A HYBRID HEURISTICS AND SIMULATION-BASED APPROACH TO DECISION SUPPORT FOR ROBOT-HUMAN TEAM CONFIGURATION

By

TERESA NIETEN

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2011

To Dan and Brandon

ACKNOWLEDGMENTS

I thank my husband, Dan, for his love, support and encouragement; my son Brandon for giving me lots of encouraging hugs and for not driving me completely batty; and my parents, Tom and Diana Griffith, my sister Vickie, and mother-in-law Lois Lastinger for their support, encouragement, and occasional pep talks. My brother-in-law Joe Nieten and my dear friends Dave and Christine Langhorne graciously volunteered to proofread for me. I would also like to thank Dr. Paul Fishwick for not giving up on me.

This work was sponsored, in part, by the United States (US) Army Research Laboratory under Cooperative Agreement W911NF-06-2-0041. Parts of this work were conducted using the Protégé resource, which is supported by grant LM007885 from the United States National Library of Medicine.

TABLE OF CONTENTS

page

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
CCS	Computing Classification System
OWL	Web Ontology Language
RDF	Resource Definition Framework
RDFS	RDF Schema
SPARQL	SPARQL Protocol And RDF Query Language
UAV	Unmanned Air Vehicle
UGV	Unmanned Ground Vehicle
UMS	Unmanned System
USAR	Urban Search And Rescue
WRP	Weighted Region Problem
XML	eXtensible Markup Language

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

A HYBRID HEURISTIC AND SIMULATION-BASED APPROACH TO DECISION SUPPORT FOR ROBOT-HUMAN TEAM CONFIGURATION

By

Teresa Nieten

April 2011

Chair: PAUL FISHWICK
Major: Computer Engineering

Missions that involve tasks such as search-and-rescue or reconnaissance have traditionally involved humans, perhaps with the assistance of one or more robots. The robots, or unmanned systems, are typically teleoperated—operated by remote control to inspect a suspicious object, for example. With the advent of newer and less expensive forms of autonomy and improved human-robot communication, the robots are becoming more capable of acting as peers to their human counterparts, rather than just tools. As the diversity of mixed human-robot teams is increased, so is the complexity of trying to answer questions regarding configuration: what robots should be used, how many, and how many humans should be employed in the teaming process?

This paper presents the research in search of that hybrid approach. Our solution is a decision support approach that employs a hybrid of simulation and artificial intelligence techniques, using a progressive-refinement queuing model to quickly bypass the least desirable configurations. This approach, realized in a software tool, considers the mission requirements and a priori data to determine the optimal team to perform that mission, given a pool of human and robotic resources.

CHAPTER 1
INTRODUCTION

> Teamwork is the ability to work together toward a common vision. The ability to direct individual accomplishments toward organizational objectives. It is the fuel that allows common people to attain uncommon results.
>
> -- Andrew Carnegie [1]

Complex tasks often require more effort than can be put forth by a single person, and the diversity in the skills and abilities of the members allows for tasks to be divided into specializations, rather than being replicated. This division of labor works to make an effective, synergistic team – provided the team collectively has the abilities and tools required to do the job. In a football game, if an offensive lineman does not perform his job, the quarterback could get sacked, and the team could lose out on the opportunity to score a touchdown. If a scrub nurse does not know how to identify surgical instruments in an operating room, the patient could die. Tchaikovsky's *1812 Overture* would end with a whimper instead of a bang if the bass drums were substituted with triangles. Whether it is a symphony, an assembly line, an operating room, a search and rescue operation, a sports team, or a military patrol, the success of the task depends on each member performing his or her assigned duties.

Just as crucial to the performance of a team is the preparation that goes into creating the team. No matter how well a group works together, if they collectively do not have the basic skills required to do the job, their effectiveness is limited and the probability of the successful completion of the job is significantly reduced. A college football team could recruit the best quarterback and receivers in the country, but if they neglect to field a center during an offensive play, the team will not be successful. Likewise, an operating room full of surgeons with no anesthesiologist will not yield a

successful operation. For a quartet to harmonize correctly, it must include a lead, tenor, bass, and baritone. In a search and rescue operation, members of the team must be able to locate victims, assess the safety of the rescue area, and assist victims in leaving the rescue area.

As man has evolved, he has learned to use tools of increasing complexity. From sharpening sticks to chipping flint to forging steel to developing complex machinery, people have become proficient at crafting tools to help get the job done – and as some of these tools themselves have become increasingly complex, we are arriving at a point that we can now consider some of these tools to be more than just tools. A subset of the tools has now evolved and matured enough that we can now elevate them to the status of individual contributors to the team effort – unique, self-sustaining team members, working alongside their human teammates to accomplish the task. We know them as either 'robots' or 'unmanned systems (UMS).'

Background

On a clear Fall day in 2001, a disaster of unthinkable proportions occurred. The twin 110-story towers of the World Trade Center (WTC) were destroyed as a pair of hijacked airplanes struck, seventeen minutes apart. At the time of the attacks, there were roughly 17,000 people inside the towers [2]. While many of those people were able to evacuate, thousands more were left trapped in the rubble, including nearly 3000 dead. Among those killed were 411 emergency response workers [3] who were trying to come to the aid of those trapped and injured inside the rubble of the buildings. Subsequently, studies have shown that the dust and pollutants that filled the air in the wake of the attacks have caused a significant increase in respiratory problems in first responders [4], causing problems to surface well after the initial danger had passed.

Some of the rescuers, however, were unaffected by the dust, danger of collapse, or tight spaces. In what is touted to be the first time, a team of rescue robots (Figure 1-1 and 1-2) was deployed to help search for victims, identify evacuation paths, inspect for structural issues, and detect hazardous materials [5].

Figure 1-1. Rescue robots used in aftermath of September 11 terrorist attacks (credit: CRASAR)

Figure 1-2. Robot in search and rescue operation

Rescue robots were also used in the aftermath of Hurricane Katrina [6], and have been used in several mine disaster rescue attempts, including the Pike River mine in New Zealand and Crandall Canyon, Utah [7]. By adding robots to the team in a rescue operation, the robots can be sent in ahead of the human members to sense the environment and detect hazardous conditions that may exist (Figure 1-3). Robots can also be useful in helping victims get to safety (Figure 1-4, Figure 1-5) or even performing first aid (Figure 1-6).

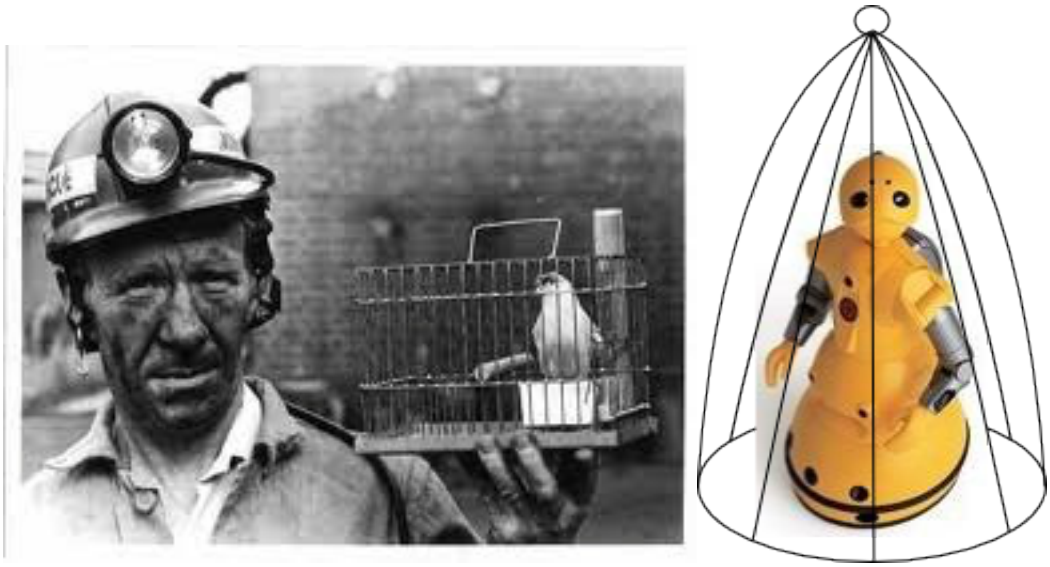

Figure 1-3. Robots can now take on the role of canary in a coal mine during a mine disaster recovery operation.

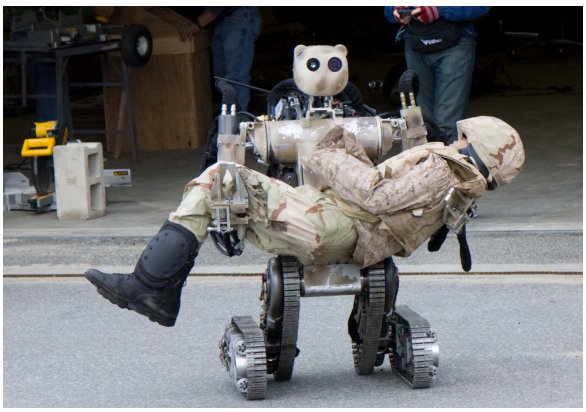

Figure 1-4. Rescue robot carries an injured soldier (credit army.mil)

Figure 1-5. Another possible rescue robot that can carry the injured party to safety

Figure 1-6. Robot assists victim in rescue exercise (credit: CRASAR, University of South Florida)

Also in 2001, Congress passed a law setting goals for the US Military to have one-third of its operational deep strike force aircraft fleet be unmanned by 2010 and one-third of its operational combat ground vehicles be unmanned by 2015[9]. The ultimate goal is to save the lives of troops by using as many UMS in hazardous situations as possible, allowing the human soldiers to stay in protected areas while the more expendable robotic counterparts neutralize or at least identify the risks – much like the goal of the Mine Safety and Health Administration (MSHA) and CRASAR with their mine rescue and recovery programs [7]. An additional goal set forth by the Joint Robotics

Program (JRP) is to reduce the ratio of human operators to robotic platforms by increasing autonomy levels of unmanned systems [10], thus encouraging the creation of more autonomous or semi-autonomous robots that can operate alongside of human members of the team. The U.S. Army's Army Model and Simulation Office (AMSO) developed a program to quickly and efficiently request and evaluate proposals for modeling and simulation research in an urban environment [8].

One way to meet the goals set by Congress and the JRP, and to help safeguard humans, whether they are military personnel, rescuers, or people trying to accomplish potentially dangerous tasks such as mining or farming, is to create teams comprised of humans and robots (Figure 1-7 and Figure 1-8). A mixed team would allow the robots to perform the more dangerous tasks while the human counterparts provide in situ decisions that humans are uncomfortable allowing machines to make, such as authorizing weapons fire or identifying a target. As the autonomy level of unmanned systems increases, the ratio of unmanned vehicles to humans can increase, including having some teams fully composed of unmanned systems. The WTC search and rescue operation showed that this is possible, but was just a first step in having cooperative teams of humans and robots.

In a military unit, as well as with most non-military teams, there is an established chain of command, with one team member dividing tasks among the entire team. As teams become more heavily unmanned, the tasking becomes more complex, especially for units composed entirely of unmanned members and the availability of a wider variety of unmanned resources available for selection. Additionally, as humans and unmanned vehicles are integrated, there is a need for consistent training of human team leads to

learn how to effectively use all members of the team on a mission and, just as important, how to choose the correct members. Determining the makeup of a team in an objective manner, based on the overall goals and success criteria of the task, is crucial in obtaining a successful outcome.

Figure 1-7. Pairing humans and robots allows the robots to do the dangerous jobs.

Figure 1-8. A team of soldiers and robot on a search and rescue or reconnaissance mission

Problem Statement

There are many factors to consider when choosing members of a team. Having the skills needed to perform one or more of the tasks within the mission is crucial; each member must physically be able to travel to and from the area of interest, either independently or be transported by another member of the team; in addition, other less obvious factors come into play, such as cost to deploy or replace. Some robots can operate autonomously, semi-autonomously, or be remotely operated; still others require operators or technicians to be deployed as part of the team due to maintenance, operability, transportability, and other collocation requirements. A given task may require an expertise that can only be developed with extensive – and expensive – training. These factors all impact the cost, including both financial and availability for other tasks, of assigning that resource to a team. With a large field of resources, both human and robot, and many variables to consider, the task of choosing the optimal configuration, whether that is time, cost, or some other combination, becomes more daunting. Continuous resource planning and replanning becomes a tedious task, one that is well suited to be automated, or at least semi-automated.

Solution

The purpose of this research was to create a methodology, realized in a software tool, which considers the mission requirements and a priori data to determine the optimal team to perform that mission, given a pool of human and robotic resources. This method employs a hybridized approach of simulation and artificial intelligence techniques, using a dynamic progressive-refinement queuing model to quickly bypass the least desirable configurations. The resulting tool can be used during mission planning to allocate the appropriate resources, or in training to help team leads be

consistent in choosing team members; it can be used prior to the operation or to dynamically reallocate members of the team as situations arise requiring re-tasking or refocusing.

Contributions to Knowledge

The contributions from this research are in two areas – 1) the hybridization of a heuristics-based semi-optimal path planning and multiple simulations in the presence of two input sets, and 2) workgroup selection automation. The goal of contribution # 1 is to demonstrate that a hybrid progressive refinement simulation and heuristic based planning method finds the desired solution, in this case a near-optimal team configuration, faster and using fewer computational resources than either a stand-alone all-paths progressive-refinement simulation or a heuristic based planning paired with brute force non-refined simulation. The goal of contribution # 2 is to delineate a clear planning approach to near optimal team-based composition where teams have human and robot members with diverse skill sets, operational characteristics, and operating costs.

CHAPTER 2
RELATED RESEARCH

Overview

The research related to this work spans several areas – teamwork between humans and UMS, team planning and scheduling, path and mission planning, and simulation. The following sections discuss the relevant related work done in those areas. First, we will look at research involving the pairing of humans and unmanned systems. Next we will investigate work done in the area of team planning and scheduling, followed by research into mission and path planning. Finally, we will focus on simulation techniques, specifically where they apply to decision making.

Teamwork between Humans and Unmanned Systems

There are several groups performing research into humans and UMS working together to solve a problem or complete a task. The Idaho National Engineering and Environmental Laboratory has performed research on developing a control architecture that allows different members of a mixed team of humans and UMS's to hand off the leadership role depending on the task, along with varying the level of autonomy of the UMS members [11][12]. Their research is focused on the interaction between the team members, the shifting roles and responsibilities between team members during an operation, and providing individual autonomy to the UMS members of the team, autonomy that is crucial during periods of poor communication capabilities.

The Human-Robot Interaction Operating System focuses on how human and robot team members communicate to solve a problem; it currently handles tasking at an individual resource request level and deals with "operational tasks" – tasks that are highly detailed and well-defined for a human/robot pair [13]. It does allow for dynamic

team assignments, but only on the level of deciding which UMS is best equipped to handle a single task requested by a human, or which human is best equipped to handle a single task requested by a UMS – basically assigning a temporary assistant to a worker.

Levesque studied teamwork using joint commitment, where members of a team work towards a common goal and mathematically negotiate the perceived possibility – or more importantly, impossibility – of achieving that goal [14]. In this work, the importance of team members working towards that common goal takes precedence over an individual's optimal results. In other words, what appears to be the best course of action for a single team member gets ignored in the hopes that the partner member will assist that member in achieving the goal, even if it appears to be futile, or the partner member will ultimately encourage that member to stop if they agree it is futile.

Communication, interpretation, coordination, and cooperation between human and robotic team members, where robotic team members primarily function as assistants to the human members, were the primary focus of a research project and resulting multi-agent architecture from the Institute for Real-Time Computer Systems and Robotics (IPR) [15]. In that study, one of the primary goals was to adapt a fully automated situation to allow the cooperation of humans, rather than isolating the robots in a fully automated factory setting. Their assertion was that UMS's are less flexible but more reliable, whereas humans are more flexible but less reliable, in terms of memory, consistency, and fatigue, so by pairing humans and robots together, the strengths of each can be exploited instead of isolating the robots from humans. In addition, the IPR

study included the domain of service robots, where robots operate alongside humans in a cooperative service mode.

The Center for Robot Assisted Search & Rescue (CRASAR), started at the University of South Florida and subsequently moved to Texas A&M, focuses on using robots for search and rescue efforts, and the human-robot interaction (HRI) required for such efforts [16]. The CRASAR work is primarily focused on how the humans involved – not only the human members of the search and rescue team but also the victims – interact with the robots to provide and receive situational awareness. In addition their work brings up the subject of communication between team members in an environment with limited communication. Subsequent work by CRASAR and MSHA defined and ranked skills, system components, and physical characteristics that are valuable in a rescue robot in different scenarios [7].

The Robotics Institute at Carnegie Mellon developed a multi-agent system (MAS) for coordinating teamwork between multiple robots and humans, specifically targeted towards an urban search and rescue (USAR) application. In their work developing RETSINA MAS, they explored the difficulty in getting humans and robots to work together seamlessly, for several reasons: it is difficult for robots to readily communicate their intentions to the human members, humans and robots are typically trained in different manners, and team-level collaboration among multi-robot teams is still an emergent technology, so most autonomous or semi-autonomous robots are still task-driven rather than team-driven [17].

The Applied Physics Laboratory (APL) at John Hopkins University (JHU/APL) modified swarm-based behaviors to study strong autonomy in multi-robot teams, relying

on wireless communication and co-observation for situational awareness in an Intelligence, Surveillance, and Reconnaissance (ISR) scenario. In strong autonomy, commander's intent and high-level operations are provided to the UMS, and the specific course of action is devised on-board the UMS using Dynamic Co-Fields [18]. In swarm behavior, which is based on swarming insect behavior, control is decentralized such that the behavior of the group is influenced by the cooperative actions of the individual members [19].

Subsequent work at JUH/APL continued the study of cooperative robotic behaviors using stigmergic potential fields, in which an individual vehicle operates in a locally-held model of that vehicle and its environment, and the vehicles and their behaviors are modeled in a hierarchical organization. The behaviors are developed as Effects Based Operations (EBO) [20]. Both of the JHU/APL efforts studied teams comprised entirely of UMS.

The TEAMCARE lab at University of Southern California (USC) presented a chronology of multi-agent teamwork research over the last twenty years. They discussed the Belief Desire Intention (BDI) model, which placed the emphasis on execution-time decision-making, and the Distributed Constraint Optimization Problem (DCOP) and the Decentralized Partially Observable Markov Decision Problem (DEC-POMDP), which put more effort into planning-time decision-making. They suggest that a better approach might be to incorporate more execution-time decision-making into the planning-time approaches, allowing for a more robust system, thus reducing the model uncertainty that produces much of the error in DCOP and DEC-POMDP [21]. The

TEAMCARE group later devised a hybrid BDI-POMDP framework to demonstrate that concept. [22]

A group from Wright State University's Department of Biomedical, Industrial, and Human Factors Engineering studied human effectiveness in working with unmanned combat aerial vehicles (UCAVs) [23]. Their focus was on reducing operator complexity rather than having humans and UMS's working as peers on a team. Reducing operator complexity allows for more autonomy in the UMS members, thus providing more opportunity for UMS members to participate in complex, team-oriented tasks.

In all of these cases, the focus of work has been on how to effectively manage existing human-robot teams or existing multi-robot teams. The effort of planning the structure of the team is largely ignored.

Team Planning and Scheduling

Murphy took the effort of team planning a step further than the others by defining workflow roles and responsibilities in a search and rescue operation and defining possible insertion points of robots into the team [16]. Defining the roles and responsibilities can help in two ways: ensuring that each member of the team has a specific role ensures that every member knows what to do and provides assurance that full coverage of needs are assigned; and if used in a team selection manner, the defined roles based on abilities can help ensure that the newly-created team has the skills needed for the job covered by the members.

In the area of team planning and scheduling, there have been several research projects that have used simulation to evaluate assignment of teams, or crews. For the railroad industry, Canadian National Rail and Circadian Technologies, Inc., used discrete event simulation to assign crews to trains. Their problem involved optimizing

crew schedules with respect to minimizing cost, subject to rules and regulations for worker safety [26]. At the same conference, a team from Embry-Riddle Aeronautical University discussed using simulation for manpower planning for aircraft maintenance workers [25]. Their goal was to develop an optimal assignment of maintenance crews based on aircraft flight and maintenance schedules; however, they assumed all workers had the same skills.

Software project planning, from resources to project scheduling, was the subject of research by Joslin and Poole. They modeled individual resource skill sets as probability distributions for contributing to the project rather than specific skills needed for a project or task [32]. With this approach, there is much work that has to be done at probability assignment time to effectively model the ability a member might have in contributing to the project.

The Multiagent Adjustable Autonomy Framework (MAAF) for mixed human/robot teams touched the areas of human robot teamwork and team planning by designing a framework for mixed teams that allows individual autonomy and teamwork, and uses software agents to assign team members to specific tasks at execution time, based on the member's abilities and availability [24]. The actual team selection is performed prior to beginning their process and their assignment is to individual tasks, rather than the composition of the team.

Miller [31] discussed the role of simulation in planning with a two-step approach in which partial plans were devised, then revised through simulation, in a constraint optimization problem. The domain he chose to model was planning equipment usage and staging on a manufacturing floor. He suggests that for a planner to be effective, it

cannot be completely domain-independent, but instead requires some knowledge pertaining to the domain in order to choose the more appropriate plan.

In each of these scenarios, individual tasks were assigned but there was little effort involved in assigning a subset of members to the team. Instead most dealt more with arrangement and scheduling of existing team members. This indicates a gap in the area of team scheduling.

Mission and Path Planning

One of the inspirations for this research was work done by Lee and Fishwick in the area of Simulation Based Planning to assist in mission planning, using multimodeling to simulate the mission plans at varying levels of abstraction, depending on time available. The simulations were demonstrated with unit-level planning [30]. Their later work included simulation-based route planning [34]. This work carries their concepts into optimal team planning, with a heuristics-based twist.

In the area of path planning, Ganapathy and Hill combined heuristics-based path planning algorithms with simulation to study the World War II Bay of Biscay U-boat hunting mission [27]. Hu, Li, Guo, Sun, and Zeng applied a heuristics-based transformation to a vehicle routing problem as part of the simulation [36]. Another area of research in simulation-based planning includes the Navy's Mine Warfare Command's Naval Mine Warfare Simulation, which uses faster-than-realtime simulations to aid in planning mine countermeasures missions, running numerous simulations using a Monte Carlo method. They also use it to perform force-level studies to determine the resource level and types required for a given scenario [37].

Another project that combined heuristics-based path-planning algorithms with simulation had similar goals to this proposed research, though at a single UAV level

[28]. In that project, they used directed acyclic graphs to model roadways in an effort for the UAV to find a ground-based vehicle traveling on those roads. In their case, they traverse all of the nodes and edges of the graph until they find the target.

In their survey on graph management and mining algorithms, Aggarwal and Wang discussed the relationship between reachability techniques and optimal routing problems in computer networks [42]. Sensor management was the initial problem a Swedish team sought to solve using simulation-based planning. In their initial demonstration, they used simulation-based planning to determine correct deployment locations of UAVs and sensors in an attempt to follow a person of interest, but they laid the groundwork for creating a simulation-based planner for allocating resources [38].

The EyeRobot project used the heuristics-based path planning technique of Weighted Region Problem (WRP) [39]. In that project, they determined the globally optimum path for a single robot to travel from one point to another, using navigability factors as weights on polygonal map regions, and then the robot performed local path planning during the traversal to avoid obstacles. Global path planning uses static map data such as roads, houses, fields, and bodies of water to determine that higher level path, and local path planning is runtime sensor-based planning. EyeRobot used a single optimal path-planning algorithm, and a polygonal region and test the viability of both vector and grid based planning.

Uncertain terrain and risks, along with uneven natural boundaries, makes planning for an autonomous vehicle difficult. One group used a fuzzy approach to dealing with these uncertainties in WRP [40]. Their goal was to plan a path through different

weighted terrains and areas that had other risks in the context of the WRP where polygonal regions are assigned weights.

