

In preparation for situated-decision strategies, I have implemented several of the proposed tier-3 Advisors, including CURIOSITY, LEARNSPATIALMODEL, ACCESS, ENTERDOOR, and EXITDOOR. I have also parameterized the configuration of Advisors in SemaFORR, which has streamlined the implementation and testing of new Advisors.

This section has described new tier-2 and tier-3 Advisors, proffered a new meta-heuristic for path planning, and proposed approaches to learn situated-decision strategies. Situated-SemaFORR should be more human-like and cognitively plausible because it draws upon weight learning, voting methods, and cognitive science to balance its various decision-making rationales. This suggests that Situated-SemaFORR will be better suited for collaborative navigation and its behavior will be more transparent to human collaborators. It is difficult to predict a priori that the result will empirically improve collaborative navigation, although related HRI work suggests that there will be an improvement when a robot's behavior is more comfortable, sociable, and natural. The next chapter describes how Situated-SemaFORR could explain its reasoning in natural language, and how the proposed work will be evaluated.

4. Proposed work: explanation and evaluation

The previous chapter proposed work inspired by cognitive science and HRI to make a robot's behavior more human-like and transparent. Effective collaboration between a robot and a person, however, also requires natural communication. When a robot travels with a human companion, the robot should be able to explain its navigation behavior in natural language. This chapter describes several methods I developed to explain Situated-SemaFORR's behavior in natural language, based upon the robot's commonsense, its qualitative reasoning, and its learned spatial model. These methods explain a robot's reactive decisions, its plans, and learned situations. The chapter concludes with a proposed evaluation plan.

4.1 Apply meaningful semantics

To build trust and understanding in its human collaborator, a robot should produce *natural explanations*, human-friendly reasons for its behavior in natural language (Kulesza et al., 2013). Such transparent, intelligible communication enables the robot to gain social acceptance and reduce confusion about its abilities. A natural explanation is more than a description of an event or a summary of the event's causes. Rather, such an explanation compares counterfactual cases, is selective about which causes are included, and recognizes that the human companion is a social being with her own beliefs and intentions (Miller, 2017). Situated-SemaFORR's cognitive basis facilitates its ability to provide natural explanations for its underlying decision making mechanism. This section describes WHY-DECISION to explain Situated-SemaFORR's reactive reasoning and WHY-PLAN to explain Situated-SemaFORR's navigation plans. It also describes how these will be combined to create WHY, a general explanation mechanism for Situated-SemaFORR that also addresses learned situations and their associated strategies. Why is more broadly applicable, however, and could give natural explanations for other robot controllers.

4.1.1 Why-Decision

Why-Decision explains a reactive navigation decision in natural language. It anticipates three likely questions from a human companion: "Why did you decide to do that?" "Why not do something else?" and "How sure are you that this is the right decision?" Why-Decision exploits Situated-SemaFORR's reasoning hierarchy and decision rationales to generate natural explanations in response to these three questions. The result is a rich, varied set of natural explanations. Why-Decision interprets SemaFORR's cognitive foundation to bridge the perceptual and representational gap between human and robot navigators. Why-Decision and Situated-SemaFORR could accompany any robot controller to provide natural explanations. Why-Decision has already been implemented as Why in Korpan et al. (2017). This subsection describes its preliminary implementation and then proposes extensions and improvements.

The first likely question asks why the robot chose a particular action. Why-Decision constructs its answer from the rationales and comments of the Advisors responsible for that choice, with templates to translate actions, comments, and decisions into natural language. Given the robot's current pose, Why-Decision maps each possible action onto a descriptive phrase for use in any [action] field. Examples include "wait" for a forward move of 0.0 meters, "inch forward" for a forward move of 0.2 meters, and "shift right a bit" for a turn in place of 0.25 radians.

Algorithm 7 is pseudocode for Why-Decision's explanation procedure for the first question. It takes as input the current situated state, target location, and spatial model, and then calculates its response based on the comments from Situated-

Algorithm 7: WHY-DECISION's explanation procedure

```
Input: situated state, target location, spatial model
Output: explanation
switch mode(decision) do
    case tier 1 decides action do
        explanation ← sentence based on VICTORY or ENFORCER
    case only 1 unvetoed action remains after tier 1 do
        explanation ← sentence based on vetoes from AVOIDWALLS
    otherwise do
        Compute relative support for tier-3 Advisors' strengths
        Categorize the support level for the chosen action
        Complete template for each Advisor with its support level and rationale
        explanation ← combined completed templates
    end
end
return explanation
```

SemaFORR's Advisors. There are three possibilities: tier 1 chose the action, tier 1 left only one unvetoed action, or tier 3 chose the action. Situated-SemaFORR only makes a decision in tier 1 if VICTORY or ENFORCER mandates it or AVOIDWALLS has vetoed all actions but the pause. NOTOPPOSITE will never select the robot's action because it can never veto more than half the turns. The applicable templates in those cases are "I could see our [target/waypoint] and [action] would get us closer to it" and "I decided to wait because there's not enough room to move forward."

The inherent uncertainty and complexity of a tier-3 decision, however, requires a more nuanced explanation. Recall, $c_{ia} \in [0, 10]$ is the comment strength of Advisor i on action a. Recall also that a tier-3 Advisor's relative support ρ_{ia} for an action a is $\rho_{ia} = (c_{ia} - \bar{c_i})/\sigma_i$ where $\bar{c_i}$ is the mean of Advisor i's comment strengths and σ_i is their standard deviation. (This is not a z-score because sampled values replace the unavailable true population mean and standard deviation.) WHY-DECISION can compare different Advisors' relative support because they have common mean 0 and standard deviation 1. If $|\rho_{ia}|$ is large, Advisor i has a strong opinion about action

	c_{ia}				ρ_{ia}			
Tier-3 Advisors	a_1	a_2	a_3	a_4	a_1	a_2	a_3	a_4
1	0	1	1	10	-0.64	-0.43	-0.43	1.49
2	0	8	9	10	-1.48	0.27	0.49	0.71
3	2	0	10	2	-0.34	-0.79	1.47	-0.34
4	3	10	1	0	-0.11	1.44	-0.55	-0.78
$\mathcal{C}$	5	19	21	22				

Table 13: Example of comments from tier-3 Advisors $i = 1, 2, 3, 4$ on actions a_1, a_2, a_3, a_4, where c_{ia} is the comment strength and ρ_{ia} is relative support

a relative to the other actions; this opinion supports a if $\rho_{ia} > 0$ and opposes it if $\rho_{ia} < 0$.

Table 13 provides a running example. It shows the original comment strengths from four tier-3 Advisors ($i = 1, 2, 3, 4$) on four actions (a_1, a_2, a_3, a_4), and the total comment strength $\mathcal{C}$ for each action. Tier 3 chooses action a_4 because it has maximum support. While Advisors 1 and 2 support a_4 with equal strength, the relative support values tell a different story: Advisor 1 prefers a_4 much more ($\rho_{14} = 1.49$) than Advisor 2 does ($\rho_{24} = 0.71$). Moreover, Advisors 3 and 4 actually oppose a_4 (-0.34 and -0.78, respectively).

Each measure in Table 14 was assigned a descriptive natural language phrase and a partition of the real numbers into three intervals. The partition allows Why-Decision to hedge in its responses, much the way people explain their reasoning when they are uncertain (Markkanen and Schröder, 1997). For example, Why-Decision maps the relative support values into three intervals. For a_4 in the running example, Advisor 1's relative support of 1.49 is translated as "want" and Advisor 4's -0.78 is translated as "don't want". Why-Decision then completes the clause template "I [phrase] to [rationale]" for each Advisor based on Table 14 and less model-specific language from Table 7. For example, if Advisor 1 were Greedy in the running example, then the completed clause template for a_4 would be "I want to get close to

relative support $\rho_{ia} \leq 0$: opposed	$(-\infty, -1.5]$	really don't want
	$(-1.5, -0.75]$	don't want
	$(-0.75, 0]$	somewhat don't want
relative support $\rho_{ia} > 0$: supportive	$(0, 0.75]$	somewhat want
	$(0.75, 1.5]$	want
	$(1.5, +\infty)$	really want
Level of agreement γ_a	$(0.45, 0.5]$	My reasons conflict
	$(0.25, 0.45]$	I've only got a few reasons
	$[0, 0.25]$	I've got many reasons
Overall support ζ_a	$(-\infty, 0.75]$	don't really want
	$(0.75, 1.5]$	somewhat want
	$(1.5, +\infty)$	really want
Confidence level Λ_a	$(-\infty, 0.0375]$	not
	$(0.0375, 0.375]$	only somewhat
	$(0.375, +\infty)$	really
Difference in overall support $\zeta_a - \zeta_{a'}$	$(0, 0.75]$	slightly more
	$(0.75, 1.5]$	more
	$(1.5, +\infty)$	much more

Table 14: Phrase mappings from value intervals to language

the target." Any references to the target will be modified to a waypoint if there is an active plan selected by tier 2.

Finally, Why-Decision combines completed clause templates into a final tier-3 explanation, but omits language from Advisors with relative support values in $(-0.75, 0.75]$ because they contribute relatively little to the decision. Why-Decision concatenates the remaining language with appropriate punctuation and conjunctions to produce its tier-3 explanation: "(Although [language from opposed Advisors],) I decided to [action] because [language from supporting Advisors]". The portion in parentheses is omitted if no opposition qualifies. If the Advisors in the running example were Greedy, ElbowRoom, Convey, and Explorer, in that order, and a_4 were move forward 1.6 m, then the natural explanation is "Although I don't want to go somewhere I've been, I decided to move forward a lot because I want to get close to our target." (Note that support from Advisors 2 and 3 fails the -0.75 filter and so they are excluded.)

This approach can also respond to "What action would you take if you were in another context?" Given the situated state and the target location, WHY-DECISION would reuse its current spatial model, generate hypothetical comments, and process them in the same way. The sentence template would substitute "I would [action]" for "I decided to [action]."

The second likely question from a human collaborator is about the robot's confidence in its decision, that is, to what extent it believes its decision will help reach the target. Again, WHY-DECISION responds based on the tier that selected the action with a procedure similar to Algorithm 7. Tier 1's rule-based choices are by definition highly confident. If VICTORY or ENFORCER chose the action then the response is "Highly confident, since [our target/the next waypoint in our plan] is in sensor range and this would get us closer to it." If AVOIDWALLS vetoed all forward moves except the pause, then the explanation is "Highly confident, since there is not enough room to move forward."

Again, tier-3's uncertainty and complexity require more nuanced language, this time with two measures: level of agreement and overall support. The extent to which the tier-3 Advisors agree indicates how strongly the robot would like to take the action. WHY-DECISION measures the level of that agreement with Gini impurity, where values near 0 indicate a high level of agreement and values near 0.5 indicate disagreement (Hastie et al., 2009). For n tier-3 Advisors and maximum comment strength 10, the *level of agreement* $\gamma_a \in$ [0,0.5] on action a is defined as

$$\gamma_a = 2 \cdot \left[\frac{\sum_{i=1}^{n} c_{ia}}{10n}\right] \cdot \left[1 - \frac{\sum_{i=1}^{n} c_{ia}}{10n}\right].$$

In the running example of Table 13, the level of agreement on a_4 is

$$\gamma_4 = 2 \cdot \left[\frac{22}{40}\right] \cdot \left[1 - \frac{22}{40}\right] \approx 0.50.$$

This indicates considerable disagreement among the Advisors in Table 13.