Planning a path for a single vehicle has its own challenges, which have been subject of much research to date. Finding an optimal path for multiple, disparate vehicles becomes a significantly more complex problem when the factors of collision avoidance and rendezvous timing are considered. In one attempt, multiple single-vehicle optimal paths were generated using a bounded curvature path planning algorithm, using both Lagrangian duality theory and a penalty function method for convex regions, and the path optimization was extended to multiple vehicles using an elastic multi-particle (vehicle) system to chain the vehicles together [41].

As several of these projects show, there has been some work to date on combining heuristic algorithms with simulation. Much of it has centered on single entity movement, when it comes to the path planning domain, or using heuristic algorithms within a simulation to plan a multi-step function, such as stacking then moving palates for shipping [56].

Simulation Techniques

Simulation is an invaluable tool in decision support. Done with care, it can make the job of the user easier and decrease costs. In their 2003 paper, Hill and Malone discuss some of the common problems encountered in simulation for decision support, and how to avoid them. The first issue, conflicting results from different models, can be mitigated by an open dialog between different modelers to ensure free exchange of ideas and opportunity for brainstorming. The second issue, different interpretations of the simulation results, is not a problem exclusive to simulation; the most effective way to

mitigate that risk is to model at the level of the audience, and to communicate regularly [29].

A team from Auburn University studied using multisimulation for decision support in an uncertain environment [54]. Their work focused on adaptive responses to changes and/or new information in the course of the simulation. Their incorporation of learning algorithms for an adaptive simulation model is of interest in a dynamic environment, such as for team replanning for a subtask that arises during an existing task.

Progressive refinement, one of the techniques we use in this project, is most often associated with visualization [43][44][45][46] and dealing with limited resources and large data sets, which is why it is such a good fit for visualization. Attempts at creating an architecture for parallel progressive refinement have demonstrated the scalability of the approach, with linear growth [45]. Indeed, scalability is a desired outcome of most projects, which is a reason that technique was chosen for our project.

Progressive refinement starts out with a simple model of course granularity, increasing the complexity and granularity of the critical elements of the model at each iteration; the initial definition of the model's dimensions and the granularity is key to the success of the problem. Using a granularity that is too fine would be equivalent to flattening the approach out to be brute-force; too course and the results at the courser levels are inaccurate and provide little to no useful feedback for subsequent levels [52].

In progressive refinement radiosity, the unshot residual energy of a scene is processed for each element, by way of determining the amount of energy reflected back. The accumulated and residual energy values are tallied, and when the

calculations for a single element are exhausted, its residual energy is reset and the algorithm iterates to the next element, until the residual energy is determined to be sufficiently low [43]. This allows the calculations to stop when sufficient detail has been uncovered without doing exhaustive and expensive calculations. Blanford discusses approaches to estimate image portions during a slow download of a large image, using spatial and grayscale variance techniques [47]. This progressive refinement approach allows the recipient of the image to get a good idea of what it looks like before it is fully downloaded.

Going beyond visualization, 20 years after writing *The Mythical Man-Month*, Brooks revisited his theories and drew a correlation between iterative, or incremental, development of a software project to progressive refinement. He compares early testing with stubs and incrementally adding new features to the software to the progressive refinement approach – both give an early, though vague, view of the results [52]. Rosenbaum and Shumann suggest that the “tour-through-the-data” ability to incrementally preview data is beneficial enough to use a progressive refinement approach, regardless of the system resources [48].

An even more compelling use for progressive refinement is demonstrated in Jet Propulsion Laboratory (JPL) research on using progressive refinement in support vector machines (SVM). In that project, they used a reduced set SVM as initial classification and used a slower, more detailed SVM with more support vectors on the results from the reduced SVM with courser granularity to correct the errors. The intended application was image classification with large data sets (pixels) [49].

The Center for Risk and Reliability, in an effort to perform simulation-based risk assessment, created a simulator that performed hierarchical planning and multi-level scheduling. The multi-level scheduling is a variation on progressive refinement, in that the simulation models are generated with multiple levels of detail. Instead of iterating through the levels, however, the multi-level planner chooses a single level to use for each model during the simulation, depending on the importance of that component for that particular scenario [35].

Game engines, such as the one used at the National Institute of Standards (NIST) USAR Test Facility, are an increasingly popular way to develop simulations. Game Engines are used by both game and simulation developers to reuse the common functionality required by a simulation – after all, a computer game is a form of simulation – to reduce the development cost [50]. The development of the model used by the simulation can be separated from the development of the engine itself, and requires subject matter expertise in the area being simulated. The OneSAF Objective System (OOS) is the U.S. Army's targeted Computer Generated Force simulation system used for training and testing across a large variety of operations. At I/ITSEC, Karr described the challenges associated with translating real world into computer models for simulation [51].

What are the Missing Pieces?

The main focus of most of the research into human/robot teaming has been how to get an existing team to achieve its common goal and how to communicate effectively between human and UMS members of the team. The focus of our research is to determine how to create an efficient, effective team, not necessarily using all available resources, and give guidance on the best path to take. Most of the research to date in

human-robot interaction has been on how to develop effective communication between the members, or how to divide tasks among the members. Very little research has been published on methodologies to choose an optimal set of members to populate that team in a consistent manner. The distinguishing factor here is that our project's goal was to optimize based on a set of disparate team members, including not only the proposed path to be taken but also the makeup of the team. In addition, our project is at the level of global path planning, since the major goal of the project is to select an optimal team, not perform the actual task.

Common sense suggests that a team should be comprised of members that have at least a subset of the skills required to do the job. For a team to complete a task successfully, the entire set of required skills must be covered among the members of the team. The question remains of how to create a team that can perform the assigned task in an optimal manner. Our research is focused on staffing a single team for a task. For future work, it could be enhanced to allow optimization across multiple tasks and multiple teams, but the goal of our research is not to assign every member to a task.

Innovation/Value Added

Heuristics-based path planning can be highly complex and highly specific to a pre-defined heuristic. When evaluating multiple combinations of team resources using purely heuristics-based algorithms, the complex heuristics algorithm must be run once for every combination of resources, leading to a very time-consuming process that is difficult to re-fit to new criteria. A simulation-based approach would estimate a result for every team going down every path and choose the best one. With high-speed computers, memory, and time, this can be done for a small sample; however, it

becomes unwieldy for a large pool of resources and a mission that can take many paths.

Somewhere between AI path planning and massive multiple-simulation using progressive refinement there is a well-balanced approach to determining an optimal team for a given task. For this hybridization to be effective, it must reduce the complexity of the heuristics portion and reduce the number of iterations required for the simulation portion. This research searches for that hybrid approach. The results of this research includes a flexible, data-driven method to determine the optimal configuration of a human-robot team for a given task and a tool that can be used in team planning, member re-tasking, and leadership training.

This method employs a hybridized approach of simulation and artificial intelligence techniques, using a progressive-refinement queuing model to quickly bypass the least desirable configurations. For a task involving movement of the team, this combination of techniques allows the user to specify paths at any fidelity, from start/end points to a series of waypoints, and skills required to perform the mission. From the points provided, the program uses modified path-planning techniques that employ some, but not all, of the variables used in the cost algorithm, to determine a set of viable paths, rather than a single, optimal path, since the exact ‘distance’ – in this case, score – is different based on the characteristics of the individual team, and the scoring method may be changed depending on the priorities of the mission. The variables, or heuristics, chosen for the path-planning algorithm were chosen because they have a similar impact on all team configurations; those that are heavily influenced by the team characteristics were left to the simulation and scoring algorithm. The resulting set of paths is then

converted to a series of queuing models that have progressively more levels of detail, and simulations are performed for all possible team configurations that fulfill the basic mission requirements. Figure 2-1 illustrates the high-level concept of team and path selection using multiple simulations.

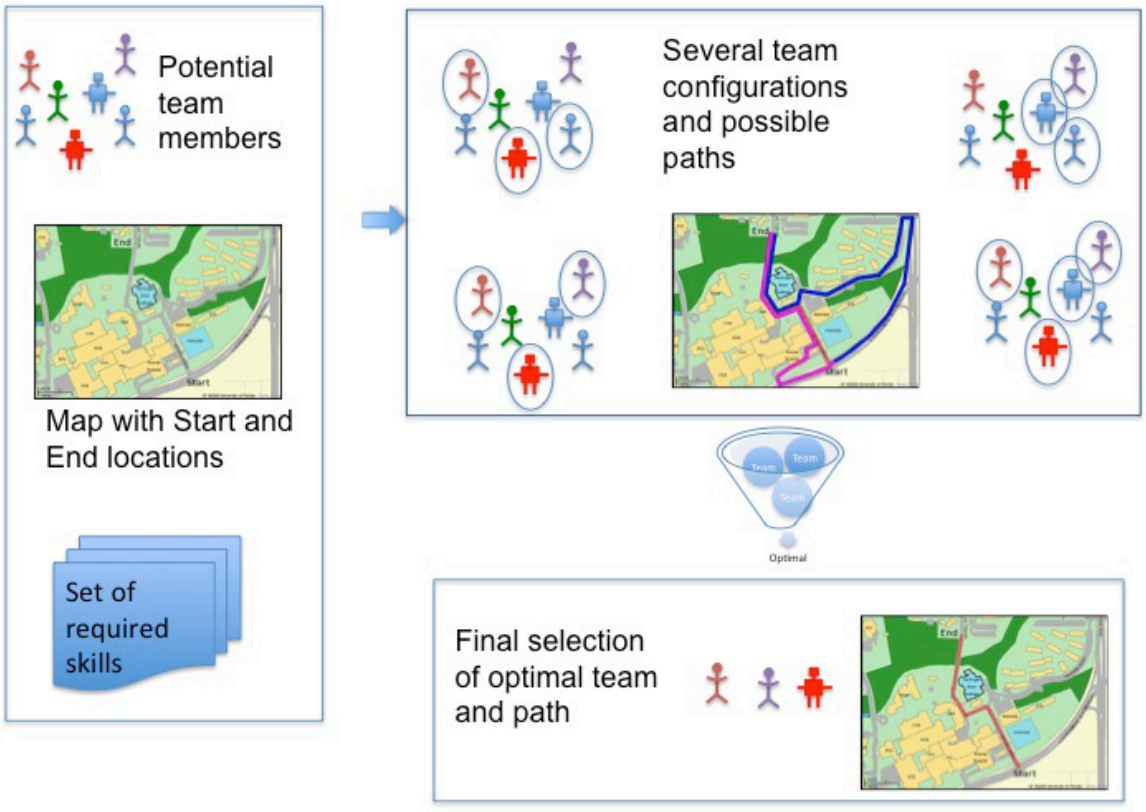

Figure 2-1. Teamwork selection starts with a pool of candidate members, a set of required skills, and a goal. Multiple simulation approach tests each viable team against each path to find the optimal team and path combination.

Contributions

Beyond a prototype that will likely sit on a shelf, for the research effort to be worthwhile, something good must come of the research. This research crosses multiple disciplines within the realm of computer science. Major classifications as designated by the Association for Computing Machinery's (ACM) Computing Classification System (CCS) covered include I.2.8 Artificial Intelligence/Heuristics, I.2.9 Artificial

Intelligence/Robotics/Workcell organization and planning, and I.6.8 Discrete Event Simulation [55]. These classifications are illustrated in Figure 2-2.

The domain of the problem space is I.2.9 Workcell organization and planning – in this case, planning the organization/structure of work groups, or teams, that contain both humans and robots, with each member having a specialized skill or set of skills that are needed to contribute to the task's goal. The first contribution from this research is in the automation of that team planning, which is aided by the designation and breakdown of skills as mission and member parameters, and operational and environmental characteristics that define how the members interact with the surroundings, such as member and passage widths, or member weights and passage weight limits. Figure 2-3 shows the specific subjects within the I.2.9 classification.

The domain of the solution space of this work encompasses both I.2.8 (AI/Heuristics) and I.6.8 (Discrete Event Simulation). These are the areas that we use to address the problem solution – how to best perform the team planning. A pure simulation approach would involve simulations of every reasonable path that every valid team configuration could take. Assuming N teams and an average of M possible paths per team, this would involve $N*M$ simulation runs, on top of the exhaustive path search, which has a complexity of $O(n!)$, where n is the number of intersections[56]. If n, N and M are relatively small, this may not be an issue; however, this solution would not scale very well. Scalability becomes an issue when there are many options from both a number of teams and number of possible paths perspective. In a very large or very complex location, the number of possible paths could become quite large – even when the paths are constrained to simple, non-cyclic paths. Traditional consumer global

positioning system (GPS) units and online mapping products attempt to stay on a main road for as long as possible, then perform more fine-tuned path planning when near the destination. This is sufficient for most recreational road travel; however, in a task such as the hostage rescue scenario, it may be undesirable to stay on a main thoroughfare for a long period of time, as this leaves the team more exposed to hostile activity. In addition, in some areas of the world, there are no larger roads to use for the initial part of the journey. With the prospect of having to send the team through a myriad of back roads and alleys through a large area, scalability becomes a factor. In addition, if the tool is used to dynamically replan, fast processing time becomes crucial.

A purely heuristic approach would require a significant number of weights to be calculated based on skills and terrain features – much of the same calculations that are involved in the simulation. Each team configuration would require its own set of weights for the path nodes, and the heuristic algorithm would still have to run *N* times. With a best-first search that runs in linear time, that might sound appealing; however, the calculations of the heuristics, which must be done on each edge *e*, must also enter into the equation. A more compelling reason to avoid the solution to run the best-first search for each team configuration is that the hybrid heuristic/simulation approach gives alternative solutions that may be almost as good, or even equally good, as we will show in chapter 4. This hybrid heuristic and progressive refinement simulation approach is the second contribution from this work, providing a faster way to reach the answer.

To reduce the computing time and complexity required to find the optimal team, this research combines the strengths of multi-simulation – faster than realtime estimate of results, ability to safely run multiple scenarios, single pass calculations - with the

strengths of a heuristic approach – quickly eliminating less desirable paths, the ability to use different weighting factors. Our goal was simple – reduce the computing time to find the best team and the most favorable path, without compromising the result. By using a heuristic approach at a higher level to reduce the number of paths followed in simulation, we were able to achieve our goal, contributing to the areas of team planning in a hybrid human/robot team, and a hybrid of heuristics-based path planning and multiple discrete event simulation.

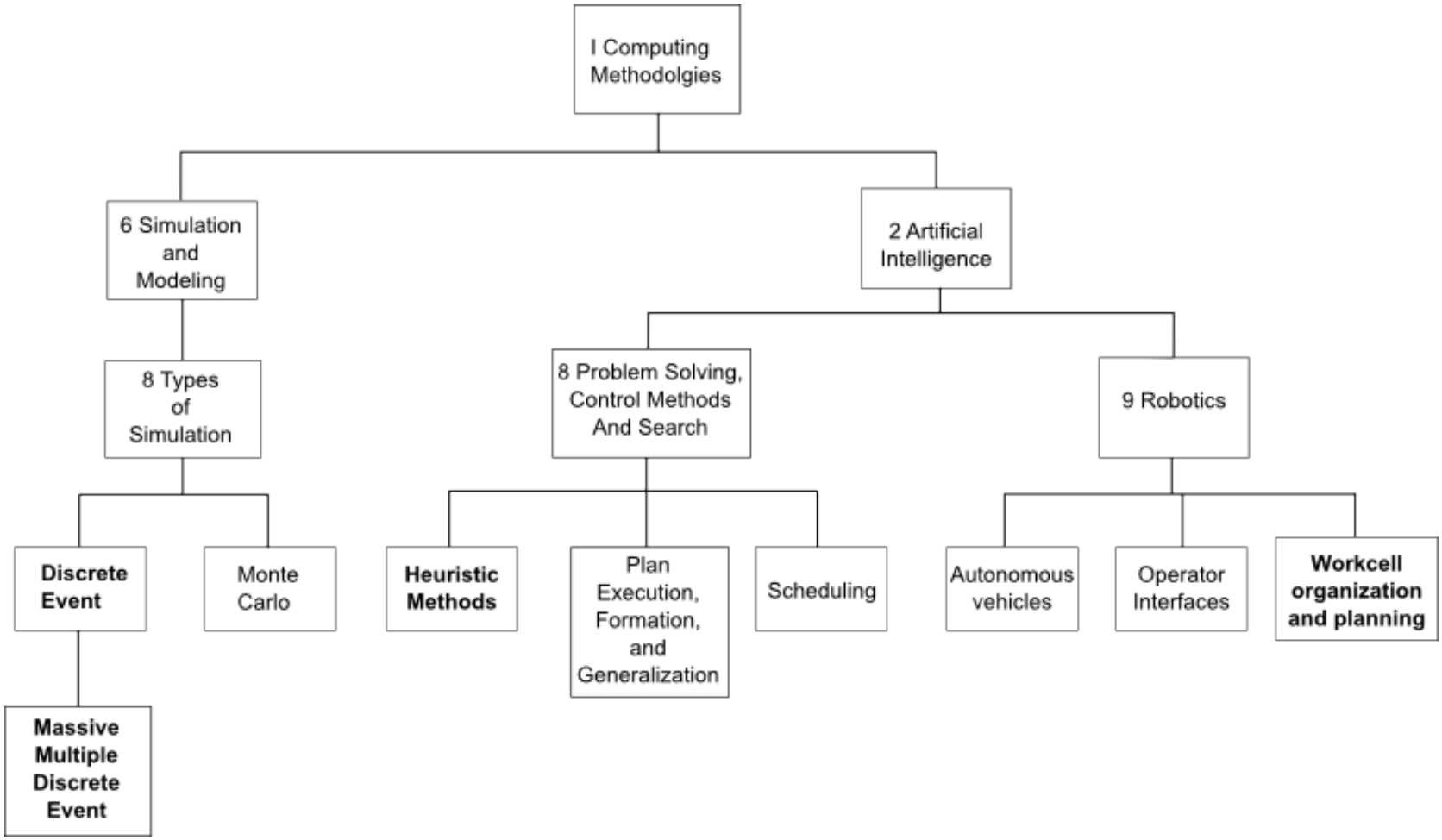

Figure 2-2. Computing Classification System Contributions

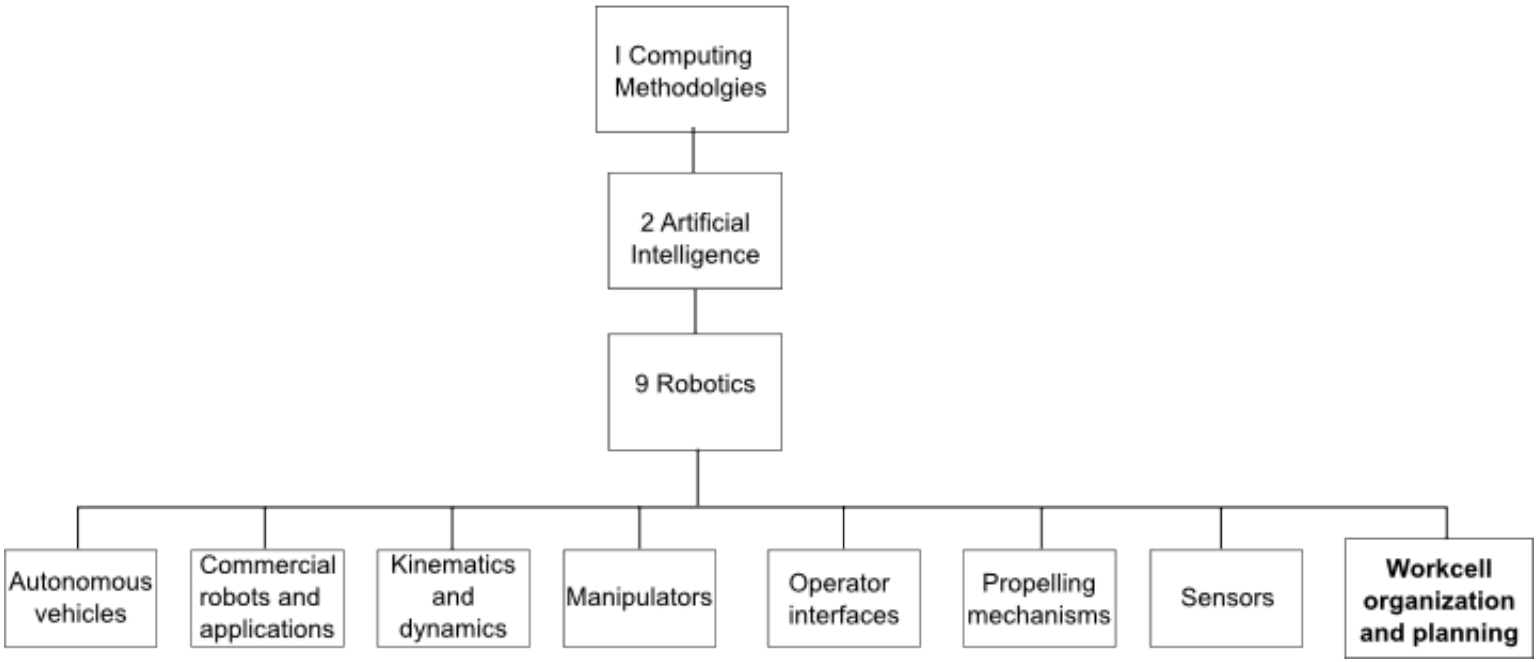

Figure 2-3. CCS Subjects of Robotics

CHAPTER 3
PROTOTYPE

The research into using simulation via progressive refinement to determine an optimal human-robot team resulted in a prototype that allows the user to define required skills, user skills, and a layout of the area in question. The prototype is designed in a modular fashion, allowing for different scoring functions, goals, and path selections to be tested. The prototype, as shown in Figure 3-1, contains an editor to create and view Resources, a mission editor to select mission properties, including skills and start/end nodes, and a team planner that performs the team and path selection and simulation.

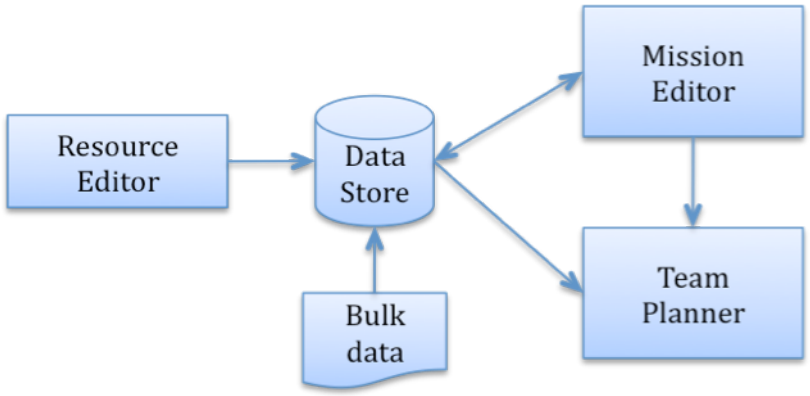

Figure 3-1. High level architecture of prototype

Ontology

The data model shown in Figure 3-2 was created as a Resource Description Framework (RDF)[58] Schema (RDFS), using Protégé, an open-source ontology editor from Stanford [59]. It displays the high-level classes that are used to represent the data. An ontology is a representation of the concepts and relationships that can be used to describe and model data specific to a particular domain. It allows the data modeler to specify object and class

properties to define the relationships, including domain and range. Having the formal definition facilitates reasoning across the data, using automated reasoners or first-order logic in the form of queries such as SPARQL or Prolog [60]. RDFS is a World Wide Web Consortium (W3C)-recommended general-purpose language for representing web-based data. It can be represented in eXtensible Markup Language (XML), which is a standard web language for representing data in a structured format [62]. XML provides a standard format for exchanging data. Namespaces within XML allow multiple RDF Schemas to be linked and referenced, providing common definitions to be shared. An XML primitive would be specified by <tag>value</tag> or the shorthand, <tag name=value />. XML instances can be grouped together into complex objects to provide a complete definition of an entity or idea. Figure 3-1 shows a portion of the XML version of the RDFS generated from the data model. Table 3-2 lists the data properties needed by Resources, Terrain, and Mission classes and is the basis of the RDFS definition.

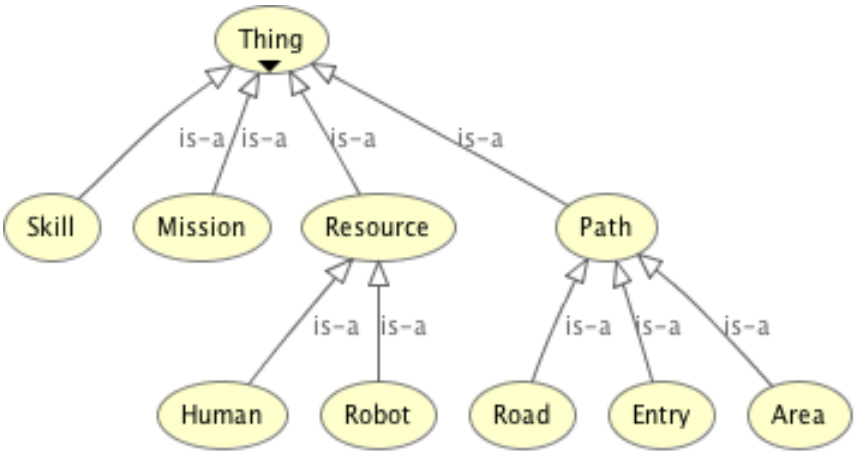

Figure 3-2. OWL class visualization

```
<owl:Class rdf:about="http://www.ufl.edu/tnieten/HRT#Resource"/>

<owl:Class rdf:about="http://www.ufl.edu/tnieten/HRT#Human">
   <rdfs:subClassOf rdf:resource="http://www.ufl.edu/tnieten/HRT#Resource"/>
</owl:Class>

<owl:Class rdf:about="http://www.ufl.edu/tnieten/HRT#Robot">
   <rdfs:subClassOf rdf:resource="http://www.ufl.edu/tnieten/HRT#Resource"/>
</owl:Class>

<owl:ObjectProperty rdf:about="http://www.ufl.edu/tnieten/HRT#hasSpeed">
   <rdfs:domain>
      <owl:Restriction>
         <owl:onProperty rdf:resource="http://www.ufl.edu/tnieten/HRT#hasSpeed"/>
         <owl:onClass rdf:resource="http://www.ufl.edu/tnieten/HRT#Resource"/>
         <owl:qualifiedCardinality
               rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
      </owl:Restriction>
   </rdfs:domain>
   <rdfs:range>
      <owl:Restriction>
         <owl:onProperty rdf:resource="http://www.ufl.edu/tnieten/HRT#hasSpeed"/>
         <owl:onClass rdf:resource="xsd#double"/>
         <owl:qualifiedCardinality
               rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
      </owl:Restriction>
   </rdfs:range>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="http://www.ufl.edu/tnieten/HRT#hasSkill">
   <rdfs:domain>
      <owl:Restriction>
         <owl:onProperty rdf:resource="http://www.ufl.edu/tnieten/HRT#hasSkill"/>
         <owl:someValuesFrom
               rdf:resource="http://www.ufl.edu/tnieten/HRT#Resource"/>
      </owl:Restriction>
   </rdfs:domain>
   <rdfs:range>
      <owl:Restriction>
          <owl:onProperty rdf:resource="http://www.ufl.edu/tnieten/HRT#hasSkill"/>
         <owl:onClass rdf:resource="http://www.ufl.edu/tnieten/HRT#Skill"/>
         <owl:minQualifiedCardinality
               rdf:datatype="&xsd;nonNegativeInteger">1</owl:minQualifiedCardinality>
      </owl:Restriction>
   </rdfs:range>
</owl:ObjectProperty>

<owl:Class rdf:about="http://www.ufl.edu/tnieten/HRT#Skill"/>
```

Figure 3-3. XML Sample

Table 3-1. Properties

Resource	Terrain	Mission
Name	Level	Skills required
Size	Name	Starting location
Weight	Location – coordinates	Goal location
Speed	Parent	Scoring function
Cost to deploy	Children	
Cost to replace	Neighbors	
Hourly rate	Danger probability	
Skills	Smallest width	
	Traversal length	
	Weight limit	

Since RDF and XML are standardized data formats, using RDF for defining the data makes conversion to and from external programs less complex, resulting in potential integration savings. In addition, there are several open source tools that allow for data visualization and manipulation in RDF, making it far easier to build a reasonable data set. The instance data is asserted as subject-predicate-object facts, or triples, in an AllegroGraph triple store [61]. Both Protégé and AllegroGraph follow the RDF standard. Storing the data as RDF triples allows the use of SPARQL to easily query the data, and to perform the initial reasoning across the facts; for instance, the set of candidate teams is generated directly from the triple store using a SPARQL query. SPARQL queries can be used to discover information about a subject, such as all information related to a Resource or set of Resources; link subjects together by common objects, such as retrieving all Resources who have a specific skill or fit into a given weight range; find information by predicate, such as retrieving a list of all Resources and their associated skills; or by any combination of linked or nested subjects, predicates, or objects.

Framework

Resource Parameters

For a team to meet its goals all members have at least one skill that is required to complete the specified task; in addition, all members have certain operational characteristics that affect their ability to perform the mission or the overall “cost” of the mission. For instance, speed is used as a factor in computing the estimated time required for the mission; weight and width are used in conjunction with weight limits and widths of passages in the terrain to determine if a given team can go down the path. For the purposes of this discussion, “cost” could be monetary or temporal, or even projected loss. In the prototype, the skills are used to match resources to the current mission, and operational characteristics are used to calculate navigability and overall score of the mission. Figure 3-5 shows the classes created for building the team. The CreateResourceBean and ResourceBean classes are used by the Resource Editor to define and store resources. A Team consists of one or more Resources, and a Resource contains one or more Skills.

For the initial prototype, we used a small set of skills for each potential member of the team, and subsequently added more skills and operational characteristics. The prototype is designed to allow skills to be added both to the resource pool and to the mission goals as needed, rather than having a static, predefined set of skills. Each available resource is also assigned physical and operational characteristics, including size, maximum speed, and weight, using the Resource Editor shown in Figure 3-4. For some scoring functions, cost to deploy and cost to replace are also considered. Since operational characteristics

are an integral part of the scoring function, they are not as dynamic as skills.

Table 3-2 lists a set of sample parameters that are used in the prototype.

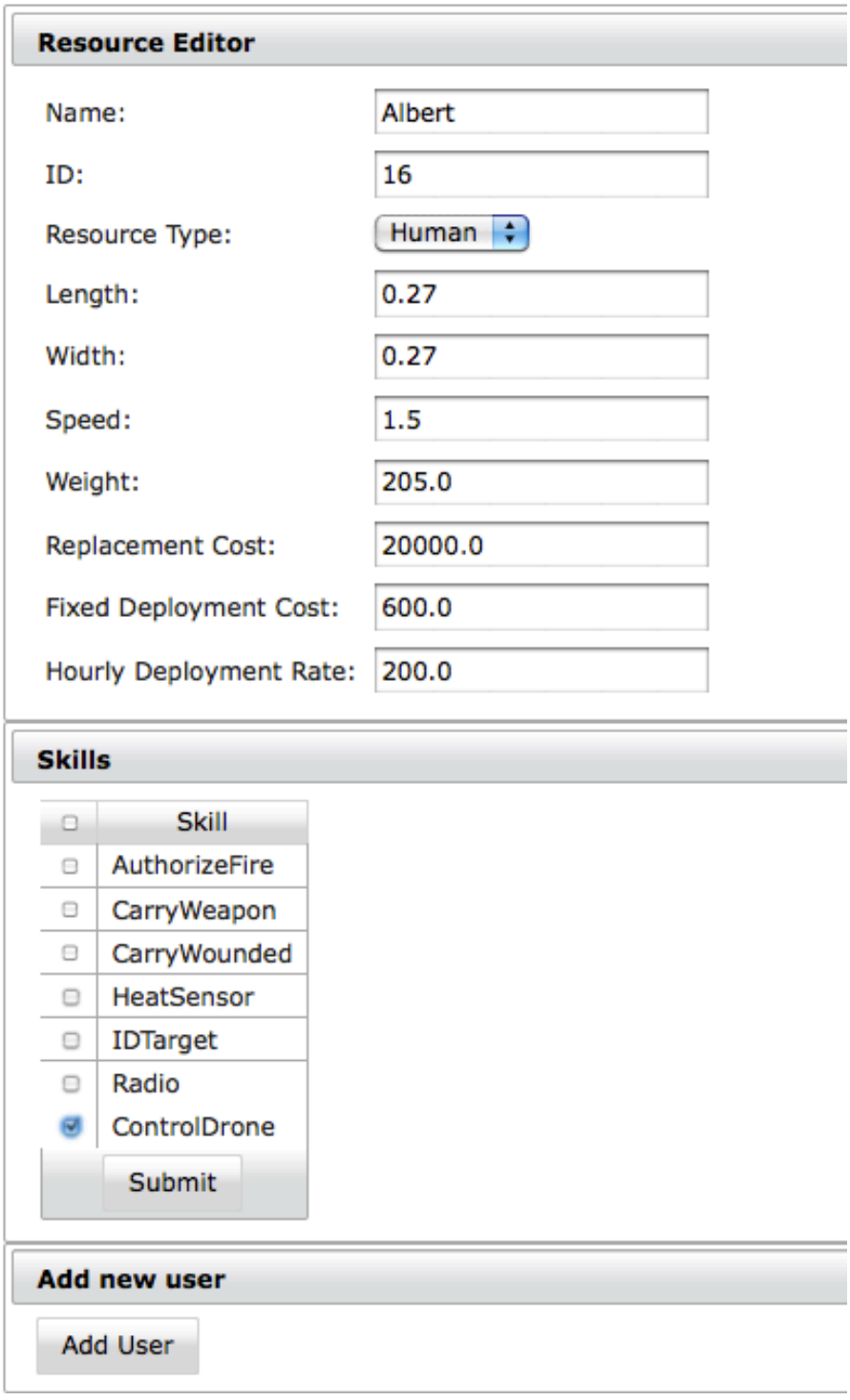

Figure 3-4. Entry panel for resource candidate

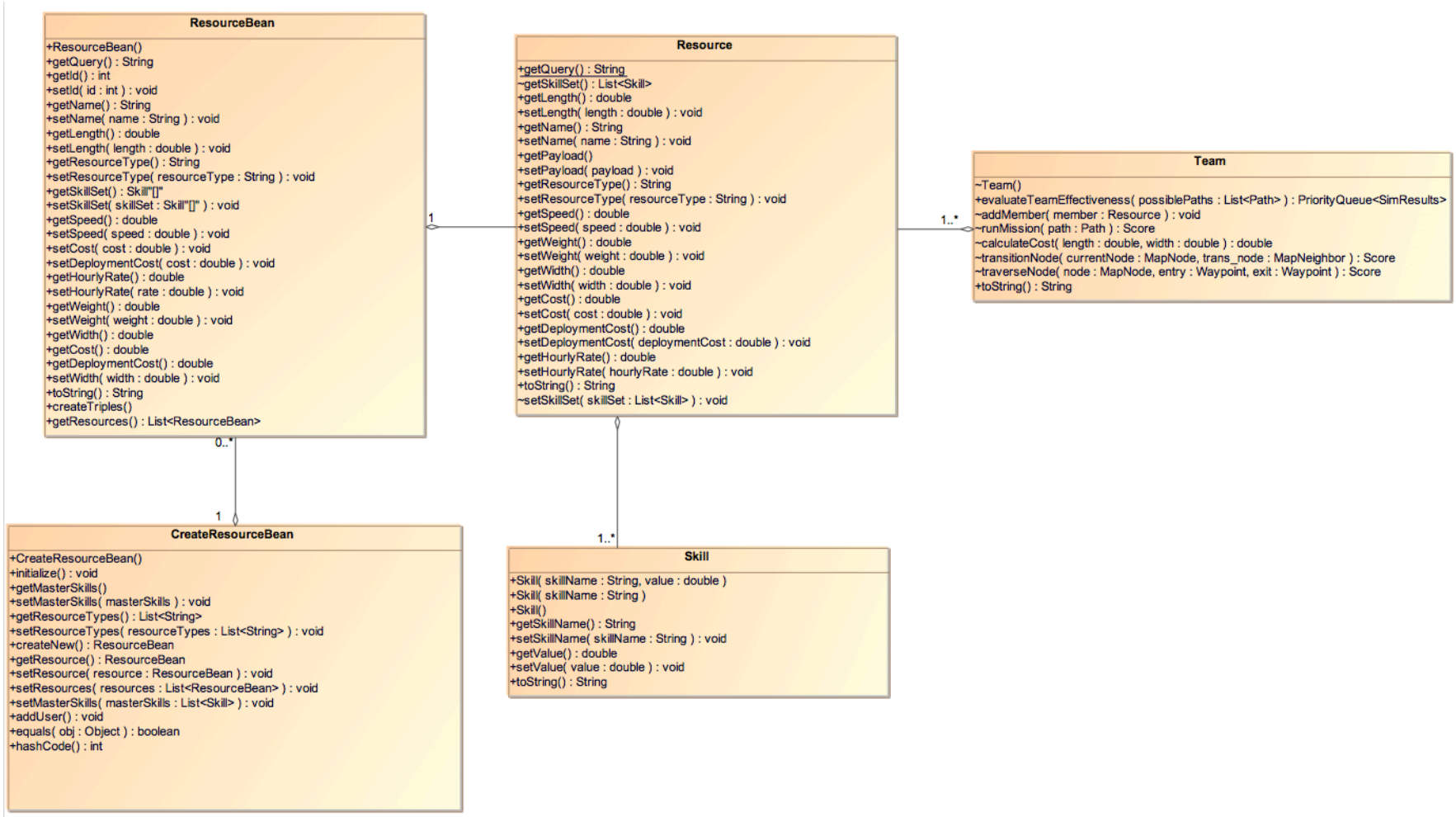

Figure 3-5. Class diagram for Resources

Table 3-2. Resource Parameters

Classification	Resource Parameters
Operational Characteristics	Width
	Weight
	Speed
	Length
	Cost to deploy
	Cost to replace
	Hourly cost to deploy
Skills	Dynamic list

The set of all possible teams is generated at the beginning of the simulation using a dynamically built SPARQL query, based on mission requirements. Figure 3-6 shows a sample query for a mission with 4 defined skills, displayed in AllegroGraph's AGWebview tool. Only a subset of the results is displayed since the query returns 120 results. In the cases where a team member has multiple skills, the current implementation assumes the member can perform all of the duties; thus a result of "Barney, Cylon, M1, and Barney" will yield a 3-member team. Future enhancements to the tool could force all members to be unique, i.e., use only one of their defined skills per mission, to allow the definition of exclusive skills to allow some overlap for skills such as carry radio and identify target, but exclude some skills such as carry weapon and carry wounded, or to allow a minimum number of members with a given skill. To enforce unique members, a FILTER could be applied to the SPARQL query in the form of FILTER (?idx != ?idy) for each pair (x,y) of resources. To enforce multiple unique members with the same skill, a FILTER could be applied as above on ids with the same skill defined. If uniqueness were already being enforced, the SPARQL query would simply have multiple entries for the same skill.

Edit query

show namespaces

```
PREFIX hrt: <http://www.ufl.edu/tnieten/HRT#>
SELECT distinct ?name1 ?name2 ?name3 ?name4
{
   ?id1 hrt:hasName ?name1;
        hrt:hasSkill ?skill1. FILTER(?skill1 = "AuthorizeFire")
   ?id2 hrt:hasName ?name2;
        hrt:hasSkill ?skill2. FILTER(?skill2 = "CarryWeapon")
   ?id3 hrt:hasName ?name3;
        hrt:hasSkill ?skill3. FILTER(?skill3 = "CarryWounded")
   ?id4 hrt:hasName ?name4;
        hrt:hasSkill ?skill4. FILTER(?skill4 = "ControlDrone")
}
```

Execute

Result

?name1	?name2	?name3	?name4
"Barney"	"Cylon"	"M1"	"Barney"
"Barney"	"Cylon"	"M1"	"Fred"
"Barney"	"Cylon"	"M1"	"H2"
"Barney"	"Cylon"	"M1"	"T2"
"Barney"	"Cylon"	"Mule"	"Barney"
"Barney"	"Cylon"	"Mule"	"Fred"
"Barney"	"Cylon"	"Mule"	"H2"
"Barney"	"Cylon"	"Mule"	"T2"
"Barney"	"Mitchell"	"M1"	"Barney"
"Barney"	"Mitchell"	"M1"	"Fred"
"Barney"	"Mitchell"	"M1"	"H2"
"Barney"	"Mitchell"	"M1"	"T2"
"Barney"	"Mitchell"	"Mule"	"Barney"
"Barney"	"Mitchell"	"Mule"	"Fred"
"Barney"	"Mitchell"	"Mule"	"H2"
"Barney"	"Mitchell"	"Mule"	"T2"
"Barney"	"Viper"	"M1"	"Barney"
"Barney"	"Viper"	"M1"	"Fred"
"Barney"	"Viper"	"M1"	"H2"
"Barney"	"Viper"	"M1"	"T2"
"Barney"	"Viper"	"Mule"	"Barney"
"Barney"	"Viper"	"Mule"	"Fred"
"Barney"	"Viper"	"Mule"	"H2"
"Barney"	"Viper"	"Mule"	"T2"

Figure 3-6. SPARQL query for team selection

Area of Interest

Figure 3-7 shows a human-readable map-based display of the area, much like the area depicted in Figure 3-8. While this is easy for a human to identify paths and features, it is virtually useless to a computer in that form. The pertinent "facts" about each feature are translated into RDF triples. At runtime, those "facts" are translated into a traversable graph for path planning and simulation.

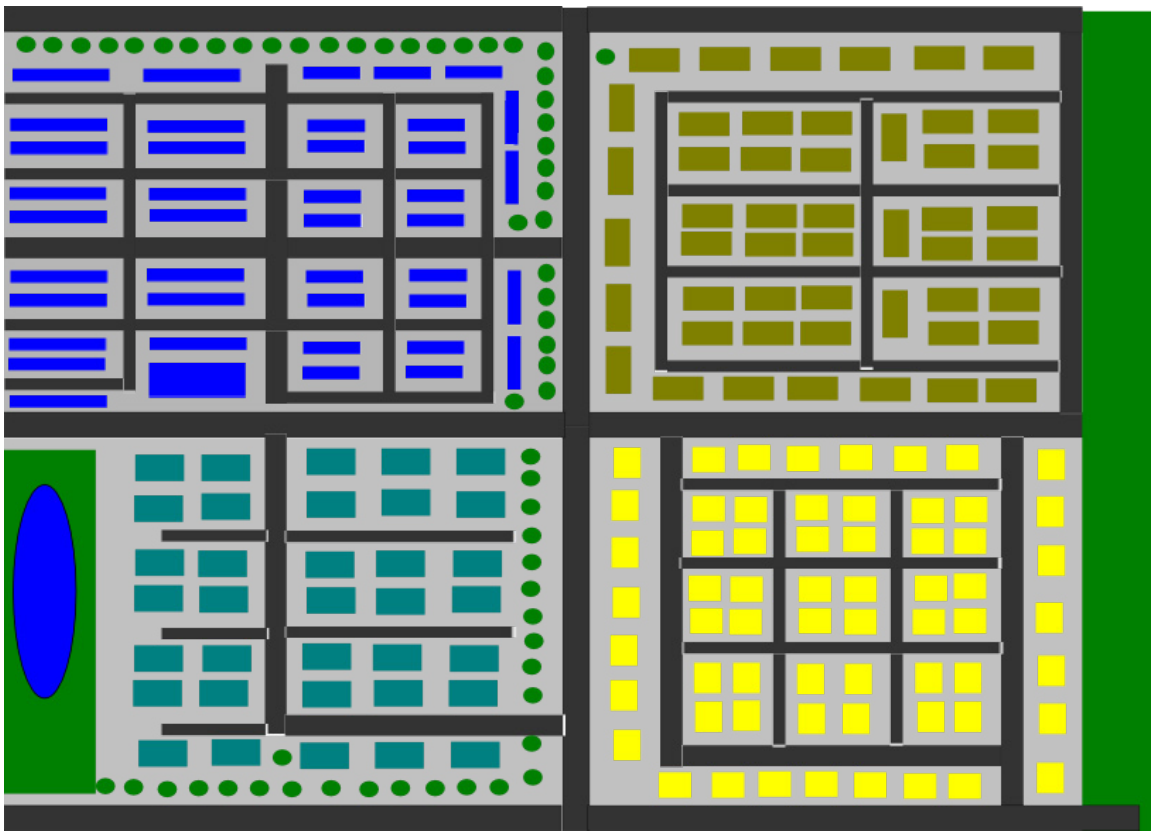

Figure 3-7. Diagram of a city area. The main City area is divided into 4 large Blocks, NW, NE, SW, SE, with major roads to the North, South, and crossing the center. Each block is divided into Sections with roads dividing the sections. Each section is further divided into Areas with smaller roads. Each Area contains houses, apartment buildings, or other buildings.

Figure 3-8. Search team in an urban area (courtesy U.S. Marines)

The terrain layout is defined as an RDF graph, using a set of RDF triples to model the individual map features, such as buildings, roads, and rooms, as well as transition points between the features. Each node in the RDF graph is represented by a *subject* in the RDF graph; features of a node are represented by *predicates*, and the individual characteristics of that node are specified in the *object* portion of the RDF triple. Figure 3-9 shows the node "NW_Area1" as it appears an RDF graph. The same node is displayed as SPARQL output as the Subject and Object of individual RDF triples in Figure 3-10. The description of the area itself occurs when "NW_Area1" is in the subject. Associations with other nodes identified by "NW_Area1" being in the object. The layout can then be displayed as an RDF graph with paths between the neighboring nodes, as displayed in Figures 3-12 and 3-13.