The second confidence measure is Situated-SemaFORR's overall support for its chosen action compared to other possibilities, defined as a t-statistic across all tier-3 comments. Let $\mu_{\mathcal{C}}$ be the mean total strength $\mathcal{C}$ of all actions under consideration by tier 3, and $\sigma_{\mathcal{C}}$ be their standard deviation. The *overall support* for action a is $\zeta_a = (\mathcal{C}_a - \mu_{\mathcal{C}})/\sigma_{\mathcal{C}}$. ζ_a indicates how much more the Advisors as a group would like to perform a than the other actions. In Table 13, the overall support ζ_4 for a_4 is 0.66, which indicates only some support for a_4 over the other actions.

WHY-DECISION weights level of agreement and overall support equally to gauge the robot's confidence in a tier-3 decision with *confidence level* $\Lambda_a = (0.5 - \gamma_a) \cdot \zeta_a$ for a. It then maps each of Λ_a, γ_a, and ζ_a to one of three intervals and then to natural language, as in Table 14, with implicit labels *low* < *medium* < *high* in order for each statistic. Two statistics *agree* if they have the same label; one statistic is lower than the other if its label precedes the other's in the ordering.

All responses to this question use a template that begins "I'm [Λ_a adverb] sure because...." If γ_a and ζ_a both agree with Λ_a, the template continues "[γ_a phrase]. [ζ_a phrase]." For example, "I'm really sure about my decision because I've got many reasons for it. I really want to do this the most." If only one statistic agrees with Λ_a, the template continues "[phrase for whichever of γ_a or ζ_a agrees]." For example, "I'm not sure about my decision because my reasons conflict." Finally, if neither statistic agrees with Λ_a, it concludes "even though [phrase for whichever of γ_a or ζ_a is lower than Λ_a], [γ_a phrase or ζ_a phrase for whichever is higher than Λ_a]." For example, "I am only somewhat sure about my decision because, even though I've got many reasons, I don't really want to do this the most." For a_4 in Table 13, Λ_4 is near 0, $\gamma_4 = 0.50$, and $\zeta_4 = 0.66$. This produces the natural explanation "I'm not sure about my decision because my reasons conflict. I don't really want to do this more than anything else."

A human collaborator makes decisions with her own mental model of the environment. When her decision conflicts with another team member's, she tries to understand why they made a different decision. WHY-DECISION can also explain Situated-SemaFORR's preference for action a over an alternative a' with a procedure similar to Algorithm 7. If tier 1 chose a, the explanation uses VICTORY or ENFORCER's rationale: "I decided not to [action a'] because I detect our [target/waypoint] and another action would get us closer to it." If AVOIDWALLS or NOTOPPOSITE vetoed a', then the natural explanation is "I decided not to [action a'] because [rationale from Advisor that vetoed it]."

The other possibility is that a' had lower total strength in tier 3 than a did. In this case, WHY-DECISION generates a natural explanation with the tier-3 Advisors that, by their comment strengths, discriminated most between the two actions. WHY-DECISION calculates $\rho_{ia} - \rho_{ia'}$ for each Advisor i. If the result lies in [-1, 1] then i's support is similar for a and a'; otherwise Advisor i displays a *clear preference*. The natural explanation includes only those Advisors with clear preferences.

The explanation template is "I thought about [action a'] (because it would let us [rationales from Advisors that prefer action a']), but I felt [phrase] strongly about [action a] since it lets us [rationales from Advisors that prefer action a]." The [phrase] is the extent to which Situated-SemaFORR prefers a to a'. It is selected based on $\zeta_a - \zeta_{a'}$, the difference in the actions' overall support, and mapped into intervals as in Table 14. The portion in parentheses is only included if any Advisors showed a clear preference for action a'. For "Why didn't you take action a_2?" on the running example, WHY-DECISION calculates the difference in overall support between a_4 and a_2 at 0.38, which maps in Table 14 to "slightly more." The differences in relative support between a_4 and a_2 for the four Advisors are 1.92, 0.44, 0.45, and -2.22. Thus, if Advisor 1 is GREEDY and prefers a_4, while Advisor 4 is EXPLORER and prefers a_2, the natural explanation is "I thought about a_2 because it would let us go somewhere new, but I felt slightly more strongly about a_4 since it lets us get closer to our target."

In summary, WHY-DECISION produces natural explanations for a robot's navigation decisions as it travels through a complex environment. These explanations are essential for collaborative navigation and are made possible by the robot controller's cognitively-based reasoning. The approach presented here generates explanations that gauge the robot's confidence and give reasons to take an action or to prefer one action over another. As a result, a human companion receives informative, user-friendly explanations from a robot as they travel together. WHY-DECISION will be extended to include the new tier-3 Advisors introduced in Section 3.3. This requires development of human-friendly descriptions for each tier-3 Advisor's rationale. WHY-DECISION will also require modification for different voting methods to select an action in tier-3, since the current approach assumes the use of sum range voting. Finally, WHY-DECISION will be able to ignore any weights that are applied to tier-3 Advisors because the weights $\alpha_{hi} \in [0, 1]$ do not change the range of the Advisors' comment strengths $c_{ia} \in [0, 10]$. The next subsection describes WHY-PLAN, which compares the perspectives of an autonomous robot and a person to produce meaningful, human-friendly explanations of Situated-SemaFORR's plans.

4.1.2 Why-Plan

Why-Plan exploits differences among robot controllers' objectives to produce clear, concise natural explanations for a plan quickly. Why-Plan addresses the question "Why does your plan go this way?" Given a plan P, Situated-SemaFORR's Advisors determine how to travel from one waypoint to the next. Why-Plan addresses P to explain, in natural language, the robot's long-range perspective. The initial version of Why-Plan was reported in Korpan and Epstein (2018). This subsection describes that implementation and then proposes extensions and improvements.

Human and robot plans to arrive at the same goal may differ because they do not capitalize on the same objective. The premise of Why-Plan is that a human plans from one perspective, objective $\mathcal{O}_M$, while the robot plans from another perspective, objective $\mathcal{O}_R$. As a running example, $\mathcal{O}_M$ is "take the shortest path" and $\mathcal{O}_R$ is the objective of CSA* (described in Section 3.3).

A question about the robot's plan could arise anywhere along its intended path. Algorithm 8 is pseudocode for Why-Plan's explanation procedure. Recall that a robot's decision state d includes its pose, and the obstacles its sensors currently detect. Why-Plan takes as input the target location t, the robot's current decision state d, its planning objective $\mathcal{O}_R$, and an alternative planning objective $\mathcal{O}_M$ attributed to the human questioner. It is assumed that the robot and the person share knowledge relevant to $\mathcal{O}_R$ and $\mathcal{O}_M$ (e.g., distances and crowding). This allows the robot to construct two plans: its own plan P_R based on $\mathcal{O}_R$, and P_M, the robot's approximation of the human's implicit plan, based on $\mathcal{O}_M$.

To compare P_R with P_M, Why-Plan applies both $\mathcal{O}_R$ and $\mathcal{O}_M$ as metrics. Let $M(P)$ measure $\mathcal{O}_M$ for a plan (e.g., its length if $\mathcal{O}_M$ is shortest path), and let $R(P)$ measure a plan under $\mathcal{O}_R$ (e.g., crowd sensitivity). Why-Plan translates its objec-

```
Algorithm 8: Why-Plan's explanation procedure
Input: decision state, target location, planning objectives $\mathcal{O}_R$ and $\mathcal{O}_M$
Output: explanation
Compute plan $P_R$ based on $\mathcal{O}_R$
Compute plan $P_M$ based on $\mathcal{O}_M$
$\Delta_R = R(P_R) - R(P_M)$
$\Delta_M = M(P_R) - M(P_M)$
switch mode($\Delta_R$, $\Delta_M$) do
    case $\Delta_R = \Delta_M = 0$ do
        explanation $\leftarrow$ sentence based on template for equivalent plans
    case $\Delta_R < 0$ and $\Delta_M = 0$ do
        explanation $\leftarrow$ sentence based on $\mathcal{O}_R$ (e.g., crowd density)
    case $\Delta_R < 0$ and $\Delta_M > 0$ do
        explanation $\leftarrow$ sentence based on $\mathcal{O}_R$ and $\mathcal{O}_M$ (e.g., crowd density and
         length)
end
return explanation
```

tives into natural language with a descriptor: κ_M for $\mathcal{O}_M$ and κ_R for $\mathcal{O}_R$. In the example, κ_M is "short" and κ_R is "crowded." Each objective $\mathcal{O}$ is also associated with user-specified numeric intervals that assign a descriptive natural language phrase *nl* to a difference Δ between two plans' measures under $\mathcal{O}$, as in Table 15. These ranges represent a human perspective on the objective.

Why-Plan calculates the difference in the plans from both perspectives. $\Delta_M = M(P_R) - M(P_M)$ is their difference from the human's perspective (e.g., length), and $\Delta_R = R(P_R) - R(P_M)$ is their difference from the robot's perspective (e.g., crowded-

	$(-\infty, -1500]$	a lot
nl_R for $\mathcal{O}_R$ = crowd-sensitive	$(-1500, -500]$	somewhat
	$(-500, 0)$	a bit
	$(0, 1]$	a bit
nl_M for $\mathcal{O}_M$ = short distance	$(1, 10]$	somewhat
	$(10, +\infty)$	a lot

Table 15: Mappings from value intervals to language

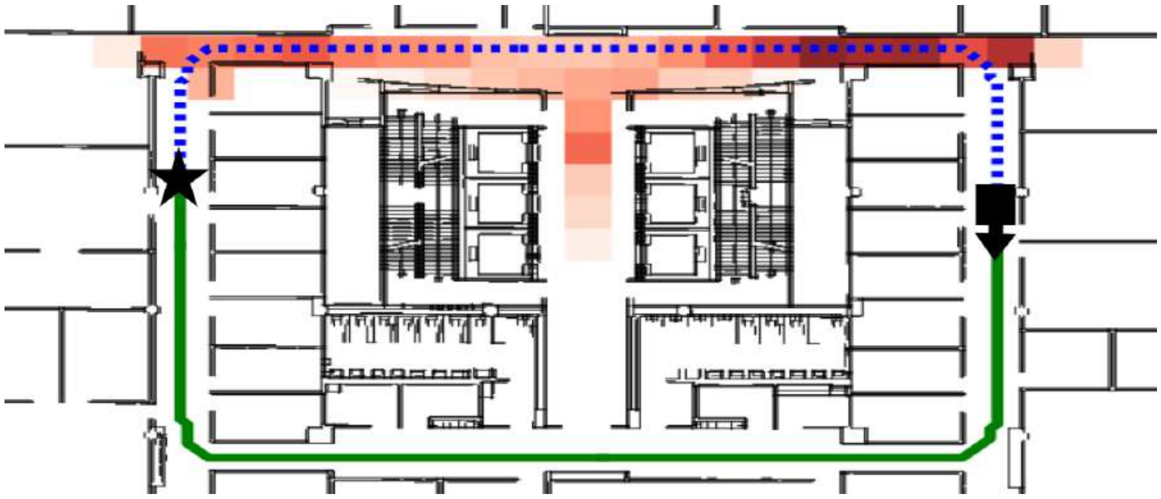


Figure 11: Why-Plan answers, “Although there may be a somewhat shorter way, I think my way is a lot less crowded.” (Korpan and Epstein, 2018, p. 1)

ness). There are several possible cases, each with its own language template. If both Δ_M and Δ_R are 0, then the plans equally address the two objectives, and Why-Plan explains “I decided to go this way because I think it’s just as κ_M and equally κ_R.” Otherwise, the plans differ with respect to one or both objectives. If Δ_R is negative (e.g., P_R is more crowd-sensitive), then Why-Plan uses the template “(Although there may be a $\langle nl_M \rangle$ κ'_M way,) I think my way is $\langle nl_R \rangle$ κ'_R” where κ'_M and κ'_R are comparators for κ_M and κ_R, respectively (e.g., “shorter” and “less crowded”). Why-Plan omits the parenthetical statement if $\Delta_M = 0$. Other cases, where $\Delta_M < 0$ or $\Delta_R > 0$, cannot occur under the shared knowledge assumption.