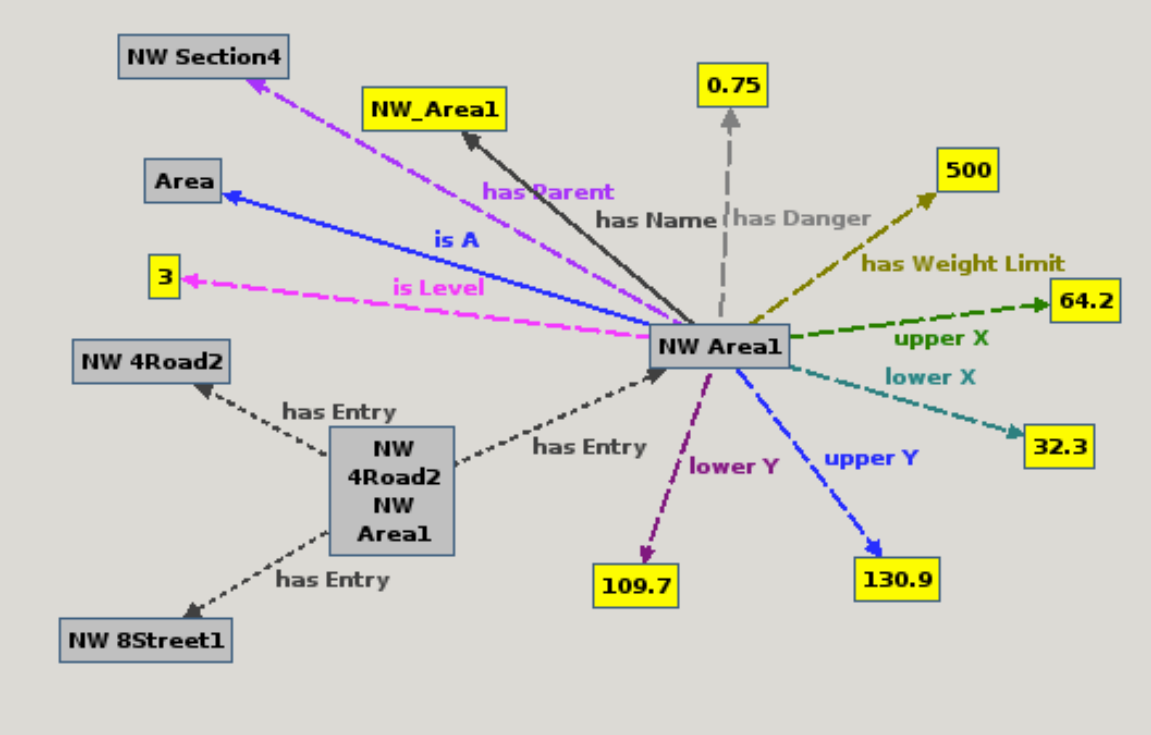

Figure 3-9. RDF relations of a named location

?p	?o
hasDanger	"0.75"
hasName	"NW_Area1"
upperY	"130.9"
upperX	"64.2"
lowerY	"109.7"
lowerX	"32.3"
isA	Area
hasParent	NW_Section4
hasWeightLimit	"500"
hasType	"1"
isLevel	"3"

Figure 3-10. SPARQL results for subject NW_Area1

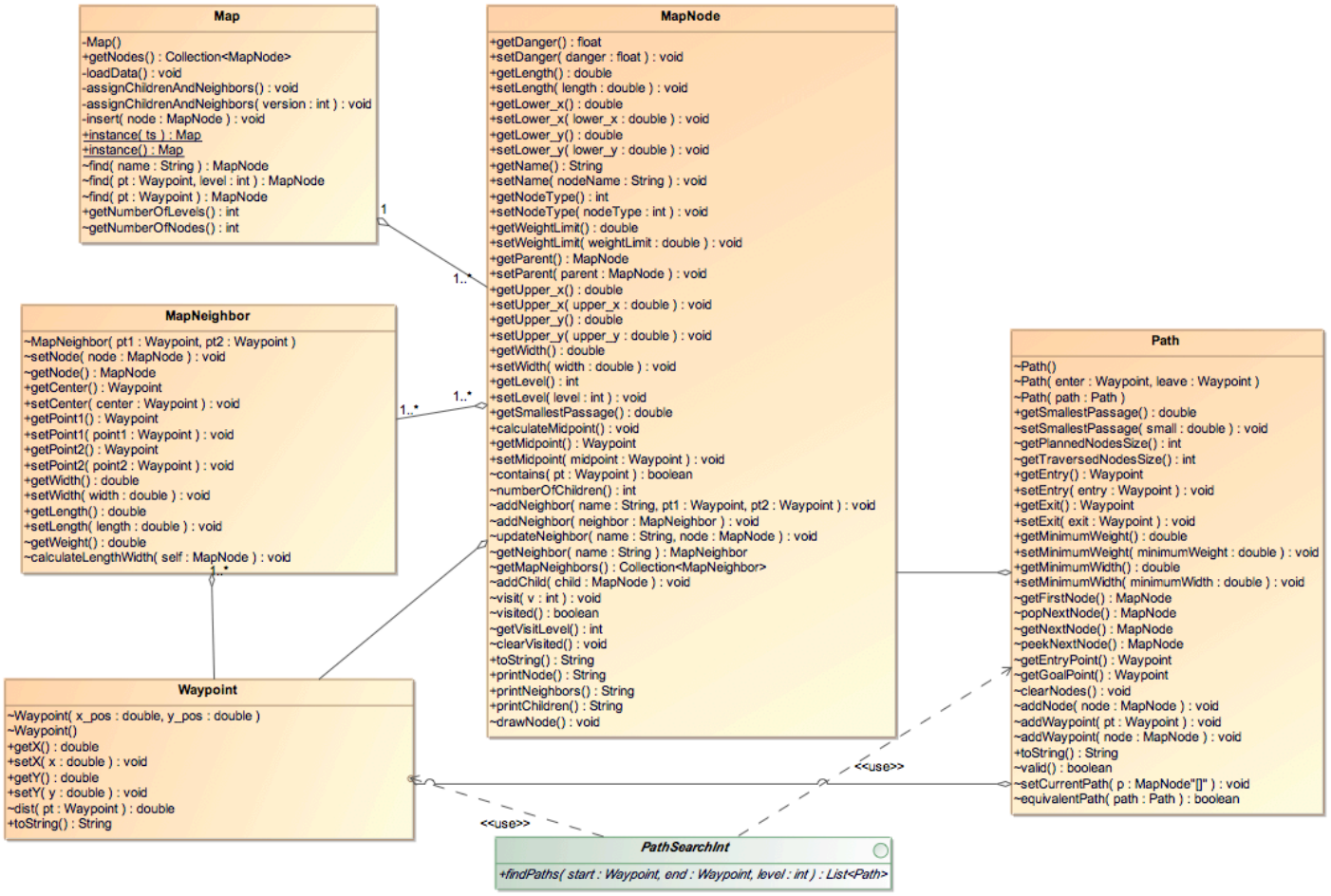

Figure 3-11. Class diagram for map-related classes

Once the scenario has started, the map data from the triplestore is loaded into a set of classes to be used by the simulation and path selection. The classes related to the terrain data are displayed in Figure 3-11. The Map singleton contains a set of MapNodes which represent each map feature, including dimensions, coordinates, and neighbors. Once the initial path selection is complete, a Path contains a series of MapNodes and corresponding Waypoints used to generate the path at each level.

Initial Path Selection

Once the set of candidate teams is created using skill matching, a set of paths must be generated. The mission is defined with starting and ending locations. The initial set of paths is determined based on the settings of the heuristic path-planning module. At initialization time, a SPARQL query retrieves all leaf-node Entry/Path item (i.e., road, block, area) pairs. The query results are used to create a traversable graph, with the Entry being a graph node and the path items being edges on the bidirectional graph. Figure 3-12 gives an RDF graph representation of this data; the resulting traversable graph is shown in Figure 3-14. The heuristics-based search function discovers a reduced set of candidate paths using the traversable graph and makes it available to the simulation portion of the program. The classes involved in the search are shown in Figure 3-15.

Several well-known path-planning algorithms were evaluated prior to implementation of the path selection portion of the prototype. One of the most common algorithms is Dijkstra's algorithm. In the variation of Dijkstra's that looks for the shortest path from *source* to *target*, as opposed to finding the shortest

path from *source* to all other nodes, the algorithm finds the single shortest path [63]. A suggested variation that finds a set of near-optimal paths removes a single node from the solution and recalculates without that node; the logistics of performing the removal and replan were deemed too cumbersome for the prototype. In addition to Dijkstra's algorithm, A* also finds the shortest path from *source* to *target*, using an admissible heuristic that is based on a distance that is likely not obtainable. A traditional A* shortest path algorithm was also eliminated from consideration for the same reason as Dijkstra's – the goal of this research was to stop at a near-optimal subset of paths and simulate the rest. However, the consideration of the admissible heuristic was adapted into our search algorithm [64].

Breadth first search was another consideration for the path-planning algorithm, since it could be expanded reasonably to continue searching beyond the optimal path to explore near-optimal paths as well. The depth-limited depth-first search was chosen above this algorithm for a number of reasons. First, the distance of the path is not directly dependent on the number of nodes traversed in the graph; so a path with fewer nodes, due to length or danger ratings, could in fact be more expensive than a path that travels more nodes. Second, the depth-first search has a higher space complexity than depth-first [65]. Other approaches including Bellman-Ford and Floyd-Warshall algorithms were eliminated because they are of most use in negative-weight graphs, which was not necessary for this problem.

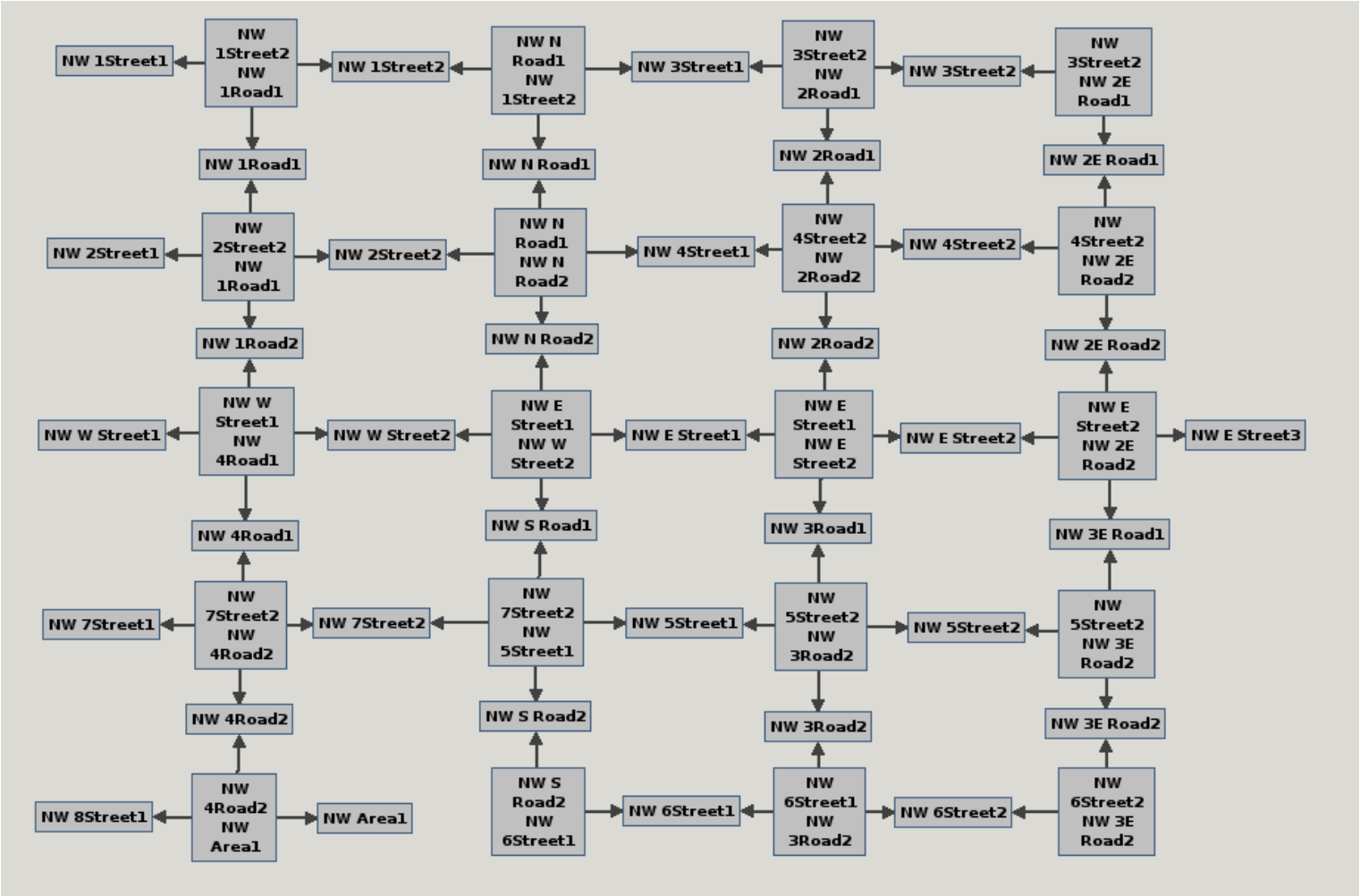

Figure 3-12. RDF graph of NW Block, leaf level. Larger squares are transitions, represented by the Entry class, between the roads or areas.

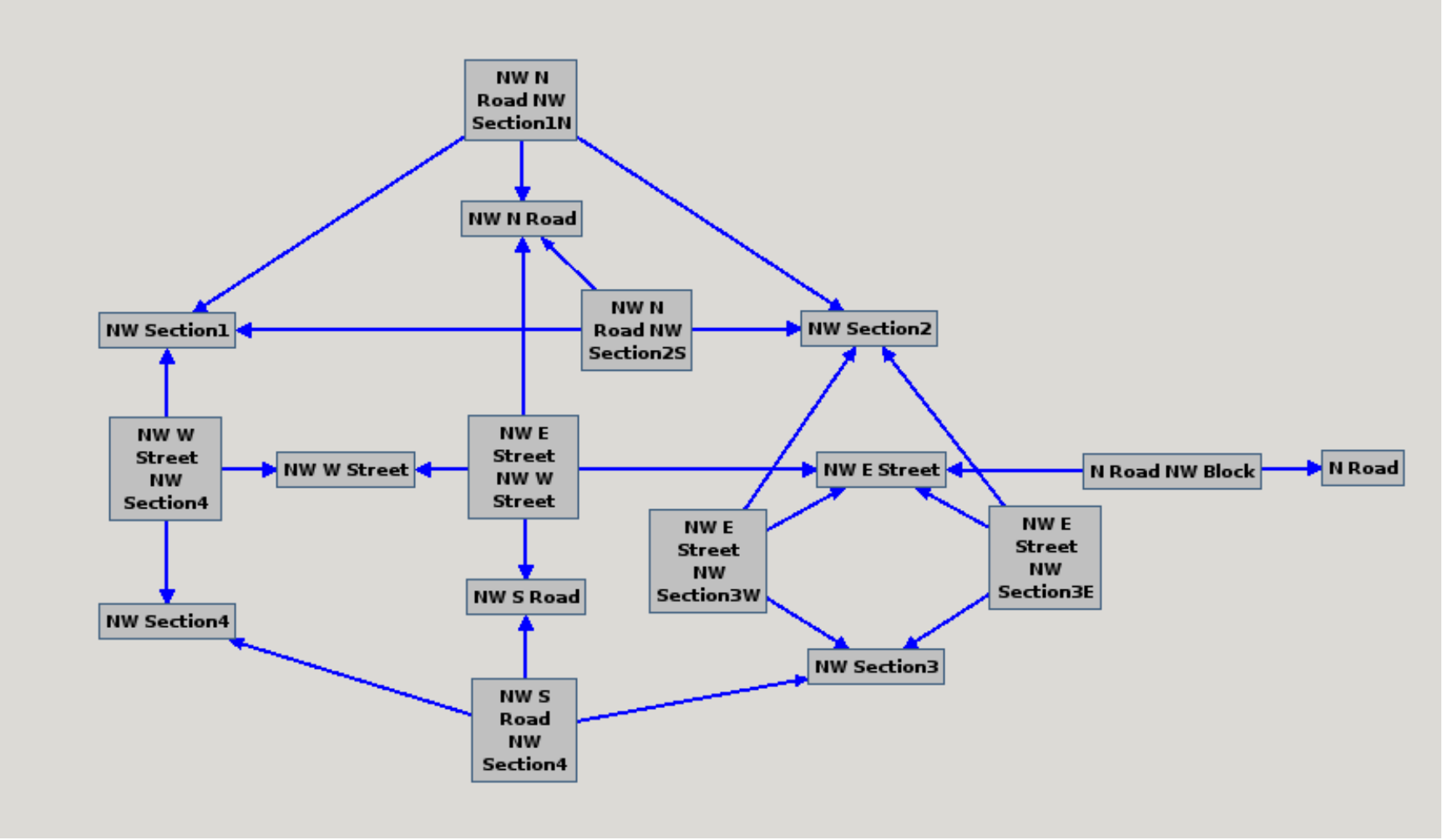

Figure 3-13. RDF graph of NW Block, level 2.

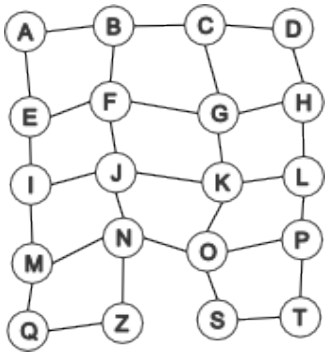

Figure 3-14. Traversal graph for path planning, derived from Figure 3-11. In this graph, the transition between two map nodes is a circular graph node; the edges are the streets in the terrain, with the length of the segment as the basis for the cost of traversing that edge.

The path search function is a modified depth-limited depth-first search. Several configurations of this search were tested, from no heuristics, which finds all possible paths between the two graph nodes, and a pair of cost-limited searches with varying limits that find a set of the shortest paths capped by the weighted distance between start and goal. The classes involved in the path search are shown in Figure 3-15. The search interface allows for replacing search algorithms.

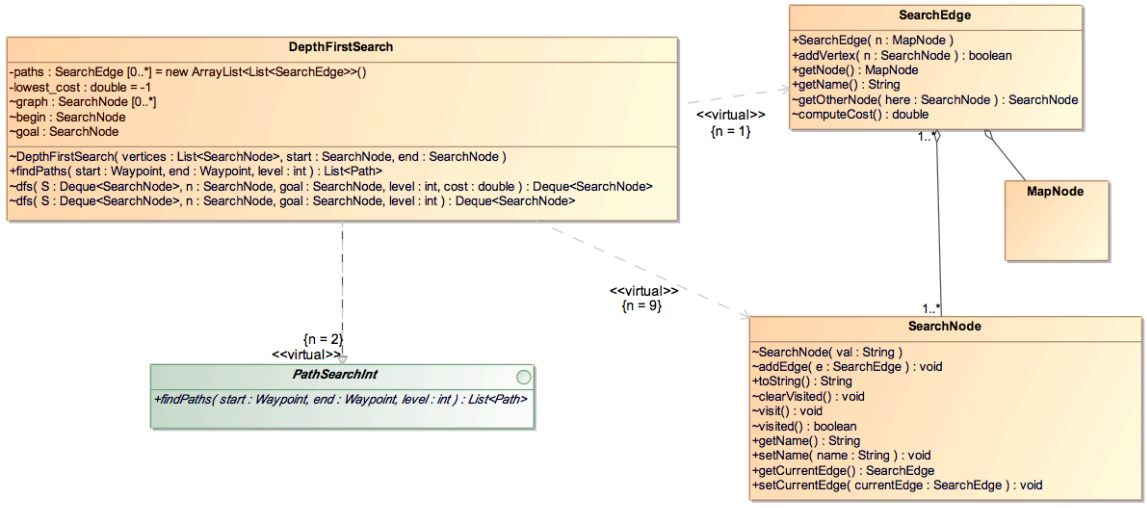

Figure 3-15. Class diagram for path selection

For the heuristics-enhanced searches, the center waypoints of the start and goal nodes are used to calculate the floor length, in distance, between the two nodes. This “as the crow flies” distance is guaranteed to be less than or equal to the shortest path. In this case, the shortest path refers to the travel distance between the two points, without regard to the number of graph nodes involved in the path. The “crow flies” distance is then multiplied by a limiting factor that places a ceiling length on the search – once a path reaches that threshold without reaching the goal, that path is abandoned. Factors tested were 3 and 4 times the absolute distance, but could vary depending on the complexity of the terrain and whether there are localized “hot spots” of dangerous activity. A more even terrain with fewer navigational limitations would be better served by having a lower threshold, closer to 2.5 - 3 times the absolute distance. One of the initial goals of this project was to find an optimal threshold between the heuristic planning and the progressive refinement; through the course of the project, however, we determined that there is no single best answer to that question but instead a guideline on how to set those heuristics and what factors to consider.

During the recursive depth-first search, the length of each edge is added to the length of the path traversed to that point; if the new length is greater than the ceiling length, that edge is skipped. This heuristic is sufficient in an area where dangerous conditions, such as possible debris in a search and rescue operation after a natural disaster or a hostage extraction operation where there is terrorist or gang activity in certain areas and the danger probability is high; or in an area with varied terrain that affects navigation, as used in the EyeRobot project [39].

Since the danger probability is a factor in the scenario we used, the length of the edge is multiplied by the danger probability. The more dangerous a passage, the higher the calculated cost of that segment of the path is. In a scenario that involves varied terrain as opposed to avoiding hostile activity, a navigability factor could be used instead. The algorithm is displayed in sequence diagrams in Figure 3-16 and 3-17. The algorithm used for heuristic path planning is:

```
heuristicSearch(start, goal)
    determine floor and ceiling costs
    search(queue, start, goal, 0)
    Generate paths from results
search(queue, node, goal, cost)
    mark node visited
    node → queue
    for each edge on node
        if edge cost + path cost < ceiling
            apply edge cost to path cost
            retrieve connected_node
            if connected_node is goal
                reverse path, place in paths list
                clear visited
                return
            search(queue, connected_node, goal, cost)
            node ← queue
            remove node cost from path cost
            clear node visited flag
```

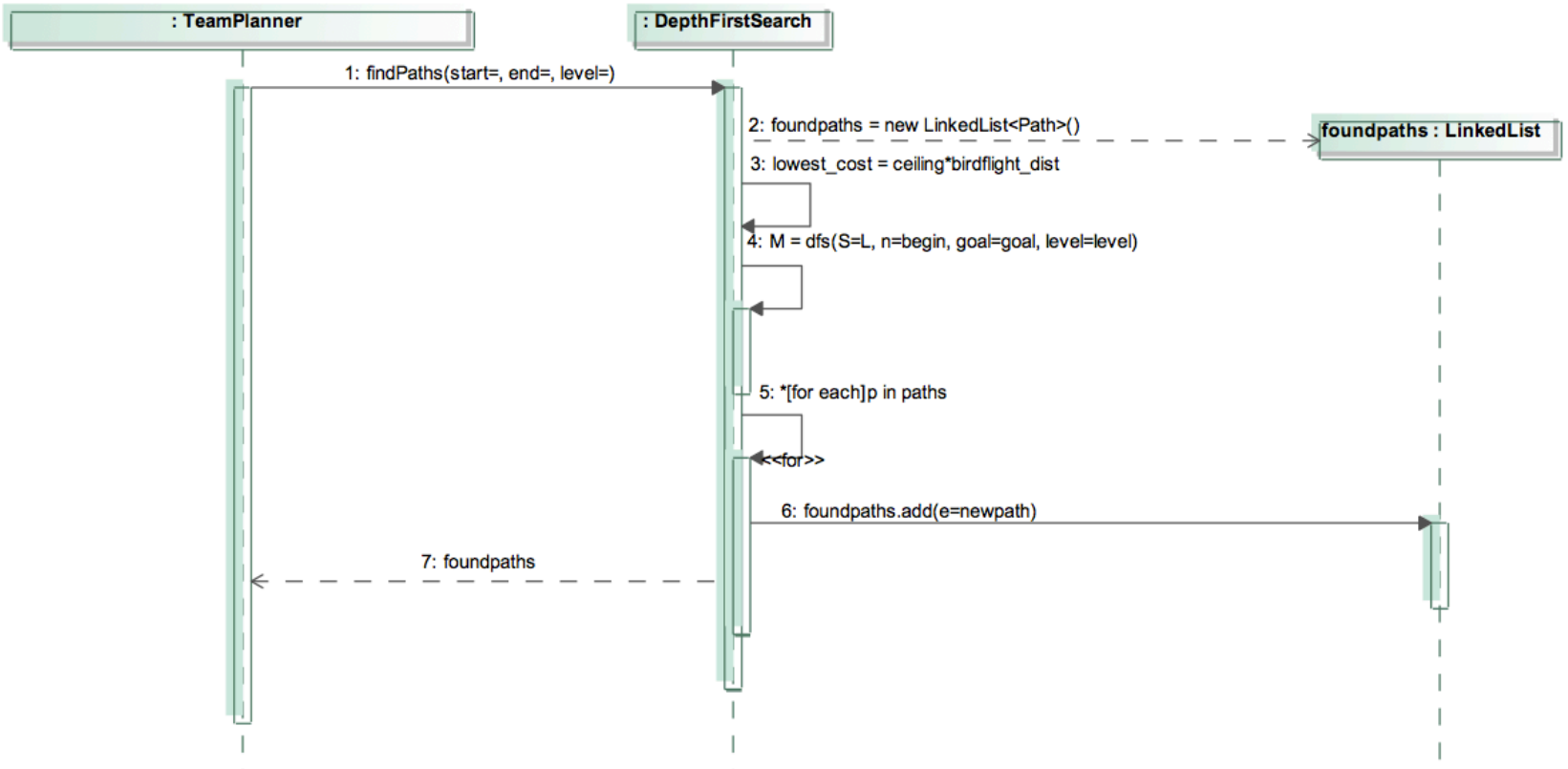

Figure 3-16. Sequence Diagram for path search initializes the heuristic cap, initiates the search, and processes the paths at the end.