Figure 11 illustrates an example for Why-Plan, where the robot (black box) faced in the direction of the arrow, its goal was the star, darker shading indicates more crowding, and $\Delta_R = -6729$ and $\Delta_M = 9.6$. Here, P_R (solid line) goes down to avoid the crowd, while P_M (dotted line) takes a shorter path through the crowd. In response to “Why does your plan go this way?” the robot explained, “Although there may be a somewhat shorter way, I think my way is a lot less crowded.”

WHY-PLAN will be expanded to answer other important questions not yet addressed, such as "Why doesn't your plan go that way?", "What makes your plan better than mine?", "What's another way we could go?", and "How sure are you about your plan?" Each question will be answered based on comparison of the objectives. To respond to "Why doesn't your plan go that way?", WHY-PLAN will compute a new plan P'_R based on $\mathcal{O}_R$ that is forced to adhere to the direction indicated by the human collaborator. Next, it will compute $\Delta_R = R(P_R) - R(P'_R)$ and $\Delta_M = M(P_R) - M(P'_R)$. WHY-PLAN will then respond based on these two measures with the template "(Although that way may be a $\langle nl_M \rangle$ κ'_M,) I think my way is $\langle nl_R \rangle$ κ'_R." WHY-PLAN will omit the parenthetical statement if $\Delta_M \leq 0$.

WHY-PLAN will also address the question "What makes your plan better than mine?" First, WHY-PLAN will compute $\Delta_M = M(P_R) - M(P_M)$ and $\Delta_R = R(P_R) - R(P_M)$. If both Δ_M and Δ_R are 0, then WHY-PLAN will explain "I think both plans are equally good." Otherwise, WHY-PLAN will respond with the template "I think my way is better because it's $\langle nl_R \rangle$ κ'_R."

The third question is "What's another way we could go?" WHY-PLAN's response would require a more nuanced approach. There are two possible responses, either provide P_M as the alternative plan or produce a new plan P'_R that is sufficiently different from P_R. In the first case, WHY-PLAN will respond "We could go that way since it's $\langle nl_M \rangle$ κ'_M but it could also be $\langle nl_R \rangle$ κ''_R" where κ''_R is an opposite comparator for κ_R (e.g., "more crowded"). If both Δ_M and Δ_R are 0, then WHY-PLAN will respond based on a new plan P'_R. WHY-PLAN will compute P'_R based on $\mathcal{O}_R$ with the restriction that P'_R's similarity to P_R is not above some threshold, where similarity is measured by the percentage of their common waypoints. WHY-PLAN will then compute $\Delta_R = R(P_R) - R(P'_R)$ and $\Delta_M = M(P_R) - M(P'_R)$, and use those

		$\Delta_M =$ (10, +∞)	$\Delta_M =$ (1, 10]	$\Delta_M =$ (0, 1]
$\Delta_R =$	$(-\infty, -1500]$	only somewhat	really	really
	$(-1500, -500]$	not	only somewhat	really
	$(-500, 0)$	not	not	only somewhat

Table 16: Mapping from Δ_R and Δ_M intervals to confidence level phrases

values to respond with the template "We could go that way but it's $\langle nl_M \rangle$ κ''_M and $\langle nl_R \rangle$ κ''_R" where where κ''_M is an opposite comparator for κ_M (e.g., "longer").

WHY-PLAN will also address a fourth question: "How sure are you about your plan?" To respond, WHY-PLAN will analyze and explain its confidence in its objective. It first compares the magnitude of Δ_R with that of Δ_M and then uses that to decide its confidence in the plan P_R. Table 15 partitions the values of Δ_R and Δ_M into three intervals. WHY-PLAN will explain the robot's confidence by where in these intervals the differences Δ_R and Δ_M lie. The combination will produce a phrase in natural language as shown in Table 16. All responses to this question will use a template that begins "I'm [confidence level phrase] sure because...." If the confidence level phrase is "really", the template will continue "my plan is $\langle nl_R \rangle$ κ'_R and only $\langle nl_M \rangle$ κ''_M than your plan." If the confidence level phrase is "only somewhat", the template will continue "even though my plan is $\langle nl_R \rangle$ κ'_R, it is also $\langle nl_M \rangle$ κ''_M than your plan." Finally, if the confidence level phrase is "not", the template will continue "my plan is $\langle nl_M \rangle$ κ''_M than your plan and only $\langle nl_R \rangle$ κ'_R."

In summary, WHY-PLAN will produce natural explanations for a robot's plan as it travels through a complex environment. These explanations are essential in collaborative navigation and are made possible through consideration of its collaborator's objective. The approach presented here will generate explanations for the robot's plan, respond to questions about alternative plans, and provide a confidence level for its plan. WHY-PLAN does not show a visual representation of the plan before it responds with an explanation because it would likely cause cognitive overload (e.g., how does the map align to the person's perspective of the environment) and require additional explanations (e.g., for all the symbols and labels on the image).

WHY-PLAN will be extended to include the new tier-2 planners introduced in Section 3.3. This includes development of human-friendly descriptions for each planner's objective. WHY-PLAN will also require modification for the proposed multi-objective planning, since the current approach assumes the use of a single planner. One possibility is to assign the robot's objective in WHY-PLAN to be the objective for the planner that was selected by sum range voting, and make the human's objective the one the selected planner performed worst on. Alternatively, WHY-PLAN could combine all the objectives in a single explanation (e.g. "Although there may be a somewhat shorter and less risky way, I think my way is somewhat less crowded and a lot more smooth"). The next subsection proposes WHY, which combines WHY-DECISION and WHY-PLAN to produce meaningful, human-friendly explanations of Situated-SemaFORR's situations and situated-decision strategies.

4.1.3 WHY

WHY will combine the explanations of Situated-SemaFORR's reactivity and deliberation into a single coherent narrative that includes an explanation of the robot's current situation and how its situated-decision strategy for that situation influenced its behavior. WHY is not intended to be a general dialogue system, rather, it will provide natural explanations for the system's hybrid, hierarchical decision making. This subsection first discusses ways in which WHY-DECISION and WHY-PLAN could be combined. It then describes how WHY's explanations will address Situated-SemaFORR's situations and situated-decision strategies.

WHY will generally address questions from a human collaborator about Situated-SemaFORR's reasoning. Although WHY will continuously generate explanations, it will only provide an explanation when asked. One way to combine WHY-DECISION and WHY-PLAN is to concatenate their responses to address both reactivity and deliberation. In this approach, WHY combines WHY-DECISION's response to "Why did you decide to do that?" with WHY-PLAN's response to "Why does your plan go this way?" to respond to "Why did you do that?" with the template "Generally, [WHY-PLAN's response]. Specifically, right now, [WHY-DECISION's response]." For example, WHY's response with this template could be "Generally, although there may be a somewhat shorter way, I think my way is a lot less crowded. Specifically, right now, although I don't want to go somewhere I've been, I decided to move forward a lot because I want to get close to our waypoint." This response would provide both the long-range perspective and explain the immediate reactive decision.

In this first approach, WHY will similarly respond to the question "How sure are you about what you are doing?" with a template that combines the responses of the two components. The response will depend on the agreement between the confidence level phrase from WHY-DECISION and WHY-PLAN. If the phrase is the same (i.e., both are "really") then the template is "[WHY-DECISION's response]. Also, [WHY-PLAN's response]." If the phrases differ then the response will acknowledge that difference based on which has higher confidence (with the order defined as "really" > "only somewhat" > "not"). The template in that case is "Although [response from component with higher confidence], [response from component with lower confidence]."

WHY will also respond to other questions with the appropriate response from either WHY-DECISION or WHY-PLAN. It will not combine the responses because the questions are specifically related to the robot's reactive decision or its plan, not both. For example, WHY-DECISION would answer "Why didn't you turn left?" and it would not be appropriate to also include WHY-PLAN's response to "Why doesn't your plan go that way?"

A second way to combine WHY-DECISION and WHY-PLAN is based on the expectations of a human companion. In this approach, WHY would provide the explanation from WHY-PLAN by default, and only provide an explanation from WHY-DECISION if the person asks a follow-up question. This approach assumes that people are more concerned with the robot's long-range plan because people may not be able to recognize individual actions or distinguish when one finishes and the next one starts. Additionally, people may only ask about an individual decision when it clearly violates their expectations or knowledge. Since this proposal has argued that Situated-SemaFORR's cognitive basis will enable a robot to move naturally and sociably, such violations should happen infrequently, and so explanations for individual decisions should rarely be needed.

A third way to construct WHY exploits the mechanism by which a human companion requests an explanation. WHY could, for example, display the questions and have the person select one. In that case, WHY would not combine explanations; it could just produce the appropriate response to the selected question. In this case, the phrasing of the questions may need to be revised based on human feedback.

The above approaches would allow WHY to explain both reactive decisions and a plan with one response. That approach will be used if the robot has not yet learned situated-decision strategies. WHY will also explain a learned situation and its associated situated-decision strategy in natural language. It will respond to "Why did you do that?" with the template "When [situation U_h], [situated-decision strategy χ_h]." A situation will be explained with the features that distinguish it from other

	e_1	e_2	e_3	$(e_1 - e_2)$	$(e_1 - e_3)$	average difference
f_1	2.7	9.1	1.9	-6.4	0.8	-2.80
f_2	1.3	6.9	8.7	-5.6	-7.4	-6.50
f_3	5.6	1.6	1.5	4.0	4.1	4.05
f_4	8.6	0.4	3.5	8.2	5.1	6.65

Table 17: An example that distinguishes the situation with exemplar e_1 from two other situations

situations. To explain a situated-decision strategy, Why will provide an explanation of the weight learning algorithm or voting method that was used for that situation.

The features with values most different from the other situations are used to explain a situation. First, the exemplar e_h for the situation U_h will be compared to the exemplars for all other situations. For each feature f_j, the difference between the feature's value in e_h and the values for the other exemplars will be computed, and the differences for each feature averaged. Those features with the largest average differences will then be used to explain the situation. Table 17 provides a running example, with four features (f_1, f_2, f_3, f_4), their values for three exemplars (e_1, e_2, e_3), and the feature-value differences among e_1, e_2, and e_3. The average difference in the feature values shows that e_1 differs from the other exemplars most along the features f_2 and f_4.

A situation-based explanation will identify the features that most distinguish the exemplar, map their differences to descriptors for the magnitude of the difference, and combine them with a natural-language description of each feature. For example, in Table 17 if f_2 is distance to a high frequency conveyor and f_4 is distance to the nearest obstacle, then the explanation in the situation with exemplar e_1 would be "We're really close to a familiar place and really far from our nearest obstacle." As in my earlier work, the natural language descriptions for features and mappings for their differences will be selected empirically. The number of features to include and the similarity of the selected features would also need to be considered.

The explanation for a situated-decision strategy will depend upon the method used to learn it. If weight learning were used, then WHY will explain based on the Advisors whose weights have the highest values. For example, if GREEDY and TRAILER had the largest weights, then the explanation would be "It's best to move closer toward our target and follow a way we've gone before." If a voting strategy were used, WHY would identify the Advisors that most contributed to the final tally (i.e., those which were dictatorial) and use them.

Finally, WHY will combine the explanations for the situation and situated-decision strategy. In the running example this could be "When we're really close to a familiar place and really far from our nearest obstacle, it's best to move closer toward our target and follow a way we've gone before." WHY will potentially provide many varied and nuanced explanations for Situated-SemaFORR. Such explanations, however, will require consideration of length, naturalness, and understandability based on human evaluation. Descriptions for geometric situations could also be similarly constructed or a human-imposed label could be applied.

WHY could also incorporate more varied language in its descriptions and templates. One way would be to use methods from natural language generation that probabilistically vary text so that it is more natural and less repetitive (Gatt and Krahmer, 2018). This approach would forgo language templates and instead generate the full text responses. WHY could also use a natural explanation for a learned condition C as the basis for explanations for the condition's member situations.

In summary, WHY will combine WHY-DECISION and WHY-PLAN to respond more generally to questions about Situated-SemaFORR's reasoning process. It will also explain Situated-SemaFORR's learned situations and their situated-decision strategies. The next section reviews preliminary results on Why-Decision and Why-Plan.

4.1.4 Preliminary work

This subsection reviews preliminary simulation results from Why-Decision and Why-Plan. The many distinct natural explanations simulate people's ability to vary their explanations based on their context (Malle, 1999). How to incorporate Why into the architecture is also addressed.

Why-Decision is implemented as a ROS package and explains navigation decisions in real time. It was evaluated in a MengeROS simulation for a real-world robot (Fetch Robotics' Freight). Why-Decision averaged less than 3 msec per explanation when the robot navigated to 230 destinations in the complex 60m×90m office environment of Figure 12. Initial results show that this approach is efficient and nuanced. Table 18 provides further details. The Coleman-Liau index measures text readability; it gauged Why-Decision's explanations over all three questions at approximately a 6th-grade level (Coleman and Liau, 1975), and thus readily understandable to a layperson.

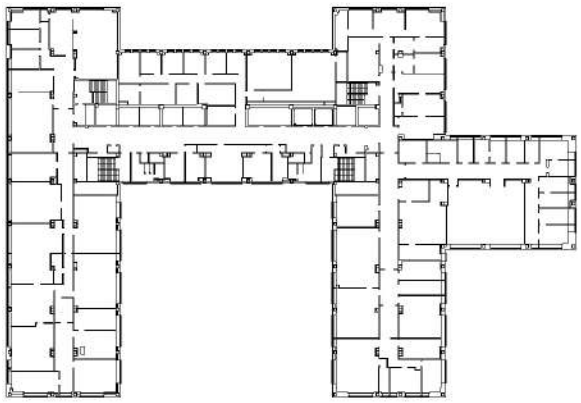


Figure 12: Tenth floor of Hunter College's North Building

Tier where decision was made	1	3	All
Number of decisions	22,982	84,920	107,902
Average computation time (ms)	0.45	3.08	2.52
Number of unique phrasings			
Why did you decide to do that?	14	31,896	31,910
How sure are you?	2	11	13
Why not do something else?	19	124,086	124,105
Total	35	155,993	156,028
Average readability			
Why did you decide to do that?	8.18	5.02	5.70
How sure are you?	10.39	7.63	8.22
Why not do something else?	3.91	6.44	5.96
Overall	5.36	6.41	6.21

Table 18: Preliminary results for Why-Decision in simulation

For action a chosen in tier 3 and every possible alternative a', Table 19 shows how often the values of γ_a, ζ_a, Λ_a, $\rho_{ia} - \rho_{ia'}$, and $\zeta_a - \zeta_{a'}$ fell in their respective Table 14 intervals. The Advisors disagreed ($\gamma_a > 0.45$) on 67.15% of decisions. Strong overall support ($\zeta_a > 1.5$) made SemaFORR strongly confident in 2.44% of its decisions ($\Lambda_a > 0.375$) and somewhat confident in 42.64% of them. When asked about an alternative, individual Advisors clearly preferred ($\rho_{ia} - \rho_{ia'} > 1$) the original decision 39.50% of the time; SemaFORR itself declared a strong preference ($\zeta_a - \zeta_{a'} > 1.5$) between the two actions 61.13% of the time.

Table 20 illustrates Why-Decision's robust ability to provide nuanced explanations for tier-3 decisions. The target appears as an asterisk and the black box and

	Low	Medium	High
Level of agreement γ_a	67.15%	30.41%	2.44%
Overall support ζ_a	2.34%	60.09%	37.57%
Confidence level Λ_a	54.92%	42.64%	2.44%
Difference in relative support $\rho_{ia} - \rho_{ia'}$	16.09%	44.41%	39.50%
Difference in overall support $\zeta_a - \zeta_{a'}$	18.48%	20.40%	61.13%

Table 19: Why-Decision's metric distributions by interval in tier-3 decisions

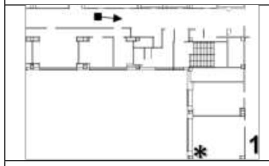
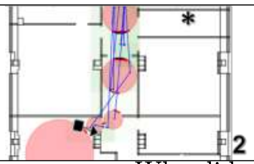
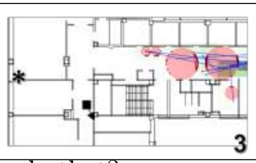
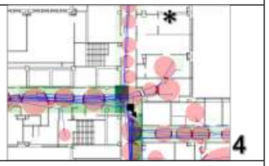
Robot's State			
			
Why did you do that?			
Although I don't want to go close to that wall, I decided to bear right because I really want to take a big step.	Although I don't want to turn towards this wall, I decided to turn right because I want to go somewhere familiar, I want to get close to our target, and I want to follow a familiar route that gets me closer to our target.	Although I really don't want to go close to that wall and I really don't want to get farther from our target, I decided to move forward a lot because I really want to go to an area I've been to a lot, I really want to take a big step, and I really want to go somewhere new.	Although I don't want to get farther from our target, I decided to bear left because I really want to go somewhere familiar and I want to leave since our target isn't here.
How sure are you?			
I'm not sure because my reasons conflict.	I'm only somewhat sure because, even though my reasons conflict, I really want to do this most.	I'm not sure because my reasons conflict.	I'm only somewhat sure in my decision because I've only got a few reasons. I somewhat want to do this most.
Why not do something else?			
I thought about turning left because it would let us stay away from that wall and get close to our target, but I felt more strongly about bearing right since it lets us take a big step and get around this wall.	I thought about shifting left a bit because it would let us get around this wall, but I felt much more strongly about turning right since it lets us go somewhere familiar and get close to our target.	I decided not to move far forward because the wall was in the way.	I thought about turning hard right because it would let us get close to our target, but I felt much more strongly about bearing left since it lets us go somewhere familiar, leave since our target isn't here, go somewhere new, and get around this wall.

Table 20: Explanations for the robot's states and any current spatial model, enlarged from Figure 12

arrow show the robot's pose. Decision 1 was made when the robot had not yet learned any spatial affordances; decision 2 was made later, when the spatial model was more mature. In decision 3, the Advisors strongly disagreed, while in decision 4 the spatial model-based Advisors disagreed with a commonsense-based Advisor.

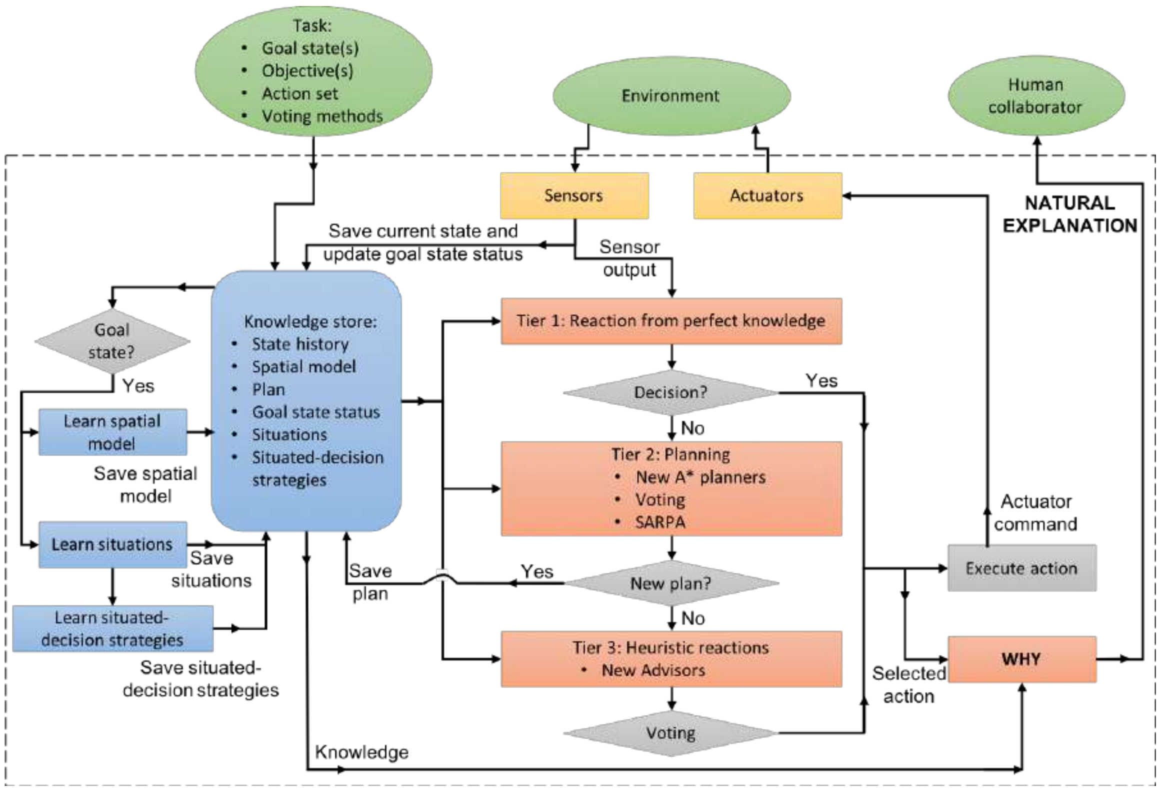


Figure 13: Situated-SemaFORR's architecture with Why. Proposed work introduced in this section is indicated in uppercase.