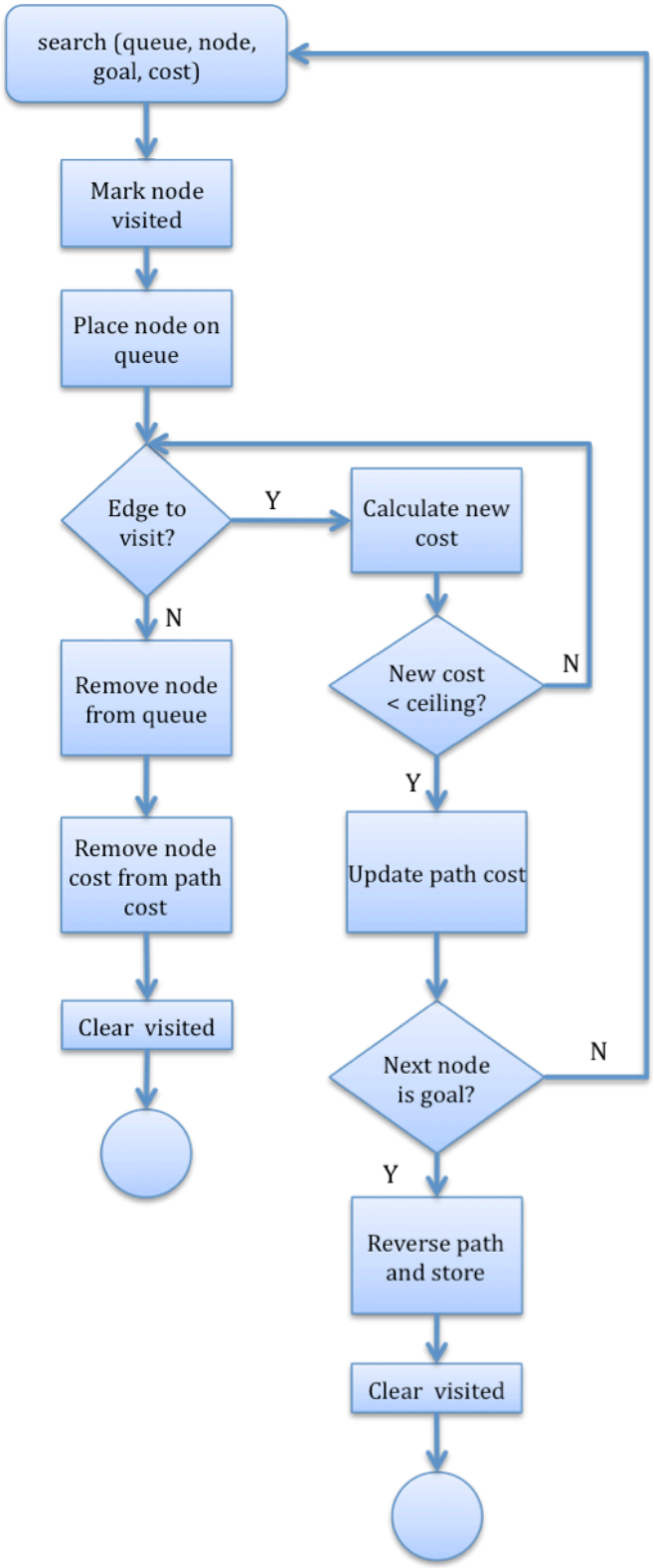

Figure 3-17. Flow diagram for capped depth first search. Cost is calculated at each node and the path is rejected if it goes beyond that cost.

Progressive Refinement

The layout of the city is stored in a set of scene graphs. Scene graphs are often used in graphics to drill down into a scene from generic/high-level overview to more and more detailed lower level. For instance, drilling down into a city block yields exterior buildings and streets, then interior buildings with room layout, to room detail. Each node of the graph, which is arranged in tree form, contains a specific feature. A node's children are parts of that node in higher detail. For instance, a building node's children would be rooms within that building [63]. This allows all operations on a node to propagate to its children and thus reduce computation [64]. A partial scene graph, as viewed through an RDF graph, of the urban layout from Figure 3-7 is depicted in Figure 3-18. The scene graph nodes are created in the MapNode class, which is detailed in Figure 3-11. The overall progressive refinement and simulation functionality is initialized and contained within the TeamPlanner and Team classes, which are displayed in Figure 3-19.

We used scene graph technology to allow the simulation to start with higher-level estimations and iterate into more and more detailed estimations. The lower the level, the more accurate the estimation should be. The depth of the simulation depends on the amount of time available to run simulations; a team built well in advance of a mission will have more time to simulate to the lowest level, but a mission needing immediate tasking may not be able to go as deep. In addition, some details may not be known for some features of the city, so the simulation must stop at the higher level for those areas, thus using less accurate scoring. As each level completes, the estimates for team performance are sorted

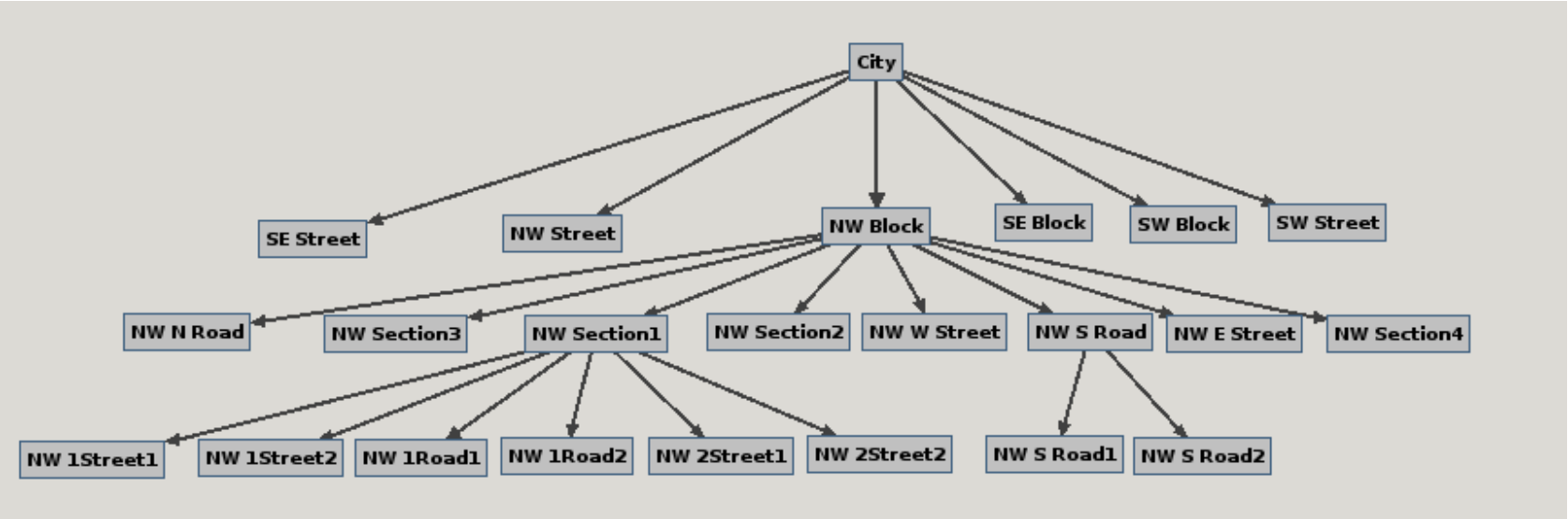

Figure 3-18. Partial scene graph layout of area. City is level 0, blocks are level 1, sections are level 2, and smaller sections of roads are level 3. The RDF graphs relating levels 2 and 3 are displayed in Figures 3-10 and 3-11.

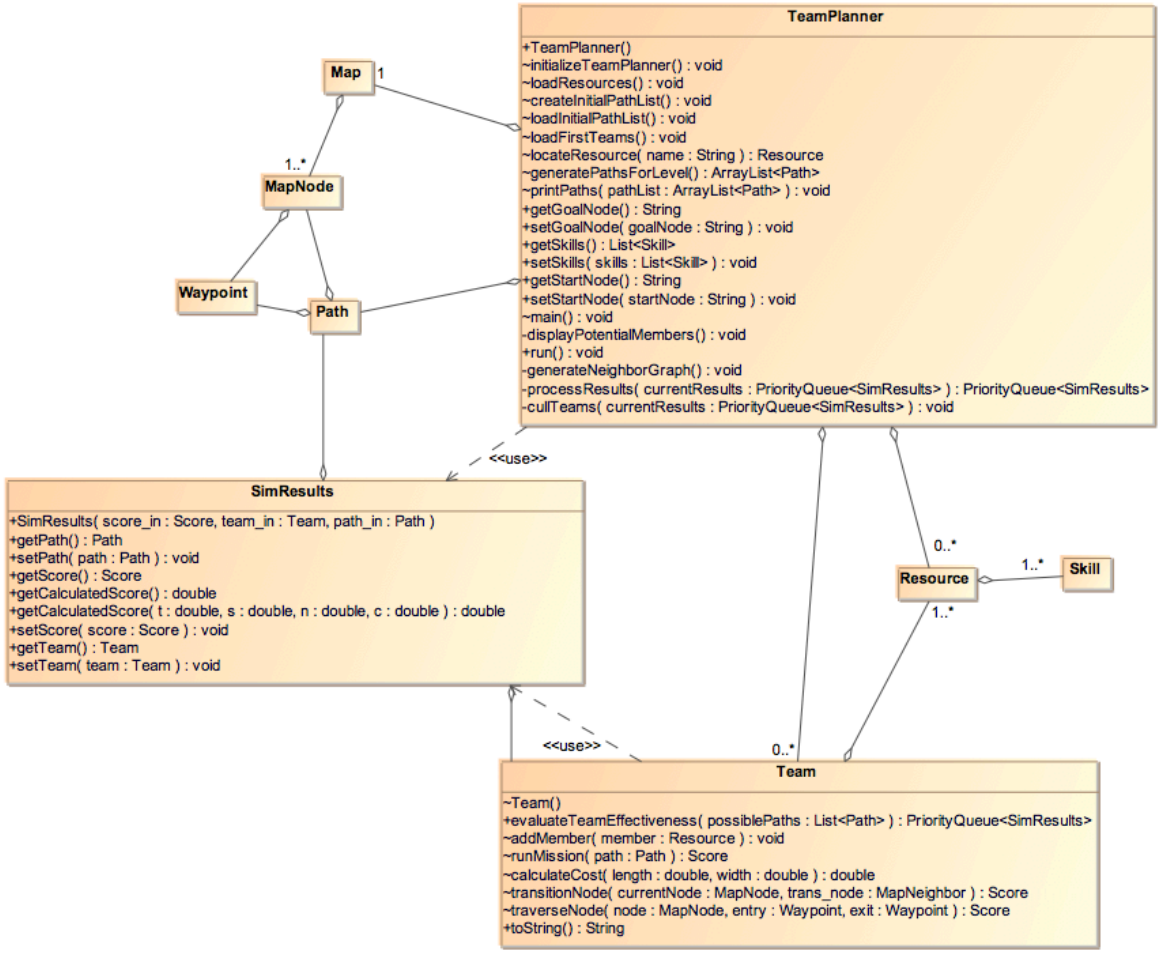

Figure 3-19. Class diagram for TeamPlanner and Team classes. TeamPlanner initializes the simulation and runs the progressive refinement. Team contains the team members and simulates each path for that team. Results are stored in SimResults.

and a configurable percentage of the teams deemed to be the most optimal are carried into the next level. This allows the number of team/path combinations to be reduced at lower levels. In a fairly shallow area layout with few levels, the progressive refinement can add overhead; however, a more complex layout can benefit significantly from the progressive refinement approach. The sequence of events from a scene graph perspective is depicted in Figure 3-20.

Simulation

Each level of the simulation starts with a required path or set of path choices for the mission at that level. This could be a detailed set of waypoints, a set of high-level waypoints, or even a start and end waypoint, depending on the mission definition and the results of the previous level's simulation. Before simulating any team configurations, the program uses those waypoints to determine a set of feasible paths to the target position, starting with the paths selected by the heuristic approach described above at the first level and a subset of those paths as the lower levels are simulated, based on the scores from previous, higher levels; this reduces the overhead of finding all of the potential paths for each team configuration. In addition, information is stored with each path that allows a quick check of whether a team member can go down that path. If a path has a narrow alley that team members over a certain dimension cannot pass through, this automatically eliminates those paths from any team configuration containing members over that size – this constraint applies primarily to UMS members, though it is feasible for a path to exist that is too small for a human member. Weight limits and member weights may also be used to constrain paths, since some unmanned vehicles can be much heavier

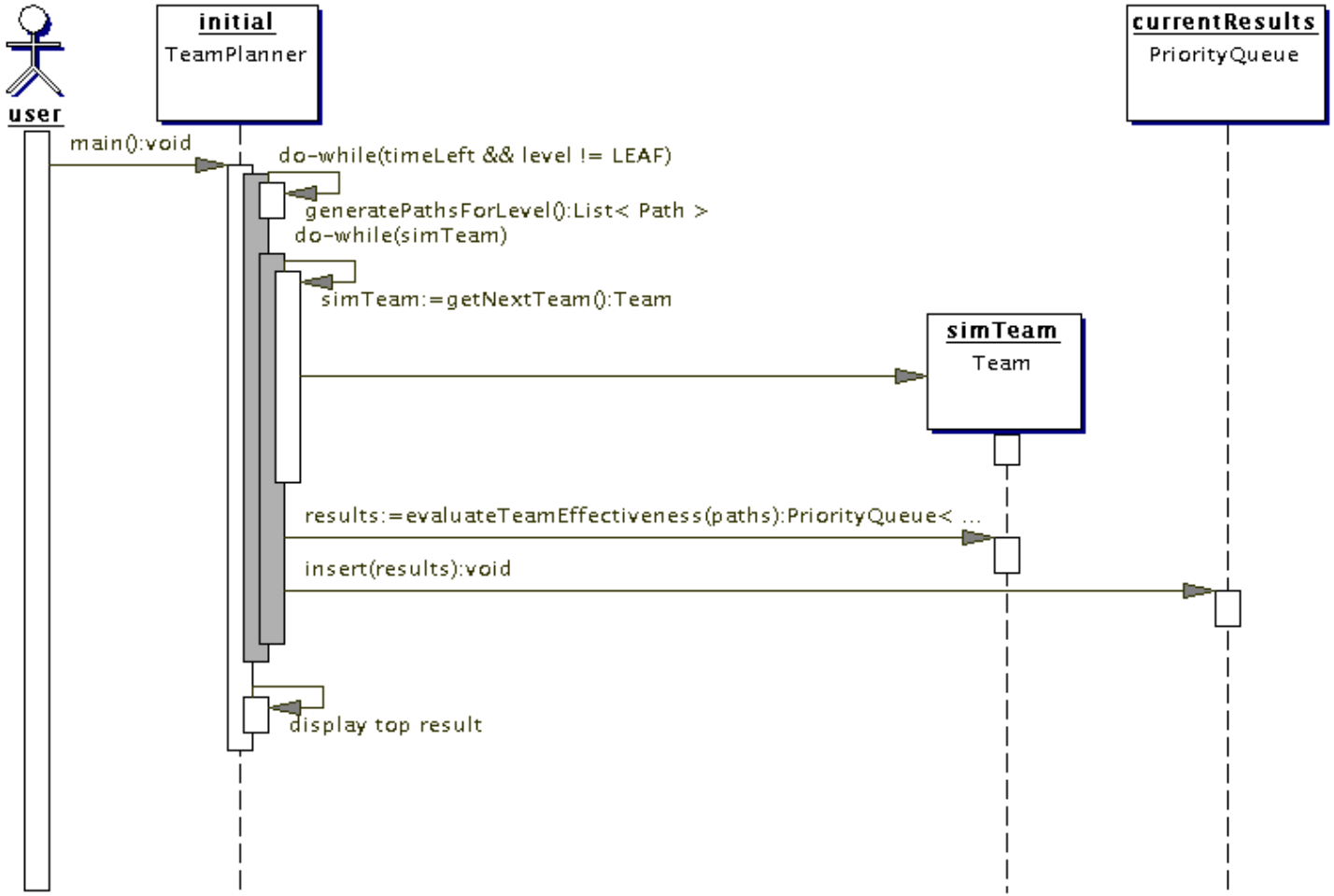

Figure 3-20. Sequence diagram of the progressive simulation

than their human counterparts. Unknowns, such as enemy presence, dangerous terrain, or other potential dangers, are represented by a probability matrix and are based on a priori data. The sequence diagram in Figure 3-21 shows the activities performed in evaluating a single team.

For each team configuration tested, the mission is simulated for each of the previously selected paths. Projected failures are discarded, and projected successes are ranked according to the assigned criteria. At the end of a level, a percentage of the teams with the best scores move on to the next level. The mission simulation is performed in Team::runMission(), which is shown as a sequence diagram in Figure 3-22. The simulation retrieves the length, width, and danger probability then calculates the traversal time for that node and the transition to the next node, and then applies those times to the running score.

The throughput is determined using a generalized deterministic queuing network. The constraint, or service time, is based on the minimum width and length of the passage and the width and speed of the team members [65]. Unlike traditionally stochastic queuing models that must account for entries arriving at different times, ours assumes the team members arrive at once and is used to determine how quickly they can pass through. Team members are allowed to travel abreast, provided their combined width is smaller than the passageway. Figure 3-23 shows a portion of the urban layout with two levels of detail – the higher-level block view and the lower level street view, with a sample path. Figure 3-24 converts those maps into queuing models for the paths at both levels of abstraction. The more abstract queuing model in (a) is obtained in one

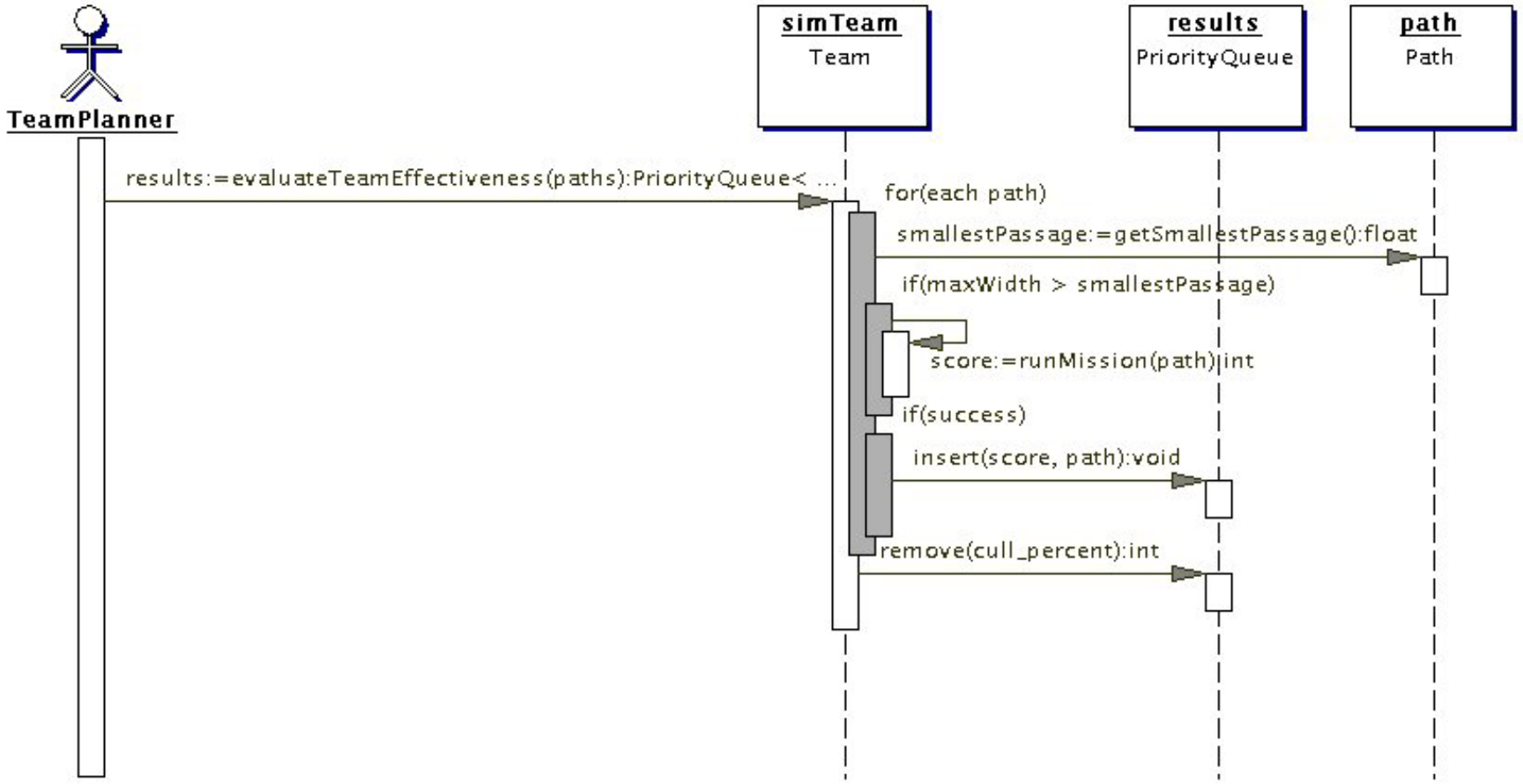

Figure 3-21. Sequence diagram for TeamEvaluation

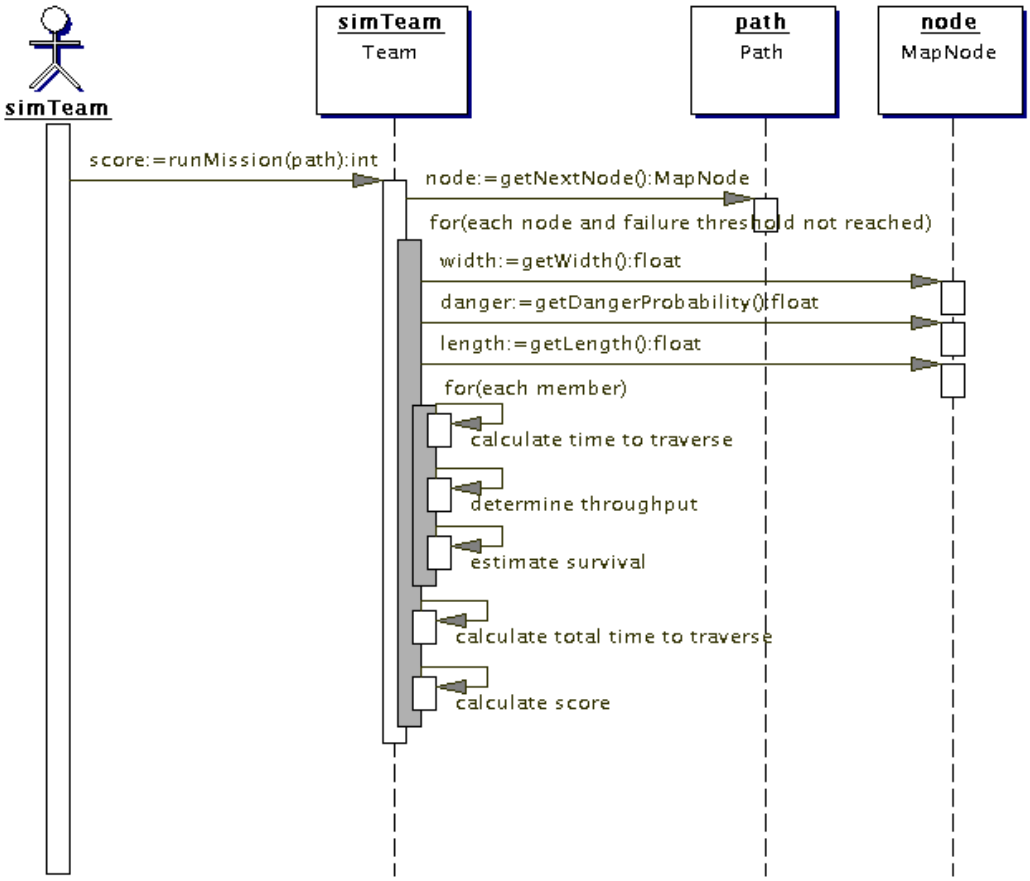

Figure 3-22. Simulation of individual path for a single team configuration

of two ways: by partitioning the lower level queuing model in (b), or by using a higher level within the scene graph defining the geometry of this area. The queuing model identifies a partial path from the entry into the NW Block on NW E Street, continuing as it turns into NW W Street, and turning in to NW Section 4 with the final goal of NW Area 1. Part (a) shows the queuing model at the “block” level, and part (b) shows it at the “section” level. The “block” level combines queuing nodes that represent major roads that pass through that block, the “section” level includes smaller local roads and sections off the larger roads at

intersections; a more detailed level would break out building queuing nodes, with as much detail as is known, such as “closet” and even “furniture.”