Why-Plan was also implemented as a package in ROS and evaluated in simulation with MengeROS for a real-world robot (Fetch Robotics' Freight). The robot navigated to 40 destinations in a complex, 60m×90m office world (The Graduate Center's Fourth Floor). Why-Plan produced explanations in real time ($\mu = 4.9$ ms, $\sigma = 2.5$ ms, $n = 3327$) while the robot traveled. Why-Plan's speed allows it to compute an explanation for each state, which it provides only when asked. Further inspection of the simulated results indicates that Why-Plan's other natural explanations for SemaFORR's plans are similarly intelligible and transparent.

Why will be a separate ROS package that analyzes Situated-SemaFORR's outputs to provide explanations seamlessly. Figure 13 shows Situated-SemaFORR's architecture with Why incorporated. After Situated-SemaFORR selects an action, it will send that action and its knowledge store to Why to produce a natural explanation. The next section discusses how Situated-SemaFORR will be evaluated.

4.2 Evaluation

Situated-SemaFORR will be evaluated with respect to two criteria: navigation performance and impact on people. Navigation performance will be measured by computation time, travel time, and path length. Impact on people will be evaluated by human subjects on several factors, including comfort, trust, understandability, and interpretability. This section reviews potential approaches for both criteria. An existing IRB #2015-0933 titled "Learning to perform consistently in human/multi-robot teams" will be amended to include the human-subject studies discussed here.

4.2.1 Navigation performance

A significant issue in collaborative navigation is that it lacks a standard testbed and standard performance metrics. This makes it difficult to compare results from different systems, especially when they make different assumptions about the environment. While path cost, computation time, and algorithmic complexity are appropriate performance metrics for path planning, no one metric is consistently used in the robot navigation literature, and none of them is adequate to evaluate performance during collaborative navigation. Without consistent and rigorous comparisons, however, it is difficult to determine which approach is superior.

To contend with these issues, the navigation performance of Situated-SemaFORR will be measured with several criteria and will be compared against ablated versions of the system. Computation time will measure the average time it takes to select an action and produce a plan. Since Why will be implemented as a separate ROS package, it will be run in parallel with Situated-SemaFORR and its computation time will be evaluated separately. Travel time will be measured as the total wall-clock time required for the robot to travel from an initial state to its target. Path length will be measured by the total distance travelled by the robot. Other possible metrics could measure how well Situated-SemaFORR does on the robot's planning objectives.

Navigation performance will be measured in simulated experiments with MengeROS on a variety of unique and challenging real-world environments. Each experiment will include multiple runs, each with multiple tasks. The results will be evaluated with a statistical analysis to look for significant differences among the ablated versions. Since Situated-SemaFORR can be run alongside another robot controller, computation time will need to be fast enough to ensure a decision cycle and learning do not lag behind the other robot controller. Alternatively, Situated-SemaFORR could do its learning in batches. Both cases will be evaluated.

4.2.2 Impact on people

Situated-SemaFORR will also be evaluated with human-subject studies to assess its impact on people. There are two possible ways to do so. One is to run a simulation experiment. In a within-subjects design, a subject would be shown two robot navigators, one controlled by Situated-SemaFORR and another controlled by an ablated version. The subject would then be asked about their preference between the two approaches. A between-subjects design would split subjects into groups and show each group a different condition. Subjects' perceptions of comfort, predictability, safety, friendliness, naturalness, and sociability of the robot would be measured on a Likert scale. Statistical analysis would then test for significant difference in subjects' ratings. Subjects could also be asked to describe their interpretation of the robot's decision-making rationales in natural language as another way to evaluate how accurately they can identify Situated-SemaFORR's underlying methods. Subjects will be recruited from an online crowd-sourcing platform, such as Amazon Mechanical Turk. The number of subjects will be determined with a power analysis.

A second human-subject approach would have the subject follow the robot as it navigates in a controlled environment to a shared destination. After that trip, a subject would be evaluated with the same metrics used in the simulated experiment. The conditions that a subject would experience would vary, based on the robot's controller (Situated-SemaFORR or an ablated version of it) and whether a within- or between-subjects design is used. Subjects would be recruited with a stratified random sampling procedure from the Hunter and CUNY communities. Stratified random sampling is used because it allows researchers to ensure equal distribution of demographics among individuals in each treatment group. Advertising on social media, in classes, and with flyers could be used to reach a wide audience. Participants could be compensated for their time to incentivize participation in the study.

Subjects for the in-person study would also be evaluated with the Santa Barbara Sense of Direction (SBSOD) questionnaire, a self-report measure of individual's spatial abilities (Hegarty et al., 2002). This measure would help to determine if a subject's spatial ability has an impact on her perception of the robot's navigation behavior. Subjects could also be asked to take the Negative Attitudes Towards Robots Scale (NATRS), a standardized questionnaire that has been used in the literature to evaluate people's perceptions of robots (Nomura et al., 2008), since a subject's negative perceptions of robots may influence her comfort with the robot during the experiment.

A subject in the physical interaction study will follow a strict experimental procedure. When a subject first arrives, she will complete a consent form and read instructions that outline the procedure for the experiment. Next, she will complete a pre-treatment questionnaire to establish her baseline navigation skill levels with SB-

SOD and perception of robots with NATRS. Afterward, she will be led blindfolded to the location of the experiment to prevent her from learning anything about the environment before she begins the treatment condition. Then, the blindfold will be removed and the subject will be asked to travel to three rooms identified by number in the indoor Hunter College environment. She will receive one room number at a time and will travel to the next one after arrival at the currently assigned room. These initial three locations will be without the presence of the robot to get a baseline measure of navigation performance. A time cutoff will be used to limit the length of the experiment in case the subject is unable to find a room in a reasonable amount of time.

After this baseline period, each subject will follow the robot as it navigates to 10 locations but now under a randomly assigned treatment condition (some version of Situated-SemaFORR). Ten locations will be used to control for any outliers (e.g., if one task takes much longer than the rest). Finally, after the 10 locations are visited, the subject will complete a post-treatment questionnaire to evaluate attitudes toward the robot, and perceptions of the robot's behavior. A random subset of the subjects will be asked to participate in an exit interview to get a more in-depth understanding of their perceptions, opinions, and navigation strategies.

Neither human-subject study described above would provide explanations of the robot's behavior; they merely evaluate the robot's behavior. Separate human-subject experiments will evaluate WHY. First a preliminary study will refine WHY's natural explanations. Because SemaFORR's spatial model is approximate and its Advisors are heuristic, precise natural language interpretations for numeric values are ad hoc. For Tables 14, 15, and 16, thousands of decisions were inspected, and then their computed values were partitioned as appeared appropriate. Both intervals and phrasing will be fine-tuned with empirical assessment by native English speakers through an online crowd-sourcing platform. Variations of WHY's responses, based on different templates and intervals, will be presented to subjects, and their preferences evaluated. Each subject will indicate how natural, readable, understandable, and human-like each explanation is. These measures will then be used to determine WHY's parameters.

After this initial study, a follow-up study will evaluate WHY's impact on people's comfort with and understanding of the robot. Subjects, recruited through an online crowd-sourcing platform, will watch the robot autonomously navigate to 5 target locations in a small, office-like virtual environment. After decisions made by the robot, the subject will be asked to what extent she understands the robot's action, and whether she wants an explanation for it. If so, she will receive explanations from WHY, and will be asked about the understandability, naturalness, and level of detail of the explanation. There will also be an exit survey on demographics, preferences for explanations, and general attitude toward robots. The results of the study would be used to determine when people would like an explanation from the robot as it navigates and how its explanations affect comfort and trust in the robot.

This section discussed methods to evaluate Situated-SemaFORR's navigation performance and its impact on people. Navigation performance will be measured with simulation experiments against ablated version of the system. Human subject studies will measure how Situated-SemaFORR affects people's comfort with and trust in the robot, as well as their understanding of the robot's behavior. The next chapter summarizes the potential contributions from Situated-SemaFORR.

5. Conclusion

Collaborative navigation is an important challenge in human-robot interaction, one that must be addressed as robots are increasingly integrated into human society. Robot navigators that travel alongside a person will need to achieve their own objectives and also ensure the comfort of their companion. The thesis of this proposal is that a robot navigator will do both when it incorporates situated cognition and meaningful semantics. Situated-SemaFORR addresses this thesis; it will learn situations and situated-decision strategies, and then explain its reasoning in natural language. This chapter summarizes the potential contributions and broader implications of the proposed work.

Chapter 3 introduced two major components of Situated-SemaFORR: situations and situated-decision strategies. Situations group a robot's experiences into potentially useful clusters. Each learned situation has an exemplar, a paradigmatic representation. Once a situation is learned, a situated-decision strategy identifies the best action or plan to take in that situation. These situated-decision strategies will incorporate new tier-3 Advisors that draw from the related work in Chapter 2, new A*-based planners, and a new tier-2 voting mechanism. Chapter 3 also proposed a novel metaheuristic, SARPA, as an alternative to A* search.

The potential contributions from the proposed work in Chapter 3 include a new approach to generalize a robot's experiences with situations and a novel way to specialize a robot's behavior to its circumstances with situated-decision strategies. This approach is potentially applicable much more broadly, to any real-world task with a robot. It would allow a robot to learn strategies that apply to different parts of its problem space. Another contribution is the use of a voting method for multi-objective path planning. This approach is more feasible than other multi-objective planners because it does not contend with multi-objective optimization. Another major contribution could be SARPA, a metaheuristic that draws from human behavior to search a space. This cognitively-based metaheuristic may support clearer explanation of a search algorithm's methodology to a human companion.

Chapter 4 discussed approaches to explain Situated-SemaFORR's decision-making rationales in human-friendly natural language. It uses WHY-DECISION to explain reactive navigation decisions and WHY-PLAN to explain navigation plans. These two methods will be combined as WHY to provide comprehensive explanations, including situations and their associated situated-decision strategies.

WHY is potentially applicable more broadly than indicated thus far. Any robot controller could have Situated-SemaFORR learn the spatial model (described and extended in Chapter 3) in parallel, and use it with WHY to produce transparent, cognitively-plausible explanations. If the alternative controller were to select action a' when Situated-SemaFORR selected a, WHY could still explain a' with any Advisors that supported it, and offer an explanation for a as well. Furthermore, once equipped with Advisor phrases and possibly with new interval mappings, any FORR-based system could use WHY to produce explanations. For example, Hoyle is a FORR-based system that learns to play many two-person finite-board games expertly (Epstein, 2001). For Hoyle, WHY could explain "Although I don't want to make a move that once led to a loss, I decided to do it because I really want to get closer to winning and I want to do something I've seen an expert do."

WHY could be incorporated into a more general dialogue system that would facilitate part of a broader conversation between a human collaborator and a robot. A preliminary FORR-based system for human-computer dialogue, could prove helpful here (Epstein et al., 2011). WHY presumes that questions arise from a difference between the human's and the robot's objectives, but they could also stem from a violation of the shared knowledge assumption between the robot and the person. A broader system for human-robot collaboration would seek the cause of such a mismatch, use explanations to resolve it, and then adjust the robot's responses based on feedback from its human partner. WHY could also use known planning objectives to detect a more complex human objective. For example, given a plan P from a person or an unknown heuristic planner, WHY could use the individual objectives in its repertoire to tease apart and then characterize how P weighted its objectives (e.g., "You appear to consider distance twice as important as travel time."). WHY could also address violations of assumptions made during planning (e.g., now-blocked passageways or currently empty areas that are typically crowded).

Chapter 4 also described methods to evaluate Situated-SemaFORR. It proposed simulation experiments to measure improvements in navigation performance, and discussed two potential human-subject studies. One study uses a simulated environment to measure possible improvements in the naturalness of the robot's behavior and people's comfort with the robot. The other examines people's perceptions and reactions in a real-world physical interaction with a robot navigator. These experiments would validate Situated-SemaFORR's proposed contributions and would provide a basis for further research.

In conclusion, this paper has proposed Situated-SemaFORR, a cognitively-based, explainable, and robust system for collaborative navigation. This work draws from artificial intelligence, robotics, human-robot interaction, cognitive science, and machine learning to address the challenges of collaborative navigation in a way that ensures people's comfort and allows the robot to achieve its goals. The potential contributions will have broad implications for both HRI and robotics, and are expected to generate further interest in collaborative navigation.

Glossary of Notation

Definitions

$\mathcal{P}$	A robot navigation problem where $\mathcal{P} = \langle S, I, A, G \rangle$
A	A set of possible actions from which the robot can select
a	An action $a \in A$ is intended to change the robot's pose
$G(s)$	A Boolean goal test that returns true if the robot's current location is a target
H	The search space for a navigation problem $\mathcal{P}$ is the set of all paths that start at an initial state
I	A set of initial states, $I \subseteq S$
o	An optimal solution is a solution in the search space with minimum path cost: $o = \arg\min_{p \in H} PathCost(p)$
P	A plan is a path that can be proved to be a solution before it is executed
p	A path is a finite ordered sequence of interleaved states and actions
S	A set of states that represent an instance of the environment
s	A state $s = \langle x, y, \theta \rangle \in S$
s_k	A goal state (a target with any pose) where $G(s_k) = True$
t	A target location $t = \langle x, y \rangle$
y	The length of a path p

SemaFORR

ε	The minimum distance from the target for the robot to be considered to have reached the target
ε_{aw}	The distance from an obstacle that AvoidWalls will not allow the robot to move within
c_{ia}	The comment strength of tier-3 Advisor i on action a, $c_{ia} \in [0, 10]$
d	A decision state records the robot's current sensor data and its pose when it makes a decision
i	Tier-3 Advisor $i \in n$

n	The number of tier-3 Advisors

SARPA

$Best$	The complete plan with best quality among all searchers in SARPA
P_A	The complete plan with best quality among all searchers in the hasty team
P_D	The complete plan with best quality among all searchers in the dog team
P_G	The complete plan with best quality among all searchers in the grid team
P_H	The complete plan with best quality among all searchers in the helicopter team

Situations

η	The minimum length of dynamic situation sequences
ι	An observed situated state $\iota \in g$
b	The number of learned situations, $\{U_1, U_2, \dots, U_b\} \subseteq O$
C	A condition groups situations based on their similarity, $C = \{e_1, e_2, \dots, e_q\}$
D	A dynamic situation represents dynamic events and patterns across time as a sequence of exemplars in chronological order, $D = \langle e_1, e_2, \dots, e_l \rangle$
e	An exemplar that represents a situation U
f_j	A feature that is part of a vector representation for a situated state, $f_j \in \langle f_1, f_2, \dots, f_m \rangle$
g	The number of observed situated states in O
h	A situation $h \in b$
j	A feature $j \in m$
l	The number of situations in a dynamic situation
m	The number of features in a vector representation for a situated state
O	A set of observed situated states, $O = \{x_1, x_2, \dots, x_g\}$, where $O \subseteq X$
q	The number of situations in a condition
r	A spatial relation that consists of a list of features and the geometric relation among those features
U	A situation $U \subseteq X$ is a subset of similar situated states

X	A set of situated states that represent an instance of the environment
x	A situated state $x \in X$ includes the robot's pose, sensor data, and additional features, such as a robot's characteristics, its accessible external information, its learned internal knowledge, its tasks and objectives, and its knowledge about its environment
z	The number of spatial relations

Situated-Decision Strategies

α_{hi}	A weight $\alpha_{hi} \in$ [0,1] based on some criterion assigned to a tier-3 Advisor for a situation U_h
$\bar{c_i}$	The mean of Advisor i's comment strengths
β	The "best" action $\beta \in A$ for a situation
χ	A situated-decision strategy selects an action a for a given situation U or a plan P for a given dynamic situation D that advances the robot toward task completion
δ	The maximal distance to the nearest line segment to prevent a line segment from being pruned in the hallway learning algorithm
$\mathcal{O}$	A path planning objective, such as "minimize distance traveled" or "avoid crowds"
ν	The number of sensor endpoints from a rangefinder
ω	The minimal number of nearby line segments to prevent a line segment from being pruned in the hallway learning algorithm
Π	The number of tier-2 objectives
π	An objective for a tier-2 planner $\pi \in \Pi$
Ψ	A set of possible plans from which the robot can select
ρ_{ia}	A tier-3 Advisor's relative support for an action a is $\rho_{ia} = (c_{ia} - \bar{c_i})/\sigma_i$ where $\bar{c_i}$ is the mean of Advisor i's comment strengths and σ_i is their standard deviation
σ_i	The standard deviation of Advisor i's comment strengths
$\Theta_\pi(P)$	The score objective π assigns to plan P
$\varepsilon_{barrier}$	The maximal distance between adjacent line segments for them to be merged in the barrier learning algorithm

ε_{door}	The maximal distance between two exits to create a door or add to an existing door
ε_{slope}	The maximal difference between adjacent line segments' slopes for them to be merged in the barrier learning algorithm
ξ	The number of learned barriers
$f(s)$	The estimated total cost of a partial plan up to node s, $f(s) = g(s) + h(s)$
$g(s)$	The total of the edge costs from the start node to a node s
$h(s)$	A heuristic that estimates the cost from a node s to the target's node t

Why

Δ	The difference between two plan's measures under $\mathcal{O}$
γ_a	The level of agreement among n tier-3 Advisors computed with Gini impurity
κ	A natural language descriptor for an objective
Λ_a	The confidence level that weighs level of agreement γ_a and overall support ζ_a equally
$\mathcal{C}$	The total comment strength $\mathcal{C}$ for an action across all tier-3 Advisors
$\mathcal{O}_M$	The robot's assumption of a human's path planning objective
$\mathcal{O}_R$	The robot's path planning objective
$\mu_\mathcal{C}$	The mean total strength $\mathcal{C}$ of all actions under consideration by tier 3
$\sigma_\mathcal{C}$	The standard deviation of the total strength $\mathcal{C}$ of all actions under consideration by tier 3
ζ_a	The overall support among n tier-3 Advisors computed as a t-statistic across all tier-3 comments
$M(P)$	A function that measures $\mathcal{O}_M$ for a plan P
nl	A descriptive natural language phrase assigned to Δ
P_M	The robot's approximation of the human's implicit plan based on $\mathcal{O}_M$
P_R	The robot's plan based on $\mathcal{O}_R$
$R(P)$	A function that measures $\mathcal{O}_R$ for a plan P

References

Aarts, H. and Elliot, A. (2012). *Goal-directed behavior.* Taylor & Francis.

Abdi, H. and Williams, L. J. (2010). Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459.

Aggarwal, C. C. and Reddy, C. K. (2013). *Data clustering: algorithms and applications.* CRC press.

Ahmed, F. and Deb, K. (2013). Multi-objective optimal path planning using elitist non-dominated sorting genetic algorithms. *Soft Computing*, 17(7):1283–1299.

Aroor, A. and Epstein, S. L. (2017). Toward Crowd-Sensitive Path Planning. In *Proceedings of AAAI Fall Symposium on Human-Agent Groups: Studies, Algorithms and Challenges*, pages 238–245. AAAI.

Aroor, A., Epstein, S. L., and Korpan, R. (2017). MengeROS: A Crowd Simulation Tool for Autonomous Robot Navigation. In *Proceedings of AAAI Fall Symposium on Artificial Intelligence for Human-Robot Interaction*, pages 123–125. AAAI.

Aroor, A., Epstein, S. L., and Korpan, R. (2018). Online learning for crowd-sensitive path planning. In *Proceedings of the 17th Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18. International Foundation for Autonomous Agents and Multiagent Systems.

Arrow, K. J., Sen, A., and Suzumura, K. (2010). *Handbook of social choice and welfare*, volume 2. Elsevier.

Back, T. (1996). *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms.* Oxford University Press, USA.

Barrett, D. P., Bronikowski, S. A., Yu, H., and Siskind, J. M. (2017). Driving Under the Influence (of Language). *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–16.

Bauer, A., Wollherr, D., and Buss, M. (2008). Human–robot collaboration: a survey. *International Journal of Humanoid Robotics*, 5(01):47–66.

Beeson, P., Modayil, J., and Kuipers, B. (2010). Factoring the mapping problem: Mobile robot map-building in the hybrid spatial semantic hierarchy. *The International Journal of Robotics Research*, 29(4):428–459.

Berkhin, P. et al. (2006). A survey of clustering data mining techniques. *Grouping multidimensional data*, pages 25–71.

Blum, C., Puchinger, J., Raidl, G. R., and Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6):4135–4151.

Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm intelligence: from natural to artificial systems.* Oxford University Press, Inc.

Boussaïd, I., Lepagnot, J., and Siarry, P. (2013). A survey on optimization metaheuristics. *Information Sciences*, 237:82–117.

Brintaki, A. N. and Nikolos, I. K. (2005). Coordinated UAV path planning using differential evolution. *Operational Research*, 5(3):487–502.

Brown, J. S., Collins, A., and Duguid, P. (1989). Situated cognition and the culture of learning. *Educational researcher*, 18(1):32–42.

Brun, M., Sima, C., Hua, J., Lowey, J., Carroll, B., Suh, E., and Dougherty, E. R. (2007). Model-based evaluation of clustering validation measures. *Pattern recognition*, 40(3):807–824.