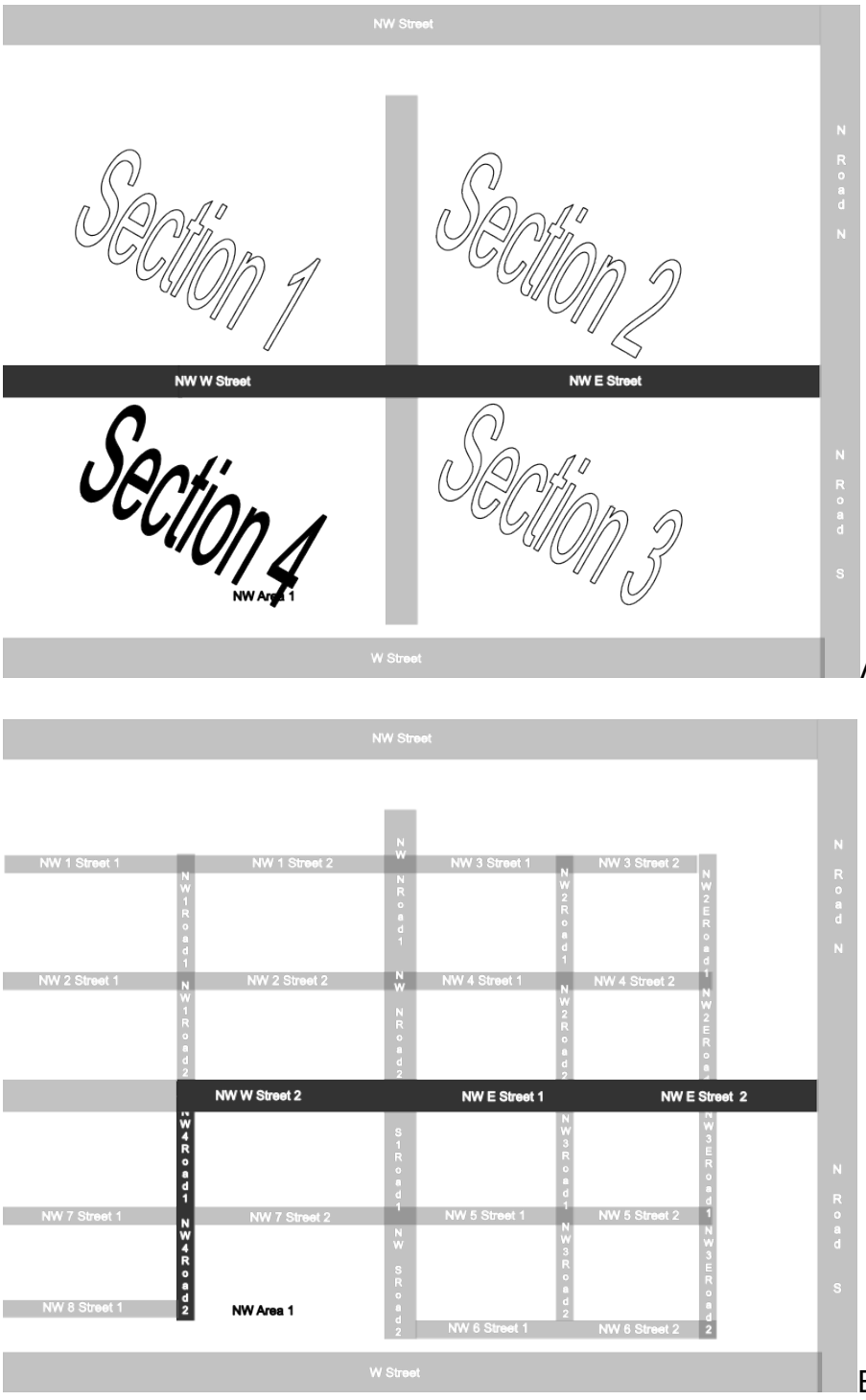

Figure 3-23. A path through the area at different levels. A) shows the level 2 structure of the map, and B) shows level 3.

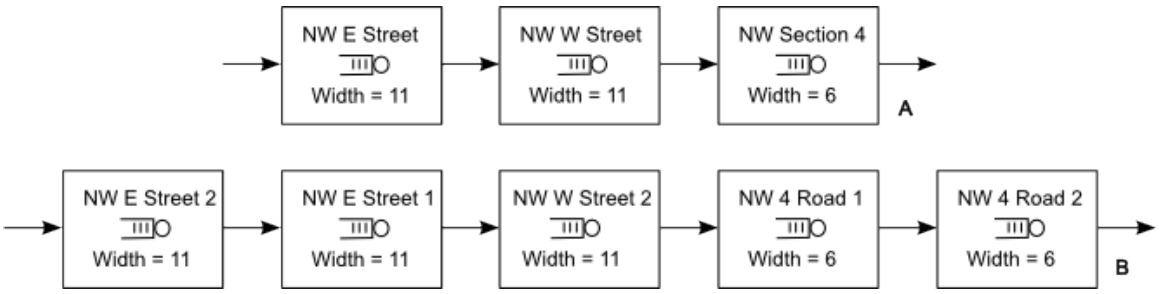

Figure 3-24. Scene graph representation of queuing model. A) shows a more abstract queuing model of the cutout area displayed in Figure 3-23 A, and B) shows the lower level queuing model of the same area, corresponding to Figure 3-23 B.

Scoring Algorithms

How do we determine an optimal team? There are many factors to consider in deciding the "winning" configuration. An obvious factor is the potential for succeeding in the mission goals. Of the team configurations that have the ability to complete the mission, the optimal team should have qualities that make it stand out from the others. Depending on mission parameters, it could be the team that can complete the mission fastest; the team with the fewest members overall and the fewest human members; the team that is least expensive to deploy; or the team that is most expendable or has the best ability to protect its human team members from injury or loss of life. Deployment cost might involve the cost associated with transporting a distant specialist, having to pay a highly skilled contractor a high bill rate, maintenance and operation costs of the unmanned systems, or replacement costs for a member that is lost in a dangerous mission, either through manufacturing cost of a UMS or training cost of a human member. The modular approach to the design allows us to assign an importance to each factor at runtime for a simple calculation, or to plug in a different, more complex formula to calculate the score for the team. Several of

these formulas are listed in Table 3-3. The results of the different scenarios and scoring models are discussed in the next chapter.

Table 3-3. Sample Formulas

Formula Type	Score
Weighted Prototype	T*time + N*size
Weighted expanded Cost	T*time + N*size + X*survival + C*cost sum(Resource:costToDeploy) + sum(time*Resource:hourlyRate)

T = importance of time to complete. N = importance of size of team.
X = importance of survival rate. C = importance of cost to deploy

Scenario

The initial target demonstration of the team selection tool was an urban search and rescue scenario. In this scenario, the goal of the mission was to travel undetected through an urban area to a building known to house hostages, enter the building, locate the hostages, free them, and get them to safety. Figure 3-25 shows a diagram with details of the mission, including the area layout, location of hostages and hostiles, and several suggested paths. The position of hostages is marked with a star. Portions of the Northeast and Southeast sections of the area have known or suspected snipers. Possible paths are marked by different color lines. Streets are numbered, with East/West being "Street" and North/South being "Road." For simplicity only the NW block is displayed, and buildings have been hidden from view. If building floor plans were available for a scenario, those details would be considered in the next level down. An arrow indicates the ingress route.

For this mission the assumed priority was the safety of the hostages, then the human team members, then the robotic team members. The initial prototype used a simple approach to success or failure. Loss or projected loss of any

hostages is deemed mission failure. Loss or projected loss of any human team member is undesirable but may be counted as a success in the absence of simulations that project no loss of human members. Loss or projected loss of unmanned human members is considered more costly/lower score than no loss, but counts as mission success. The success criteria are localized and could easily be changed based on mission or task constraints.

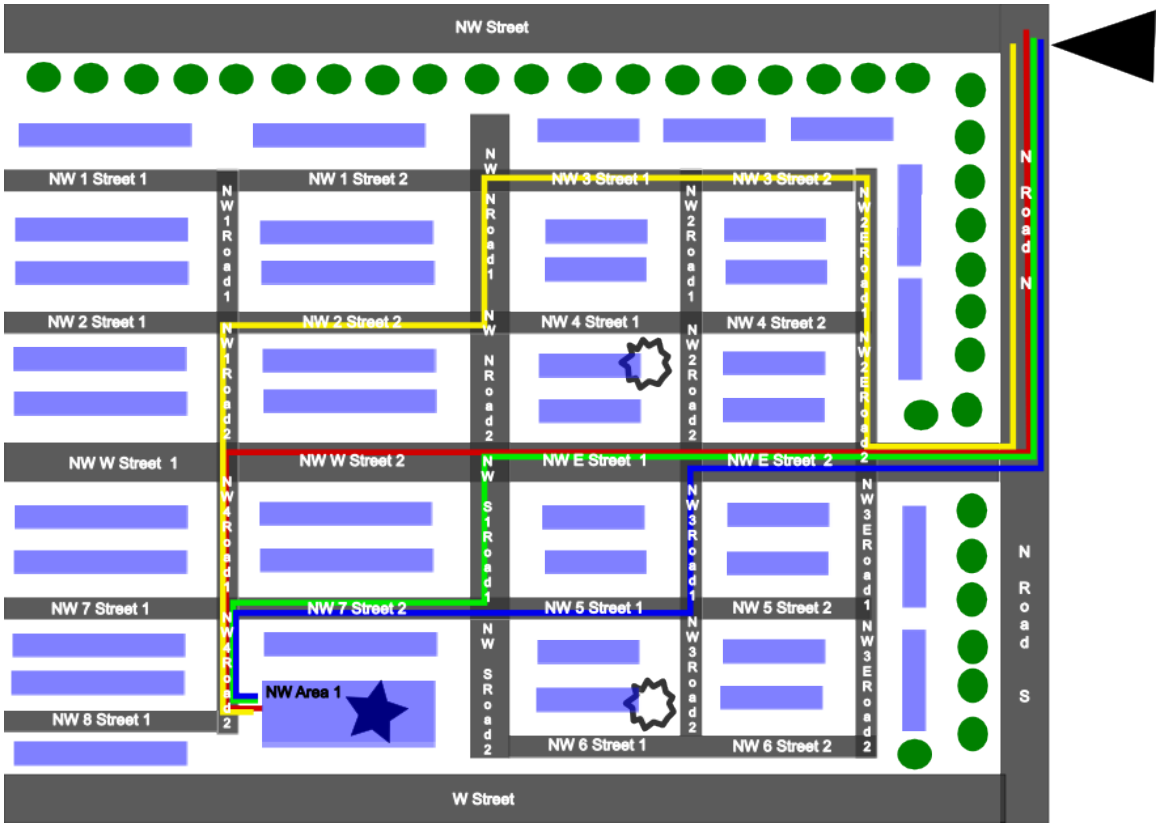

Figure 3-25. Diagram of a mission scenario with suggested paths. Starting point is the Arrow in the upper right, goal is NW Area1 which is marked with a star. Areas of higher danger probability are marked with a jagged circle.

CHAPTER 4
ANALYSIS

Comparison

To test the feasibility of the hybrid heuristic/progressive refinement approach, a brute-force simulation approach, a pure progressive refinement simulation approach, a hybrid heuristic/brute-force simulation approach, and the hybrid heuristic/progressive refinement approach were used to test multiple scenarios and cost functions. Their results were evaluated for performance, scalability, and accuracy. Memory needed for each approach was also tallied and evaluated for suitability in laptops and even smaller mobile devices.

On the simulation side, the brute-force simulation approach tests every team against every available path, with no intermediate levels that reduce the data set. The progressive refinement approach is tested in two configurations – reducing the bottom 10% of the teams after each level, and reducing the bottom 20% of the teams after each level. Each remaining team is tested against every available path. On the path planning side, there are three configurations used – all paths, and two heuristic-based path searches using a heuristic based on length and danger of each pathway, subject to a cap of either 3 or 4 times the straight-line distance from the start to the end. Each scenario is tested against 9 combinations – each of the three path searches paired with each of the three simulation configurations. Units used in the final prototype are generic scaled units based meters and meters per second.

Testing Environment

The prototype was written in Java using JSF 2.0 and PrimeFaces components. Since the prototype ran in a Java virtual machine, the application had to be redeployed

between runs so the garbage collection delays in a running program did not skew the memory footprint results. The test runs were performed on a MacBook Pro with a 3.06 GHz Intel Core 2 Duo processor, 4 GB 1067 MHz DDR3 Memory, using Mac OS X 10.6.6. The web application was run from within Netbeans in a Glassfish 3.0.1 application server.

Scenarios

Multiple scenarios were run to test performance at different sizes of the mission area, complexities of available paths, number of skills required, and number of available team combinations. Also included for comparison, but not included in the final results, are the initial test results from the early prototype. The early prototype contained a small set of members and skills, a small mission area, and an initial subset of paths instead of a generated path set. This prototype followed the pure progressive refinement approach. The current prototype contains more members, a larger set of physical characteristics, and more skills to choose from. Skills, for the purposes of the prototype, are defined with a simple label, such as “Carry Wounded,” and are used to choose which team members are relevant to each mission.

Simulation-Based Progressive Refinement, the Early Test

As part of the prototype phase, an initial set of test data was created. The results are provided below. The initial prototype was written in C++ with file-based data and was subsequently converted to Java for easier display creation.

Mission

The team must travel from the intersection of First & Main to the hostage location in Blue House, room 1 (BH Room 1), using one of 4 predefined paths. The area layout

is displayed in Figure 4-1, and the initial set of 12 potential team members is in Table 4-1. Units for length and width are in meters.

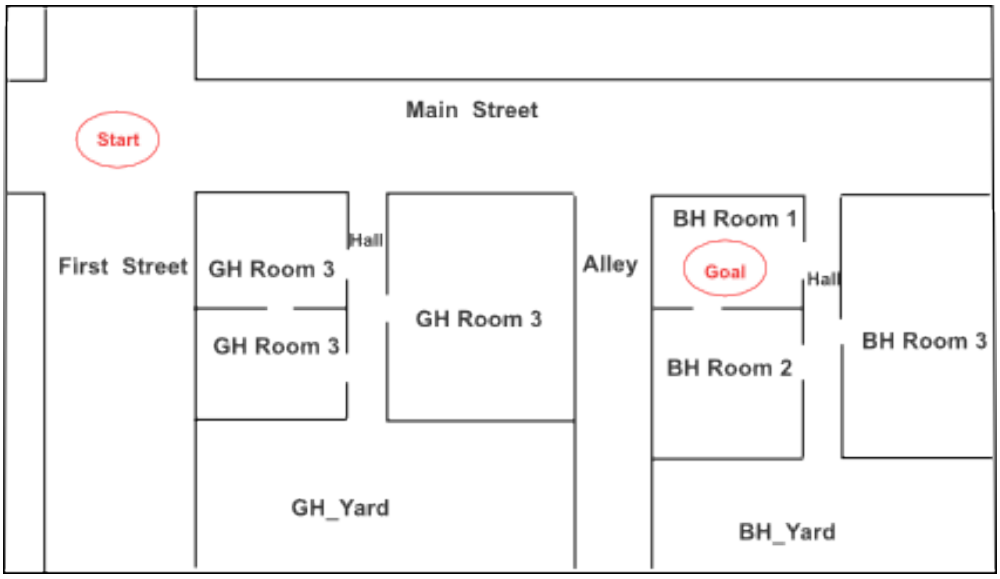

Figure 4-1. Mission Area with goals

Skills required:

- Weapon
- TargetId
- WeaponAuth
- HeatSensor
- Gurney

Table 4-1. Resources available in initial prototype

Name	Type	Length	Width	Weight	Speed	Skill
Fred	Human	0.5	0.5	180	5.0	TargetId
Barney	Human	0.5	0.5	180	5.0	WeaponAuth
Mitchell	Human	0.5	0.5	180	5.0	Weapon
Huey	UMS	2.0	2.0	25.0	5.0	HeatSensor
Talon	UMS	2.0	3.0	30.0	6.0	HeatSensor, TargetId
Cylon	UMS	4.0	4.9	120.0	8.0	Weapon
Viper	UMS	3.0	4.0	85.5	10.0	Weapon, TargetId
B9	UMS	5.0	5.0	35.5	8.0	HeatSensor
Robbie	UMS	5.0	4.5	35.5	15.0	HeatSensor
Mule	UMS	5.0	5.0	35.5	10.0	Gurney
M1	UMS	3.0	2.5	35.5	5.0	Gurney
T2	UMS	1.0	1.0	35.5	15.0	WeaponAuth

Results

This simulation generated 144 possible team combinations. The scores for level 2 ranged 148.587 to 313.218. In this scenario, the same team won the top two spots, with different recommended paths. Scores were computed based on navigational concerns only. Total simulation time was 0.96 seconds

Winning team and path

- Score 148.587
- Team members: Mitchell, Talon, M1, T2
- Path is shown in Figure 4-2.

Second place team and path

- Score 152.58
- Team members: Mitchell, Talon, M1, T2
- Path is shown in Figure 4-3.

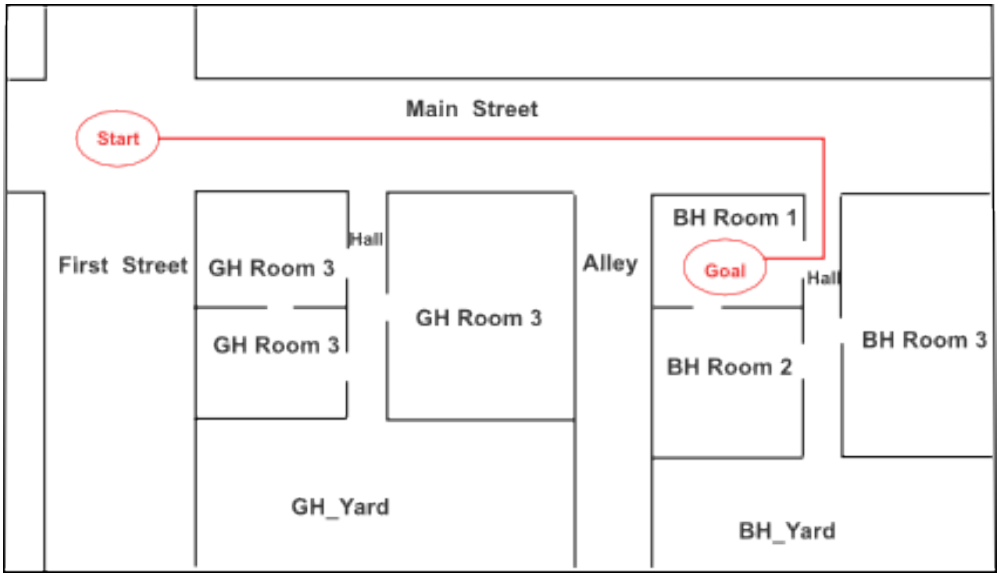

Figure 4-2. First place team/path

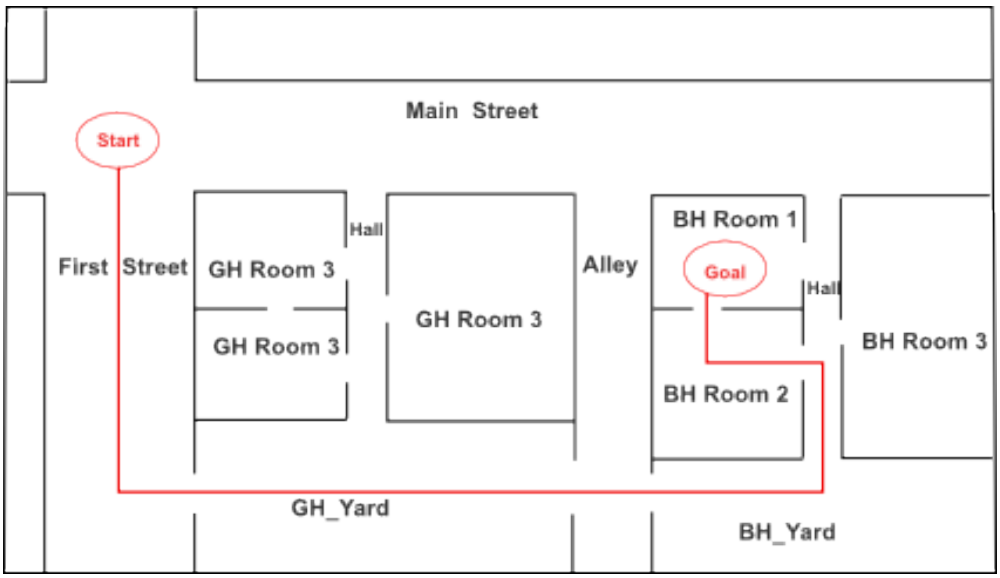

Figure 4-3. Second place team/path

Final Prototype, Scenario One

The first scenario included 15 potential members, displayed in Table 4-2, four skills, and a simple mission area with 4 levels. Nine variations were run against this data set – progressive refinement and brute force simulation each run with a distance ceiling of 3 and 4 times the direct distance, and with all paths. All of the progressive refinement variations were run with the bottom 10% and 20% of the teams removed after each level, starting at level 1 (the second level). The runtime results are displayed in Table 4-3. All nine variations had the same winners and second place team/path combinations. The run with a ceiling of three times the direct distance and a 20% removal rate had the best time. The only surprise was in the all paths scenario, the 20% removal performed slightly worse than the 10% removal.

Table 4-2. Resources available in scenario 1-2

Name	Type	Length	Width	Weight	Speed	Deploy Cost	Hourly Rate	Skill
B9	Robot	0.751	0.751	35	2.00	600	60	Heat Sensor
Barney	Human	0.152	0.152		0.80	2200	220	Authorize Fire Control Drone
Cylon	Robot	0.747	0.610	120	1.60	100	80	Carry Weapon
Fred	Human	0.152	0.152	150	0.78	1500	150	Control Drone ID Target
H1	Human	0.152	0.152	150	0.80	1900	190	Authorize Fire ID Target
H2	Human	0.152	0.152	180	0.90	1500	275	Control Drone Radio
Huey	Robot	0.305	0.305	23	1.10	500	75	Heat Sensor
M1	Robot	0.457	0.381	35	1.10	1000	75	Carry Wounded
Mitchell	Human	0.152	0.152	160	0.90	1100	200	Carry Weapon Authorize Fire
Mule	Robot	1.000	1.000	35	2.20	1200	110	Carry Wounded
R1	Robot	0.305	0.305	30	1.90	400	60	Authorize Fire Heat Sensor
Robbie	Robot	0.751	0.686	35	1.00	700	70	Heat Sensor
Talon	Robot	0.457	0.305	30	1.20	600	66	Heat Sensor ID Target
T2	Robot	0.152	0.152	35	1.50	750	250	Authorize Fire Control Drone
Viper	Robot	0.610	0.457	85	2.20	1500	150	ID Target

Table 4-3. Evaluation of approaches, scenario 1

Approach	Runtime, ms
Hybrid, low ceiling, 20% cut	7235
Hybrid, low ceiling, 10% cut	9214
Brute force, low ceiling	9828
Hybrid, high ceiling, 20% cut	12147
Hybrid, high ceiling, 10% cut	15870
Brute force, high ceiling	20952
Progressive refinement, all paths, 20% cut	50371
Progressive refinement, all paths, 10% cut	63092
Brute force, all paths	69829

Mission

The team must travel from NW E Street 3 to the hostage location in NW Area 1. The area layout is displayed in Figure 4-4. There are trouble spots near the intersection of the main roads that pass through the NW Block, as indicated on the map.