Bussone, A., Stumpf, S., and O'Sullivan, D. (2015). The Role of Explanations on Trust and Reliance in Clinical Decision Support Systems. In *2015 International Conference on Healthcare Informatics (ICHI)*, pages 160–169. IEEE.

Canny, J. (1988). *The complexity of robot motion planning.* MIT press.

Černỳ, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications*, 45(1):41–51.

Chai, J. Y., Fang, R., Liu, C., and She, L. (2016). Collaborative Language Grounding Toward Situated Human-Robot Dialogue. *AI Magazine*, 37(4):32–45.

Charalampous, K., Kostavelis, I., and Gasteratos, A. (2017). Recent trends in social aware robot navigation: A survey. *Robotics and Autonomous Systems*, 93:85–104.

Chen, G. (2015). Deep learning with nonparametric clustering. *arXiv preprint arXiv:1501.03084*.

Chik, S., Yeong, C., Su, E., Lim, T., Subramaniam, Y., and Chin, P. (2016). A review of social-aware navigation frameworks for service robot in dynamic human environments. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, 8(11):41–50.

Chown, E., Kaplan, S., and Kortenkamp, D. (1995). Prototypes, location, and associative networks (plan): Towards a unified theory of cognitive mapping. *Cognitive Science*, 19(1):1–51.

Chrastil, E. R. and Warren, W. H. (2014). From cognitive maps to cognitive graphs. *PloS one*, 9(11):e112544.

Clancey, W. J. (1997). *Situated cognition: On human knowledge and computer representations.* Cambridge University Press.

Coleman, M. and Liau, T. L. (1975). A Computer Readability Formula Designed for Machine Scoring. *Journal of Applied Psychology*, 60(2):283–284.

Congleton, R. D. (2004). The median voter model. In *The encyclopedia of public choice*, pages 707–712. Springer.

Conlin, J. A. (2009). Getting around: making fast and frugal navigation decisions. *Progress in brain research*, 174:109–117.

Coombs, C. H. and Avrunin, G. S. (1977). Single-peaked functions and the theory of preference. *Psychological review*, 84(2):216–230.

Dalton, R. C. (2003). The secret is to follow your nose: Route path selection and angularity. *Environment and Behavior*, 35(1):107–131.

Dietterich, T. G. et al. (2000). Ensemble methods in machine learning. *Multiple classifier systems*, 1857:1–15.

Dobslaw, F. (2010). Recent development in automatic parameter tuning for metaheuristics. In *Proceedings of the 19th Annual Conference of Doctoral Students-WDS 2010*, pages 54–63.

Dorigo, M., Birattari, M., and Stutzle, T. (2006). Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39.

Elhamifar, E., Sapiro, G., and Vidal, R. (2012). See all by looking at a few: Sparse modeling for finding representative objects. In *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1600–1607. IEEE.

Epstein, S. L. (1994). For the Right Reasons: the FORR Architecture for Learning in a Skill Domain. *Cognitive Science*, 18(3):479–511.

Epstein, S. L. (1998). Pragmatic navigation: Reactivity, heuristics, and search. *Artificial Intelligence*, 100(1):275–322.

Epstein, S. L. (2001). Learning to Play Expertly: A Tutorial on Hoyle. In *Machines That Learn to Play Games*, pages 153–178. Nova Science Publishers, Inc.

Epstein, S. L., Aroor, A., Evanusa, M., Sklar, E. I., and Parsons, S. (2015). Learning spatial models for navigation. In *International Workshop on Spatial Information Theory*, pages 403–425. Springer.

Epstein, S. L., Passonneau, R., Gordon, J., and Ligorio, T. (2011). The Role of Knowledge and Certainty in Understanding for Dialogue. In *2011 AAAI Fall Symposium Series*, pages 90–97.

Fink, G. A. (2008). n-gram models. *Markov Models for Pattern Recognition: From Theory to Applications*, pages 95–113.

Flach, P. (2012). *Machine learning: the art and science of algorithms that make sense of data.* Cambridge University Press.

Fong, S., Deb, S., and Chaudhary, A. (2015). A review of metaheuristics in robotics. *Computers & Electrical Engineering*, 43:278–291.

Foo, P., Warren, W. H., Duchon, A., and Tarr, M. J. (2005). Do humans integrate routes into a cognitive map? map-versus landmark-based navigation of novel shortcuts. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 31(2):195–215.

Friedenberg, J. and Silverman, G. (2011). *Cognitive science: An introduction to the study of mind.* Sage.

Gallistel, C. R. (1990). *The organization of learning.* The MIT Press.

Gatt, A. and Krahmer, E. (2018). Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.

Geng, N., Gong, D., and Zhang, Y. (2013). Robot path planning in an environment with many terrains based on interval multi-objective PSO. In *2013 IEEE Congress on Evolutionary Computation*, pages 813–820. IEEE.

Giannopoulos, I., Kiefer, P., Raubal, M., Richter, K.-F., and Thrash, T. (2014). Wayfinding decision situations: A conceptual model and evaluation. In *International Conference on Geographic Information Science*, pages 221–234. Springer.

Gigerenzer, G., Todd, P. M., ABC Research Group, t., et al. (1999). *Simple heuristics that make us smart.* Oxford University Press.

Glover, F. (1989). Tabu search-part I. *ORSA Journal on computing*, 1(3):190–206.

Glover, F. (1990). Tabu search-part II. *ORSA Journal on computing*, 2(1):4–32.

Glover, F. W. and Kochenberger, G. A. (2003). *Handbook of metaheuristics*, volume 57. Springer Science & Business Media.

Golledge, R. G. (1999). Human wayfinding and cognitive maps. *Wayfinding behavior: Cognitive mapping and other spatial processes*, pages 5–45.

Goodrich, M. A. and Schultz, A. C. (2007). Human-robot interaction: a survey. *Foundations and trends in human-computer interaction*, 1(3):203–275.

Gordon, D. and Subramanian, D. (1997). A cognitive model of learning to navigate. In *Proceedings of the 19th Conference of the Cognitive Science Society*, volume 25, pages 271–276.

Gordon, D., Subramanian, D., Haught, M., and Kobayashi, R. (1998). Modeling individual differences in learning a navigation task. In *Proceedings of the 20th Conference of the Cognitive Science Society*, volume 20, pages 418–423.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182.

Hall, E. T. (1963). A system for the notation of proxemic behavior. *American anthropologist*, 65(5):1003–1026.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer, 2 edition.

Hayduk, L. A. (1978). Personal space: An evaluative and orienting overview. *Psychological Bulletin*, 85(1):117–134.

Hayduk, L. A. (1994). Personal space: Understanding the simplex model. *Journal of Nonverbal Behavior*, 18(3):245–260.

Hegarty, M., Richardson, A. E., Montello, D. R., Lovelace, K., and Subbiah, I. (2002). Development of a self-report measure of environmental spatial ability. *Intelligence*, 30(5):425–447.

Helbing, D., Keltsch, J., and Molnar, P. (1997). Modelling the evolution of human trail systems. *Nature*, 388(6637):47–50.

Helbing, D. and Molnar, P. (1995). Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282–4286.

Henkel, Z., Bethel, C. L., Murphy, R. R., and Srinivasan, V. (2014). Evaluation of proxemic scaling functions for social robotics. *IEEE Transactions on Human-Machine Systems*, 44(3):374–385.

Hernández, C., Asín, R., and Baier, J. A. (2015). Reusing previously found A* paths for fast goal-directed navigation in dynamic terrain. In *Proceedings of the AAAI Conference of Artificial Intelligence (AAAI)*, pages 1158–1164.

Hochmair, H. and Frank, A. U. (2000). Influence of estimation errors on wayfinding-decisions in unknown street networks–analyzing the least-angle strategy. *Spatial Cognition and Computation*, 2(4):283–313.

Hölscher, C., Tenbrink, T., and Wiener, J. M. (2011). Would you follow your own route description? Cognitive strategies in urban route planning. *Cognition*, 121(2):228–247.

Hoos, H. H. (2011). Automated algorithm configuration and parameter tuning. In *Autonomous Search*, pages 37–71. Springer.

Iaria, G., Petrides, M., Dagher, A., Pike, B., and Bohbot, V. D. (2003). Cognitive strategies dependent on the hippocampus and caudate nucleus in human navigation: variability and change with practice. *Journal of Neuroscience*, 23(13):5945–5952.

Jiang, R., Ge, S. S., Tangirala, N. T., and Lee, T. H. (2016). Interactive navigation of mobile robots based on human's emotion. In *International Conference on Social Robotics*, pages 243–252. Springer.

Kallai, J., Makany, T., Karadi, K., and Jacobs, W. J. (2005). Spatial orientation strategies in morris-type virtual water task for humans. *Behavioural Brain Research*, 159(2):187–196.

Karaboga, D. and Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, 39(3):459–471.

Kendon, A. (1990). *Conducting interaction: Patterns of behavior in focused encounters*, volume 7. CUP Archive.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of the 1995 IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE.

Kennedy, W. G., Bugajska, M. D., Marge, M., Adams, W., Fransen, B. R., Perzanowski, D., Schultz, A. C., and Trafton, J. G. (2007). Spatial Representation and Reasoning for Human-Robot Collaboration. In *Proceedings of the Twenty Second Conference on Artificial Intelligence*, volume 7, pages 1554–1559.

Kim, B. and Pineau, J. (2016). Socially adaptive path planning in human environments using inverse reinforcement learning. *International Journal of Social Robotics*, 8(1):51–66.

Kirkpatrick, S., Gelatt, C., and Vecchi, M. (1983). Optimization by simulated annealing. *Science*, 220:671–680.

Kirsch, A. (2016). Heuristic decision-making for human-aware navigation in domestic environments. In Benzmüller, C., Sutcliffe, G., and Rojas, R., editors, *GCAI 2016. 2nd Global Conference on Artificial Intelligence*, volume 41 of *EPiC Series in Computing*, pages 200–213. EasyChair.

Koenig, S. and Likhachev, M. (2002). D* lite. In *Proceedings of the AAAI Conference of Artificial Intelligence (AAAI)*, pages 476–483.

Koenig, S., Likhachev, M., and Furcy, D. (2004). Lifelong planning. *Artificial Intelligence*, 155(1):93–146.

Kolodner, J. (2014). *Case-based reasoning.* Morgan Kaufmann.

Korf, R. E. (2014). Search: A survey of recent results. In *Exploring Artificial Intelligence: Survey Talks from the National Conferences on Artificial Intelligence*, pages 197–237. Morgan Kaufmann.

Korpan, R. and Epstein, S. L. (2018). Toward natural explanations for a robot's navigation plans. In *Proceedings of Workshop on Explainable Robotic Systems at HRI 2018.*

Korpan, R., Epstein, S. L., Aroor, A., and Dekel, G. (2017). Why: Natural explanations from a robot navigator. In *Proceedings of AAAI 2017 Fall Symposium on Natural Communication for Human-Robot Collaboration.*

Kruse, T., Pandey, A. K., Alami, R., and Kirsch, A. (2013). Human-aware robot navigation: A survey. *Robotics and Autonomous Systems*, 61(12):1726–1743.

Kuipers, B. (1978). Modeling spatial knowledge. *Cognitive science*, 2(2):129–153.

Kuipers, B. (2000). The spatial semantic hierarchy. *Artificial intelligence*, 119(1-2):191–233.

Kulesza, T., Stumpf, S., Burnett, M., Yang, S., Kwan, I., and Wong, W.-K. (2013). Too Much, Too Little, or Just Right? Ways Explanations Impact End Users' Mental Models. In *2013 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 3–10. IEEE.

Kuncheva, L. I. and Whitaker, C. J. (2003). Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine learning*, 51(2):181–207.

Landsiedel, C., Rieser, V., Walter, M., and Wollherr, D. (2017). A Review of Spatial Reasoning and Interaction for Real-World Robotics. *Advanced Robotics*, 31(5):222–242.

Latini-Corazzini, L., Nesa, M. P., Ceccaldi, M., Guedj, E., Thinus-Blanc, C., Cauda, F., Dagata, F., and Péruch, P. (2010). Route and survey processing of topographical memory during navigation. *Psychological research*, 74(6):545–559.

Law, M. T., Urtasun, R., and Zemel, R. S. (2017). Deep spectral clustering learning. In *International Conference on Machine Learning*, pages 1985–1994.

Lazar, J., Feng, J. H., and Hochheiser, H. (2017). *Research methods in human-computer interaction.* Morgan Kaufmann.

Leighton, J. P. (2004). Defining and describing reason. *The Nature of Reasoning*, pages 3–11.

Liang, J.-H. and Lee, C.-H. (2015). Efficient collision-free path-planning of multiple mobile robots system using efficient artificial bee colony algorithm. *Advances in Engineering Software*, 79:47–56.

Lim, B. Y., Dey, A. K., and Avrahami, D. (2009). Why and Why Not Explanations Improve the Intelligibility of Context-Aware Intelligent Systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2119–2128. ACM.

Luber, M., Spinello, L., Silva, J., and Arras, K. O. (2012). Socially-aware robot navigation: A learning approach. In *2012 IEEE/RSJ international conference on Intelligent Robots and Systems (IROS)*, pages 902–907. IEEE.

Malle, B. F. (1999). How People Explain Behavior: A New Theoretical Framework. *Personality and Social Psychology Review*, 3(1):23–48.

Manikas, T. W., Ashenayi, K., and Wainwright, R. L. (2007). Genetic algorithms for autonomous robot navigation. *IEEE Instrumentation & Measurement Magazine*, 10(6):26–31.

Markkanen, R. and Schröder, H. (1997). *Hedging and Discourse: Approaches to the Analysis of a Pragmatic Phenomenon in Academic Texts*, volume 24. Walter de Gruyter.

Masehian, E. and Sedighizadeh, D. (2010). A multi-objective PSO-based algorithm for robot path planning. In *2010 IEEE International Conference on Industrial Technology (ICIT)*, pages 465–470. IEEE.

Mead, R. and Matarić, M. J. (2017). Autonomous human–robot proxemics: socially aware navigation based on interaction potential. *Autonomous Robots*, 41(5):1189–1201.

Miller, T. (2017). Explanation in artificial intelligence: Insights from the social sciences. *arXiv preprint arXiv:1706.07269*.

Mittal, S. and Deb, K. (2007). Three-dimensional offline path planning for UAVs using multiobjective evolutionary algorithms. In *2007 IEEE Congress on Evolutionary Computation*, pages 3195–3202. IEEE.

Moussaid, M., Garnier, S., Theraulaz, G., and Helbing, D. (2009). Collective information processing and pattern formation in swarms, flocks, and crowds. *Topics in Cognitive Science*, 1(3):469–497.

National Search and Rescue Council (2017). *National search and rescue manual*. Australian Maritime Safety Authority (AMSA).

Nomura, T., Kanda, T., Suzuki, T., and Kato, K. (2008). Prediction of human behavior in human–robot interaction using psychological scales for anxiety and negative attitudes toward robots. *IEEE Transactions on Robotics*, 24(2):442–451.

Oh, J., Howard, T. M., Walter, M. R., Barber, D., Zhu, M., Park, S., Suppe, A., Navarro-Serment, L., Duvallet, F., Boularias, A., et al. (2016). Integrated Intelligence for Human-Robot Teams. In *International Symposium on Experimental Robotics*, pages 309–322. Springer.

Okal, B. and Arras, K. O. (2016). Learning socially normative robot navigation behaviors with bayesian inverse reinforcement learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2889–2895. IEEE.

Park, C., Ondřej, J., Gilbert, M., Freeman, K., and O'Sullivan, C. (2016). HI robot: Human intention-aware robot planning for safe and efficient navigation in crowds. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3320–3326. IEEE.

Pazzaglia, F. and De Beni, R. (2001). Strategies of processing spatial information in survey and landmark-centred individuals. *European Journal of Cognitive Psychology*, 13(4):493–508.

Pearl, J. (1984). *Heuristics: Intelligent search strategies for computer problem solving.* Addison-Wesley Pub. Co., Inc., Reading, MA.

Perera, V., Selveraj, S. P., Rosenthal, S., and Veloso, M. (2016). Dynamic Generation and Refinement of Robot Verbalization. In *25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 212–218. IEEE.

Pérez-Fernández, R., Alonso, P., Díaz, I., Montes, S., and De Baets, B. (2017). Monotonicity as a tool for differentiating between truth and optimality in the aggregation of rankings. *Journal of Mathematical Psychology*, 77:1–9.

Petrovic, S. and Epstein, S. (2006). Relative Support Weight Learning for Constraint Solving. In *AAAI Workshop on Learning for Search*, pages 115–122.

Petrovic, S., Epstein, S. L., and Wallace, R. J. (2007). Learning a mixture of search heuristics. In *Proceedings of CP-07 Workshop on Autonomous Search*, pages 1–15.

Polikar, R. (2006). Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, 6(3):21–45.

Poole, D. L. and Mackworth, A. K. (2010). *Artificial Intelligence: Foundations of Computational Agents.* Cambridge University Press.

Raubal, M. and Worboys, M. (1999). A formal model of the process of wayfinding in built environments. In *International Conference on Spatial Information Theory*, pages 381–399. Springer.

Richter, K.-F. and Winter, S. (2014). Cognitive aspects: How people perceive, memorize, think and talk about landmarks. In *Landmarks*, pages 41–108. Springer.

Rios-Martinez, J., Spalanzani, A., and Laugier, C. (2015). From proxemics theory to socially-aware navigation: A survey. *International Journal of Social Robotics*, 7(2):137–153.

Rokach, L. (2010). Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1):1–39.

Rosenberg, A. and Hirschberg, J. (2007). V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL).*

Rosenthal, S., Selvaraj, S. P., and Veloso, M. (2016). Verbalization: Narration of Autonomous Mobile Robot Experience. In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 16, pages 862–868.

Rossi, F., Venable, K. B., and Walsh, T. (2011). A short introduction to preferences: between artificial intelligence and social choice. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 5(4):1–102.

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65.

Russell, S. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach.* Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition.

Sabour, S., Frosst, N., and Hinton, G. E. (2017). Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, pages 3859–3869.

Scalise, R., Rosenthal, S., and Srinivasa, S. (2017). Natural Language Explanations in Human-Collaborative Systems. In *Proceedings of the Companion of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, pages 377–378. ACM.

Scholtz, J. (2003). Theory and evaluation of human robot interactions. In *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, pages 125–135. IEEE.

Spiers, H. J. and Maguire, E. A. (2008). The dynamic nature of cognition during wayfinding. *Journal of Environmental Psychology*, 28(3):232–249.

Stentz, A. (1994). Optimal and efficient path planning for partially-known environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3310–3317. IEEE.

Svenstrup, M., Tranberg, S., Andersen, H. J., and Bak, T. (2009). Pose estimation and adaptive robot behaviour for human-robot interaction. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3571–3576. IEEE.

Talebpour, Z., Viswanathan, D., Ventura, R., Englebienne, G., and Martinoli, A. (2016). Incorporating perception uncertainty in human-aware navigation: A comparative study. In *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 570–577. IEEE.

Thrun, S. (1998a). Finding landmarks for mobile robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 958–963. IEEE.

Thrun, S. (1998b). Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71.

Truong, X.-T. and Ngo, T.-D. (2016). Dynamic social zone based mobile robot navigation for human comfortable safety in social environments. *International Journal of Social Robotics*, 8(5):663–684.

Tversky, B. (1993). Cognitive maps, cognitive collages, and spatial mental models. In *European Conference on Spatial Information Theory*, pages 14–24. Springer.

Unhelkar, V. V., Pérez-D'Arpino, C., Stirling, L., and Shah, J. A. (2015). Human-robot co-navigation using anticipatory indicators of human walking motion. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 6183–6190. IEEE.

Van Erp, M., Vuurpijl, L., and Schomaker, L. (2002). An overview and comparison of voting methods for pattern recognition. In *Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition*, pages 195–200. IEEE.

Wagner, S. and Wagner, D. (2007). *Comparing clusterings: an overview*. Universität Karlsruhe, Fakultät für Informatik Karlsruhe.

Wan, R. (2003). *Browsing and searching compressed documents*. PhD thesis, The University of Melbourne.

Warren, W. H., Rothman, D. B., Schnapp, B. H., and Ericson, J. D. (2017). Wormholes in virtual space: From cognitive maps to cognitive graphs. *Cognition*, 166:152–163.

Weisberg, S. M. and Newcombe, N. S. (2016). How do (some) people make a cognitive map? routes, places, and working memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 42(5):768–785.

Werner, S., Krieg-Brückner, B., and Herrmann, T. (2000). Modelling navigational knowledge by route graphs. In *Spatial Cognition II*, pages 295–316. Springer.

Werner, S., Krieg-Brückner, B., Mallot, H. A., Schweizer, K., and Freksa, C. (1997). Spatial cognition: The role of landmark, route, and survey knowledge in human and robot navigation. In *Informatik'97 Informatik als Innovationsmotor*, pages 41–50. Springer.

Wettschereck, D., Aha, D. W., and Mohri, T. (1997). A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. In *Lazy Learning*, pages 273–314. Springer.

Wiener, J. M., Schnee, A., and Mallot, H. A. (2004). Use and interaction of navigation strategies in regionalized environments. *Journal of Environmental Psychology*, 24(4):475–493.

Wilfong, R. E. (2004). Model search and rescue plan. *Emergency Management Training Specialists*.

Wolbers, T. and Hegarty, M. (2010). What determines our navigational abilities? *Trends in Cognitive Sciences*, 14(3):138–146.

Yang, X., Cai, M., and Li, J. (2016). Path planning for unmanned aerial vehicles based on genetic programming. In *2016 Chinese Control and Decision Conference (CCDC)*, pages 717–722. IEEE.

Zhao, C., Hiam, J. W., Morgan, J. H., and Ritter, F. E. (2011). A multi-strategy spatial navigation model in a text-based environment. In *Proceedings of the 20th Conference on Behavior Representation in Modeling and Simulation*, pages 251–258.