Skills required:

- Authorize Fire
- Carry Weapon
- Carry Wounded
- Heat Sensor

Winning scenario, time

- Score time = 107.55, cost = $3105.97
- Team: R1, Mule, Viper
- Path: As shown in Figure 4-5.

Second place scenario, time

- Score time = 127.71, cost = $1707.10
- Members: Cylon, R1, Mule
- Path: As shown in Figure 4-5.

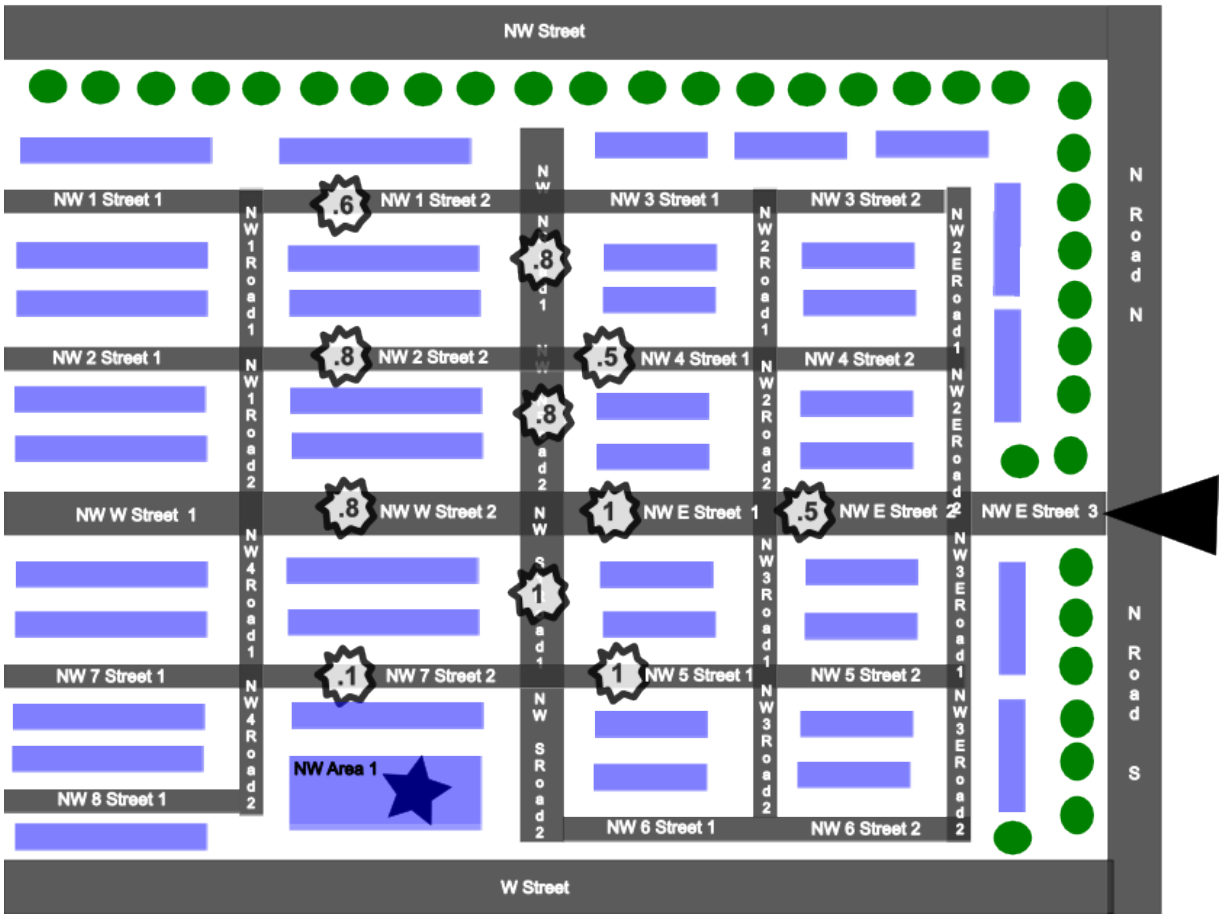

Figure 4-4. Scenario 1 layout. Starting point is at the arrow on NW E Street 3, goal is NW Area 1, marked with a star. Danger areas and their values are marked.

Winning scenario, cost

- Score cost = $1510.32, time = 185.77
- Team: R1, M1, Cylon
- Path: As shown in Figure 4-5, followed by several other paths up to 642.62 time, $1535.70 cost

Second place team, cost

- Score time = 127.71, cost = $1707.10
- Members: Cylon, R1, Mule
- Path: As shown in Figure 4-5.

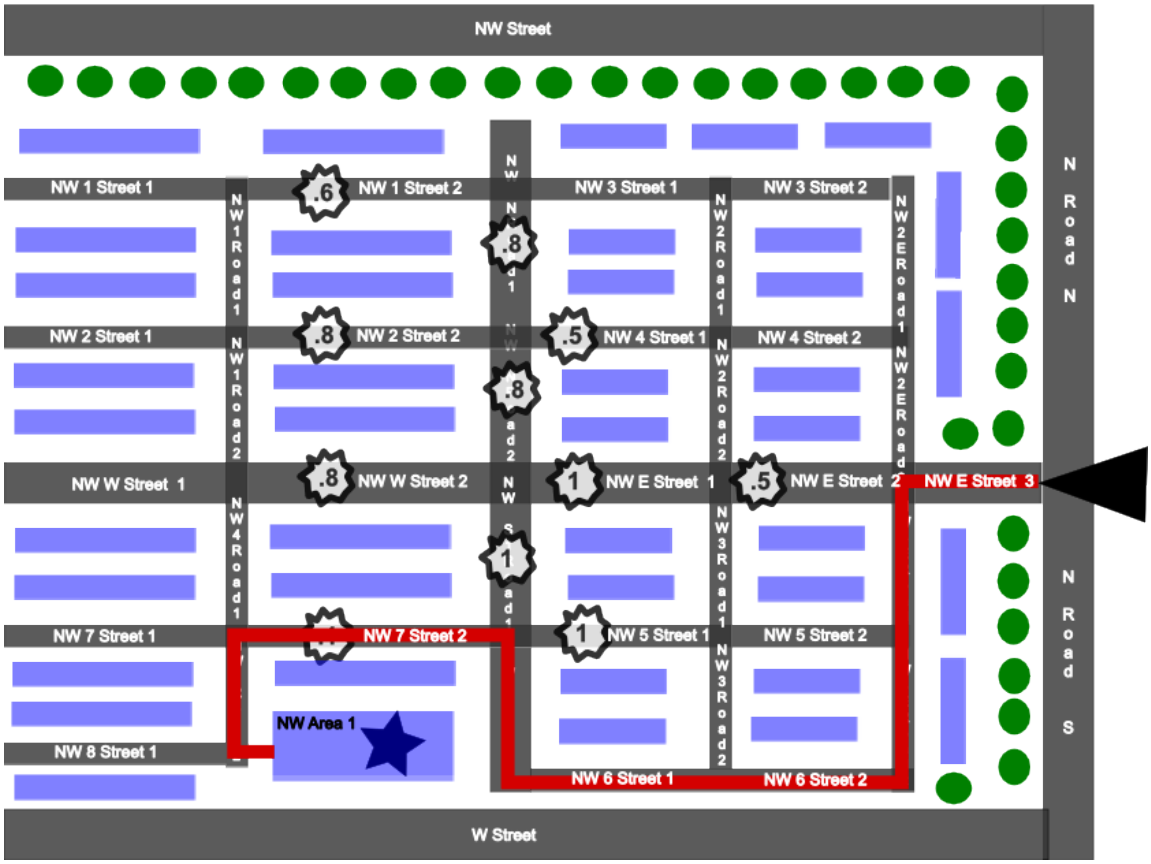

Figure 4-5. Scenario 1 winning path. Path displayed in red is the winning path. Both first and second place teams in timing and cost chose the same path.

Results, low ceiling

This simulation generated 150 possible team combinations and 27 leaf-level paths, using a ceiling of 3 times the direct distance. The scores for level 4 ranged a time of 107.55 to 1249.50 and cost of $1510.32 to $5636.70. Overall scores were computed based on speed and cost independently. In this scenario, there was a single time winner and a single different cost winner with multiple paths, so the priority, either speed or cost, would determine which team was chosen. However, the same team/path came in second place for both cost and time, so with a cost function that provided equal balance between cost and time, the second place team would likely be chosen.

With a brute-force simulation, all 150 teams are simulated against all 27 paths. In the case where 10% of the lowest-performing teams are cut each level, the first level runs 150 teams with 1 path, the second level has 135 teams and 23 paths, and the third has 121 teams and 27 paths. By cutting the bottom 20% teams with each level, the same winning teams are generated with 150, 120, and 96 teams for each level. This configuration performed the best, with no loss of accuracy.

Results, higher ceiling

This simulation generated 150 possible team combinations and 68 leaf-level paths, using a ceiling of 4 times the direct distance. The scores for level 4 ranged a time of 107.55 to 1477.14 and cost of $1507.94 to $5663.57. Overall scores were computed based on speed and cost independently. The winners of this scenario, which was less restrictive on the paths included in the simulation, were identical to the more restrictive ceiling. This outcome shows that, for this data set, the bounding of 3 times the ceiling was sufficient. The number of teams run for each level was consistent with the low ceiling results.

Results, all paths

This simulation generated 150 possible team combinations and 287 leaf-level paths, using no cost ceiling. The scores for level 4 ranged a time of 107.55 to 1881.06 and cost of $1510.32 to $5687.10. Overall scores were computed based on speed and cost independently. The winning results, as expected, were unchanged from the heuristic-based runs; however the run time was significantly higher.

Final Prototype, Scenario Two

The second scenario included the same 15 potential members and the same area, with the exception of a new obstruction. The same set of 9 configurations was run

against that scenario. The results shown in Table 4-4 show that once again the hybrid approach with a 20% reduction in teams per level performed the best.

Table 4-4. Evaluation of approaches, scenario 2

Approach	Runtime, ms
Hybrid, low ceiling, 20% cut	5455
Hybrid, low ceiling, 10% cut	7479
Brute force, low ceiling	7840
Hybrid, high ceiling, 20% cut	10007
Hybrid, high ceiling, 10% cut	11255
Brute force, high ceiling	13742
Progressive refinement, all paths, 20% cut	32857
Progressive refinement, all paths, 10% cut	49261
Brute force, all paths	42343

Mission

The team must travel from NW E Street3 to the hostage location in NW Area 1. The area layout is displayed in Figure 4-6. There are trouble spots near the intersection of the main roads that pass through the NW Block and a vehicle is partially blocking the intersection between NW_3ERoad1, NW_3ERoad2, and NW_5Street2. This blockage allows only the humans and a few of the UMS members to travel that route.

Skills required:

- Authorize Fire
- Carry Weapon
- Carry Wounded
- Heat Sensor

Winning scenario, time

- Score time = 132.79, cost = $3107.37
- Team: R1, Viper, Mule
- Path: As shown in Figure 4-7.

Second place scenario, time

- Score time = 142.55, cost = $3107.92:
- Team: R1, Viper, Mule
- Path: As shown in Figure 4-8.

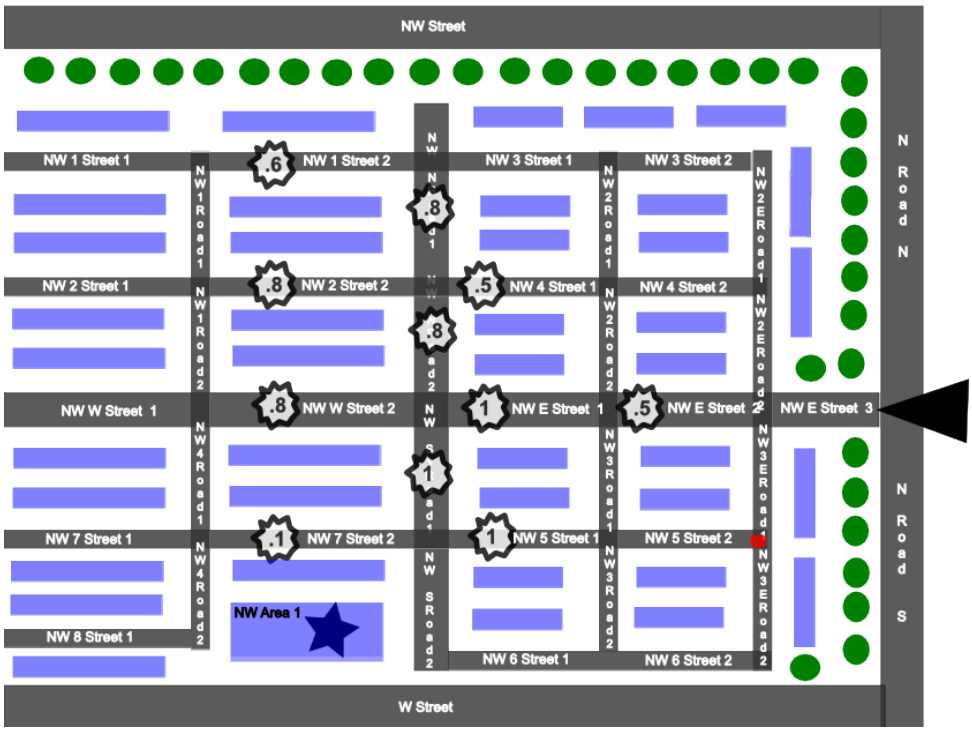

Figure 4-6. Layout for scenario 2. There is an obstruction in the intersection between NW 3 E Road 1, NW 5 Street 2, and NW 3 E Road 2 that reduces the width of the passage.

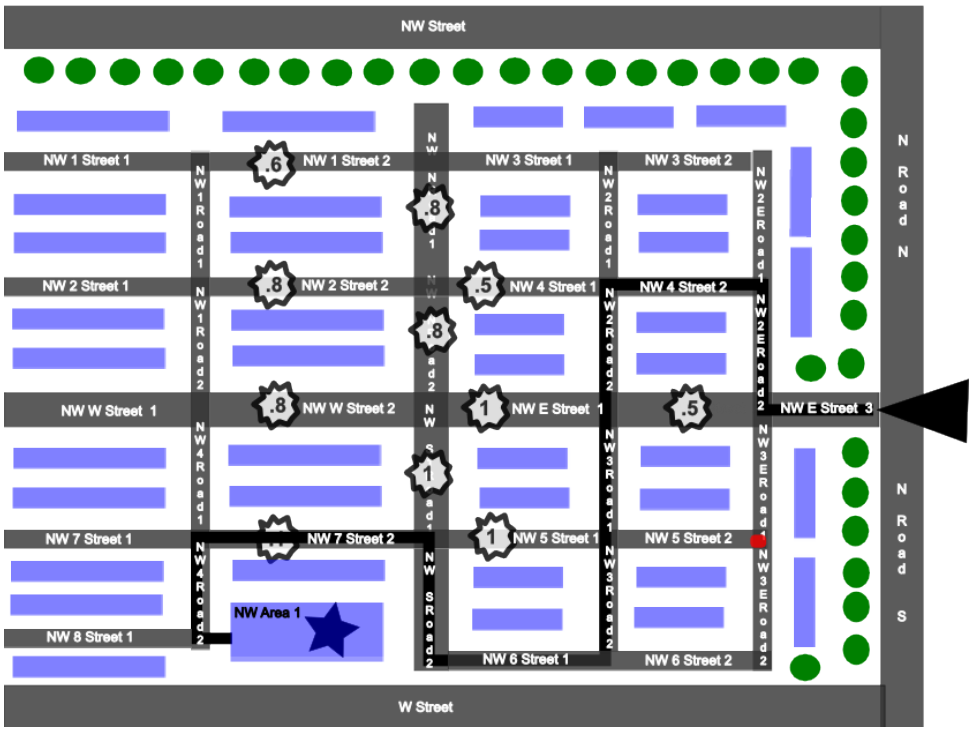

Figure 4-7. Winning path for scenario 2. Because of the blockage and the higher percentage of danger, the most effective path is to go up a block to NW 4 Street 2 and back down.

Winning scenario, cost

- Score time = 229.37, cost = $1512.74
- Team: M1, Cylon, R1
- Path: As shown in Figure 4-7, plus several other paths for same team at increasing costs

Second place team, cost

- Score time = 157.69, cost = $1708.76
- Team: Cylon, R1, Mule
- Path: As shown in Figure 4-7.

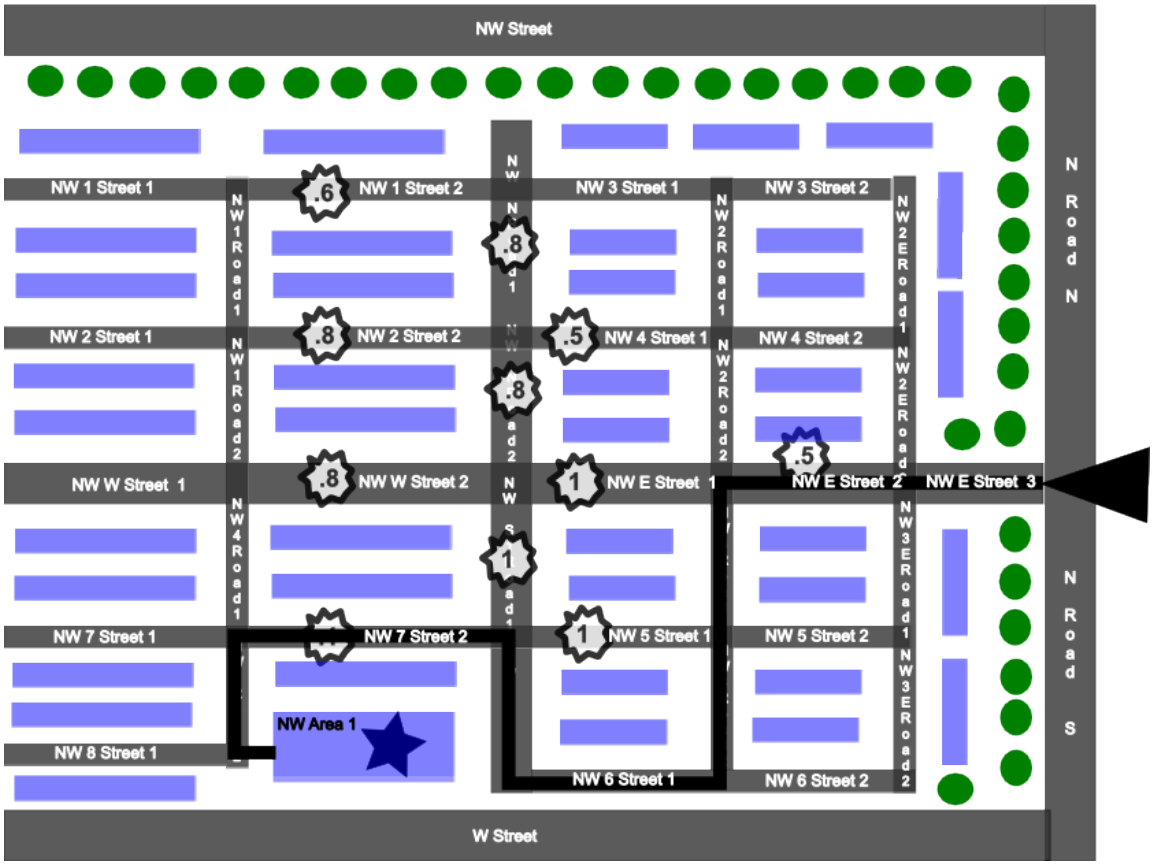

Figure 4-8. Second place path for scenario 2.

Results, lower ceiling

This simulation generated 150 possible team combinations and 27 leaf-level paths, using a ceiling of 3 times the direct distance. The scores for level 4 ranged a time of 142.55 to 1231.91 and cost of $1512.92 to $5651.07. Overall scores were

computed based on speed and cost independently. In this scenario, there was a single time winner and a single different cost winner with several different paths, so the priority, speed or cost, would determine which team was chosen. However, the same team/path came in second place for both cost and time, so given a cost function that adequately combined cost and time, the second place team would likely be chosen. The lower ceiling, however, caused the winning path/team combination for both time and cost to be eliminated.

Results, higher ceiling

This simulation generated 150 possible team combinations and 68 leaf-level paths, using a ceiling of 4 times the direct distance. The scores for level 4 ranged a time of 132.79 to 1455.43 and cost of $1512.74 to $5653.21. Overall scores were computed based on speed and cost independently. In this scenario, one team took first and second places with 2 different paths, with scores ranging from 132.79 to 149.44. A different team led the cost with 40 different paths and costs ranging from $1512.74 to 1535.46. This scenario, with the higher ceiling on the heuristic, came in with a different winner than the lower ceiling, with a lower time score and a lower cost score. The early, heuristics-based path planning is designed to choose the paths most likely to be favorable, without knowledge of individual team constraints. In this particular cost function, the danger level was used to choose the type of movement – efficient, slow and cautious, or a slower leapfrogging move called bounding overwatch. These movements are dependent on individual resource speeds and widths, and overall team values, as opposed to the calculation done in the path selection, which is based purely on length and danger. The higher ceiling and the all paths results were the same, so

the heuristics-based planning was still an improvement; it just needed a higher ceiling to get the correct answer.

Results, All Paths

This simulation generated 150 possible team combinations and 287 leaf-level paths, using no cost ceiling. The scores for level 4 ranged a time of 132.79 to 1858.08 and cost of $1512.74 to $5678.30. Overall scores were computed based on speed and cost independently. All three versions – progressive refinement with 80% retention, progressive refinement with 90% retention, and brute force – got the same values for the winners, with more results on the slow side for the progressive refinement steps.

Final Prototype, Scenario Three

The third scenario included the same 15 potential members and the same area as scenario two, with a change in the skills. The list of resources is displayed in Table 4-6. In the first two scenarios, skills were assigned to members somewhat randomly, including “authorization to fire” being held by both human and UMS resources. Due to the estimated costs and speeds of the UMS resources, this caused all of the winning members to be unmanned – a scenario that technology is not ready for. By the same token, technology – and society – is not ready, nor may it ever be ready, to give non-humans the power to authorize weapons fire. This scenario was created to remove authorization to fire from the UMS members both to accurately reflect the technology, and to provide a mechanism to force at least one human onto the team. The same set of 9 configurations was run against that scenario. The results shown in Table 4-5 show that once again the hybrid approach with a 20% reduction in teams per level performed the best.

Table 4-5. Evaluation of approaches, scenario 3

Approach	Runtime, ms
Hybrid, low ceiling, 20% cut	3474
Hybrid, low ceiling, 10% cut	4210
Brute force, low ceiling	5656
Hybrid, high ceiling, 20% cut	6689
Hybrid, high ceiling, 10% cut	7980
Brute force, high ceiling	7680
Progressive refinement, all paths, 20% cut	20888
Progressive refinement, all paths, 10% cut	23169
Brute force, all paths	22805

Mission

The team must travel from NW E Street3 to the hostage location in NW Area 1. The area layout, unchanged from the previous scenario, is displayed in Figure 4-6. There are trouble spots near the intersection of the main roads that pass through the NW Block and a vehicle is partially blocking the intersection between NW_3ERoad1, NW_3ERoad2, and NW_5Street2. This blockage allows only the humans and a few of the smaller UMS members to travel that route.

Skills required:

- Authorize Fire
- Carry Weapon
- Carry Wounded
- Heat Sensor

Winning scenario, time and cost

- Score time = 224.22, cost = $2512.46
- Team: R1, Mitchell, M1
- Path: As shown in Figure 4-9.

Second place scenario, time, and cost second place team

- Score time = 224.22, cost = $2612.46
- Team: Huey, Mitchell, M1
- Path: As shown in Figure 4-9.

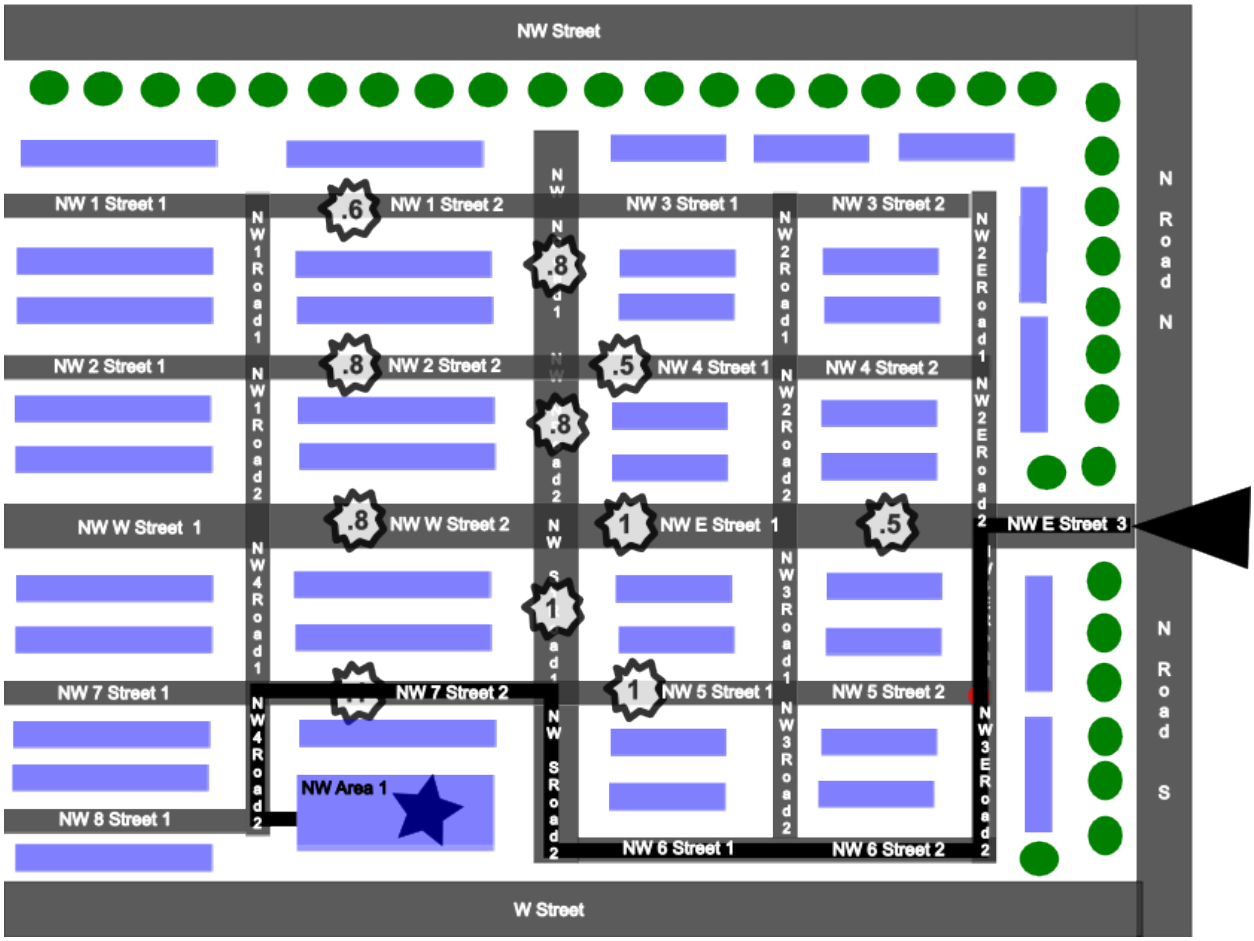

Figure 4-9. Winning path, scenario 3

Results, lower ceiling

This simulation generated 60 possible team combinations and 27 leaf-level paths, using a ceiling of 3 times the direct distance. The scores for level 4 ranged a time of 142.55 to 1231.91 and cost of $1512.92 to $5651.07. Overall scores were computed based on speed and cost independently. In this scenario, three teams tied for first place, with slightly different costs – one of which was also the cost winner. As was the case with the other two scenarios, the team winning in cost had several more paths that came in before the next team. All nine versions – progressive refinement with 80% retention, progressive refinement with 90% retention, and brute force; all paths, high

ceiling, low ceiling – got the same values for the winners, with more results on the slow side for the progressive refinement steps. The progressive refinement with an 80% retention and the lower ceiling had the best runtime with no loss in accuracy for the winning team/path combinations. The 80% retention progressive refinement approach processed 60 teams and 1 path for the first pass, 48 teams with 23 paths for the second pass, and 38 teams with 27 paths for the third pass. The 90% retention progressive refinement approach processed 60 teams and 1 path for the first pass, 54 teams with 23 paths for the second pass, and 48 teams with 27 paths for the third pass.

Results, higher ceiling

This simulation generated 60 possible team combinations and 68 leaf-level paths, using a ceiling of 4 times the direct distance. The scores for level 4 ranged a time of 224.22 to 1858.08 and cost of $2512.46 to $5653.21. Winning results are identical to the lower ceiling results across all three versions. The 80% retention progressive refinement approach processed 60 teams and 1 path for the first pass, 48 teams with 36 paths for the second pass, and 38 teams with 68 paths for the third pass. The 90% retention progressive refinement approach processed 60 teams and 1 path for the first pass, 54 teams with 36 paths for the second pass, and 48 teams with 68 paths for the third pass.

Results, All Paths

This simulation generated 60 possible team combinations and 287 leaf-level paths, using no cost ceiling. The scores for level 4 ranged a time of 224.22 to 1858.08 and cost of $2512.46 to $5678.30. Overall scores were computed based on speed and cost independently. The runtime was slightly faster for the brute force than the 90% retention progressive refinement, but the 80% retention progressive refinement had the

lower runtime. This is due to the smaller number of teams with the given configuration, so fewer teams are cut from the 90% progressive refinement, thus more iterations are run than the brute force, which runs through all of the teams, all of the leaf nodes. The 80% retention progressive refinement approach processed 60 teams and 1 path for the first pass, 48 teams with 78 paths for the second pass, and 38 teams with 287 paths for the third pass. The 90% retention progressive refinement approach processed 60 teams and 1 path for the first pass, 54 teams with 78 paths for the second pass, and 48 teams with 287 paths for the third pass.

Table 4-6. Resources available in scenario 3

Name	Type	Length	Width	Weight	Speed	Deploy Cost	Hourly Rate	Skill
B9	Robot	0.751	0.751	35	2.00	600	60	Heat Sensor
Barney	Human	0.152	0.152		0.80	2200	220	Authorize Fire Control Drone
Cylon	Robot	0.747	0.610	120	1.60	100	80	Carry Weapon
Fred	Human	0.152	0.152	150	0.78	1500	150	Control Drone ID Target
H1	Human	0.152	0.152	150	0.80	1900	190	Authorize Fire ID Target
H2	Human	0.152	0.152	180	0.90	1500	275	Control Drone Radio
Huey	Robot	0.305	0.305	23	1.10	500	75	Heat Sensor
M1	Robot	0.457	0.381	35	1.10	1000	75	Carry Wounded
Mitchell	Human	0.152	0.152	160	0.90	1100	200	Carry Weapon Authorize Fire
Mule	Robot	1.000	1.000	35	2.20	1200	110	Carry Wounded
R1	Robot	0.305	0.305	30	1.90	400	60	Heat Sensor
Robbie	Robot	0.751	0.686	35	1.00	700	70	Heat Sensor
Talon	Robot	0.457	0.305	30	1.20	600	66	Heat Sensor ID Target
T2	Robot	0.152	0.152	35	1.50	750	250	Control Drone
Viper	Robot	0.610	0.457	85	2.20	1500	150	ID Target

CHAPTER 5
CONCLUSIONS

Effectiveness of Hybrid Approach

The results of the testing shows that adding a heuristic semi-optimal path selection to the progressive refinement approach does significantly improve the efficiency of the algorithm in large data sets. In each testing scenario, the hybrid approach using the most restrictive cap on the path during path selection and the most aggressive culling of poorly performing teams was the fastest overall. In one of the scenarios the more restrictive, aggressive hybrid eliminated the winning path, but the most optimal team was still a front-runner using a different path. That missing path was found when the cap on the path length was raised slightly.

Our testing shows that with properly chosen heuristics, the hybrid approach can achieve the same accuracy, with far lower cost, as the progressive refinement alone or the brute force simulation. With a complex scoring function, the path selection heuristics must be chosen with care. This also holds true for paths with an indirect route. For instance, if the starting and goal points are fairly close together but separated by an impenetrable boundary, the cap on path length must be increased to get any paths. In addition, a heuristic in that scenario might be the change in distance between current node and goal node – while some movement away from the goal is necessary, a constant trending in the wrong direction is likely to yield a costly path.

The heuristic multiple-path search, as opposed to all paths or single best path search, showed an improvement in processing time when combined with each of the simulation methods – the tighter the cap, the faster the ensuing simulation. Conversely, the manner of the simulation also makes a difference. In a scenario where

there are few teams, a leaf-layer brute-force simulation can sometimes be faster than a conservative progressive refinement simulation – that is, a progressive refinement where fewer of the lower-performing teams are eliminated before the next level. A more aggressive elimination was faster than brute force in all instances, however.

On their own, the heuristic multiple-path search and progressive refinement each displayed performance gains with no loss to accuracy; combined, they made even greater performance improvements. Our testing indicates that the performance improvements are greater with a bigger data set, especially with the aggressive progressive refinement.

Contributions

Our goal in this research was to contribute to the knowledge in two broad areas, as designated by the ACM's Computing Classification System – Artificial Intelligence and Simulation. The contribution to Artificial Intelligence was two-fold – the first contribution was the definition of a methodology to determine a near-optimal team from a larger pool of resources, both human and robot, each individual possessing a set of useful skills and having unique operational and physical characteristics. The second contribution was a bridge between Artificial Intelligence and Simulation in the form of a hybrid heuristic path planning/multiple discrete event simulation approach, using progressive refinement. We have demonstrated in the prototype that there is an efficiency advantage to creating that hybrid; in a dynamic immediate-need situation, that efficiency advantage could be key.

Future Work

The scoring scenarios used in our study used simplistic team movement. A future enhancement to this simulation would be to incorporate more realistic team movements

that include having members split up to go different routes, including the frequently used scenario where the UMS travels ahead of the human members to test environmental factors, such as air quality and ground stability. Subject matter experts in the field of the task would need to be interviewed to accurately model the team movements. Also, as different team operational scenarios are devised, more complex scoring functions should be developed to include factors such as a member's proficiency at a given skill. Other intangibles that should be considered are minimum or maximum requirements for humans or robots. Based on the cost functions that relied exclusively on speed and cost, the winning teams all ended up comprised of all robots. This prompted the third scenario, in which one of the skills was limited to human members. Current technology, while improving, is not mature to the point of having all robots on an autonomous team. Identifying skills that only a human, or only a robot, can perform would ensure that at least one human and at least one robot are members of the final team.

Visualization plays a large part in conveying information and understanding. While graphical user interfaces were built for the prototype for entering resources and selecting mission parameters, the task of building a visual, interactive display of the area fell beyond the scope of our work. This would be a most welcome addition to the project, not only for aesthetics and quicker conveyance of results, but also to aid in the task of entering the map data, which was a tedious, time-consuming task of hand-creating n-triples. A good visual editor with the capability to add other metadata like weight limits and probability of danger and to generate paths and entryways automatically would have eliminated mistakes and a great deal of eye strain. Automatic generation of map data is crucial for larger areas of interest.

Another improvement would be to add the third dimension to the prototype, and to include unmanned air vehicles (UAV). UAVs are increasingly crucial resources in search and rescue operations and in military operations, and the technology is maturing to the point where UAVs can act more autonomously. The addition of UAVs would require not only 3D modeling and planning, but also more complex team behaviors, since limitations on UAVs are different than limitations on ground-based team members.

The heuristic graph-based search fit the path-planning domain, but could be used for other applications as well. Social network analysis, or friend-of-a-friend discovery, is a common graph-walking application. Used to discover relationships between people, entities, or events, heuristics can be applied based on strength of relationships, allowing the user to investigate several possible linkages between two entities. For instance if John knows Mary, and Mary is Ann's sister, and Fred shops at the same store as Mary, the strength of the relationship between John and Mary is stronger than Fred and Mary, so John may be more directly correlated to knowing Ann as well.

The work done in team planning could go beyond the urban search and rescue scenario used in the prototype. It could be used in an agricultural or factory setting to determine which combination of equipment would function most efficiently for a given task. The costing function might be adapted to penalize teams that have idle time – faster members having to wait for slower members, resulting in a more evenly-matched team as opposed to the fastest overall.

LIST OF REFERENCES

[1] A. Carnegie. (2009, November 23). *Quotes for Teamwork* [Online]. Available: http://www.motivatingquotes.com/teamwork.htm

[2] J. Averill, *et al*. "Occupant Behavior, Egress, and Emergency Communications," Final *Reports of the Federal Building and Fire Investigation of the World Trade Center Disaster*. National Institute of Standards and Technology, Gaithersburg, MD, Rep. NIST NCSTAR 1-7, Sept. 2005.

[3] Wikipedia contributors. (2009, November 18). "September 11 attacks," *Wikipedia, The Free Encyclopedia* [Online]. Available: http://en.wikipedia.org/wiki/September_11_attacks.

[4] The Mount Sinai Hospital / Mount Sinai School of Medicine (2009, November 3). "World Trade Center Responders Plagued With Asthma; 9/11 Responders Twice As Likely To Have Asthma," *ScienceDaily*. [Online]. Available: http://www.sciencedaily.com/releases/2009/11/091103144818.htm

[5] University of South Florida. (2009, November 18). "History of Rescue Robots," *Center for Robot Assisted Search & Rescue* [Online]. Available: http://www.crasar.org/rescuerobots/history.htm

[6] A. Boyle, (2005, August). *How high-tech is coming to the rescue: Scientists bring gadgets to post-Katrina disaster scene* [Online]. Available: http://www.msnbc.msn.com/id/9131498/

[7] R. Murphy, J. Kravitz, S. Stover, and R. Shoureshi, "Mobile Robots in Mine Rescue and Recovery," *IEEE Robotics & Automation Magazine*, vol. 16, no. 2, pp. 91-103, 2009.

[8] T. Cioppa, J. Willis, N. Goerger, L. Brown, "Research Plan Development for Modeling and Simulation of Military Operations in Urban Terrain," *Proceedings of the 2003 Winter Simulation Conference*, vol. 1, pp. 1046-1051, 2003.

[9] Floyd D. Spence National Defense Authorization Act for Fiscal Year 2001, HR.4205.ENR, Section 220 (2001).

[10] OUSD (AT&L) Defense Systems/Land Warfare and Munitions (2005). *Joint Robotics Program Master Plan*. [Online]. Available: http://www.jointrobotics.com/library01.php

[11] D. J. Bruemmer, D. D. Dudenhoeffer, J. Marble, "Dynamic-Autonomy for Urban Search and Rescue," *Proceedings of the 2002 AAAI Mobile Robot Workshop*, Edmonton, Canada, August 2002, p. 3337.

[12] D.J. Bruemmer, and M. Walton. “Collaborative Tools for Mixed Teams of Humans and Robots,” *Proceedings of the Workshop on Multi-Robot Systems,* Washington D.C., March 2003, pp. 219-229.

[13] T. Fong, C. Kunz, L Hiatt, M Bugajska, “The Human-Robot Interaction Operating System,” *Proc. 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, Salt Lake City, Utah, 2001.

[14] H. J. Levesque, P. R. Cohen, and J. H. T. Nunes. “On Acting Together,” *Proceedings of AAAI-90*, Boston, MA, 1990, pp. 94–99.

[15] T. Laengle, T. Hoeniger, L. Zhu, “Cooperation in Human-Robot-Teams,” *Proceedings of the IEEE International Symposium on Industrial Electronics, ISIE '97*, Guimaraes, Portugal, vol.3, pp. 1297-1301, 1997.

[16] R. Murphy, “Human-Robot Interaction in Rescue Robotics,” *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews, vol. 34, no. 2*, pp. 138 – 153, May 2004.

[17] I. Nourbakhsh, et al, “Human-Robot Teaming for Search and Rescue,” *IEEE Pervasive Computing: Mobile and Ubiquitous Systems*, pp. 72-78, 2005.

[18] R. Chalmers, D. Scheidt, T. Neighoff, S. Witwicki, R. Bamberger, “Cooperating Unmanned Vehicles,” *Proceedings of the AIAA 1st Intelligent Systems Technical Conference*, 2004.

[19] F. Lacombe, “Modeling Swarm Behavior,” *PHYSorg.com*, 21 February 2006. [Online]. Available: http://www.physorg.com/news11060.html.

[20] D. Scheidt, J. Stipes, “Cooperating Unmanned Vehicles,” *Proceedings of the 2005 IEEE International Conference on Networking, Sensing and Control*, pp. 326-331, 2005.

[21] M. Taylor, M. Jain, C. Kiekintveld, J. Kwak, R. Yang, Z. Yiu, M. Tambe, “Two Decades of Multiagent Teamwork Research: Past, Present, and Future”, *Postproceedings of Collaborative Agents REsearch and Development (CARE) 2010 workshop*, 2011

[22] T. Gupta, P. Varakantham, T. Rauenbusch, M. Tambe, “Demonstration of Teamwork in Uncertain Domains using Hybrid BDI-POMDP Systems,” *Proc. 6th Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS),* article 264*,* 2007.

[23] S. Prabhala, J. Gallimore, S. Narayanan, “Human Effectiveness Issues in Simulated Uninhabited Combat Arial Vehicles,” *Proceedings of the 2003 Winter Simulation Conference*, vol. 1, pp. 1034 – 1038, 2003.

[24] A. Freedy, O. Sert, E. Freedy, J. McDonough, G. Weltman, M. Tambe, T. Gupta, W. Grayson, P. Cabrera, “Multiagent Adjustable Autonomy Framework (MAAF) for

Multi-Robot, Multi-Human Teams," *Proceedings of the Symposium on Collaborative Technologies & Systems,* pp. 498 – 505, 2008.

[25] M. Bazargain-Lari, P. Gupta, S. Young, "A Simulation Approach to Manpower Planning," *Proceedings of the 2003 Winter Simulation Conference*, pp. 1677-1685, 2003.

[26] R. Guttkuhn, T. Dawson, U. Trutschel, J. Walker, M. Moroz, "A Discrete Event Simulation for the Crew Assignment Process in North American Freight Railroads," *Proceedings of the 2003 Winter Simulation Conference*, pp. 1686-1692, 2003.

[27] S. Ganapathy, R. Hill, "Dynamic Path-Planning for Search and Destroy Missions – The Bay of Biscay Scenario," *Proceedings of the 2003 Winter Simulation Conference*, pp. 999-1003, 2003.

[28] F. Kamrani, A. Rassul, "Simulation-Aided Path Planning of UAV," *Proceedings of the 2007 Winter Simulation Conference*, pp. 1306-1314, 2007.

[29] C. Hill, L. Malone, "Caveats For Simulation Modeling In Support of Decision Making," *Proceedings of the 2003 Winter Simulation Conference,* pp. 1102-1109, 2003.

[30] J. Lee, P. Fishwick, "Real-Time Simulation-Based Planning for Computer Generated Force Simulation," *Simulation,* vol. 63, pp. 299-315, 1994.

[31] D. Miller, "The Role of Simulation in Planning," *Proceedings of the 1987 Winter Simulation Conference*, pp. 530-533, 1987.

[32] D. Joslin, W. Poole, "Agent-Based Simulation for Software Project Planning," *Proceedings of the 2005 Winter Simulation Conference*, pp. 1059-1066, 2005.

[33] G. Drake, J. Smith, "Simulation System for Real-Time Planning, Scheduling, and Control," *Proceedings of the 2006 Winter Simulation Conference*, pp. 1083-1090, 2006.

[34] J. Lee, P. Fishwick, "Simulation-Based Real-Time Decision Making for Route Planning," *Proceedings of the 1995 Winter Simulation Conference*, pp. 1087-1095, 1995.

[35] H. Nejad, D. Zhu, A. Mosleh, "Hierarchical Planning and Multi-Level Scheduling for Simulation-Based Probabilistic Risk Assessment," *Proceedings of the 2007 Winter Simulation Conference*, pp. 1189-1197, 2007.

[36] X. Hu, Y. Li, J. Guo, L. Sun and A. Zeng, "A Simulation Optimization Algorithm With Heuristic Transformation and its Application to Vehicle Routing Problems," *International Journal of Innovative Computing, Information and Control,* vol. 4, no. 5 pp. 1169-1181, May 2008.

[37] R. Hoeckley, B. K. Williams, and J. A. Kent, “The Monte Carlo Approach to Mine Warfare,” *Navy League of the United States,* February 2001. [Online]. Available: http://www.navyleague.org/sea_power/feb_01_05.php

[38] P. Svenson, C. Martenson, “SB-Plan: Simulation-Based Support for Resource Allocation and Mission Planning*,” Proceedings of the Conference on Civil and Military Readiness (CIMI 2006)*, Enkoping, Sweden, 2006.

[39] B.J.H. van Basten, “Path Planning and Online Obstacle Avoidance in Weighted Regions,” Masters Thesis, Utrecht University, August 2006.

[40] X. Ning, S. Shihuang, F. Xizhou, “A Fuzzy Approach to the Weighted Region Problem for Autonomous Vehicles,” *Proceedings of the 1993 International Symposium on Intelligent Control*, Chicago, IL, August 1993.

[41] N. Motee, A. Jadbabaie, G. Pappas, “A Duality Theory for Path Planning for Multiple Vehilcles,” *IEEE Transactions on Robotics* (under review), 2011.

[42] C. Aggarwal, H. Wang, “Graph Data Management and Mining: A Survey of Algorithms and Applications,” *Managing and Mining Graph Data*, Advances in Database Systems 40, DOI 10.1007/978-1-4419-6045-0_2, Springerlink, 2010.

[43] G. Coombe, M. Harris, “Global Illumination Using Progressive Refinement Radiosity,” *GPU Gems 2*, Taunton, MA: Addison-Wesley, 2005, ch. 39.

[44] M. Cohen, S. Chen, J. Wallace, D. Greenberg, “A Progressive Refinement Approach to Fast Radiosity Image Generation,” *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, pp. 75-84, 1988.

[45] S. Nandy, R. Narayan, V. Visvanathan, P. Sadayappan, P. Chauhan, “A Parallel Progressive Refinement Image Rendering Algorithm on a Scalable Multithreaded VLSI Processor Array,” *International Conference on Parallel Processing, 1993. ICCP 1993*, vol. 3, pp. 94-97, Aug 1993.

[46] N. Holzschuch, F. Sillion, G. Drettakis, “An Efficient Progressive Refinement Strategy for Hierarchical Radiosity,” *5th Eurographics Workshop on Rendering*, 1994.

[47] R. Blanford, “Progressive Refinement Using Local Variance Estimators,” *IEEE Transactions on Communications*, vol. 41, no. 5, pp 749-759, 1993.

[48] R. Rosenbaum, H. Schumann, “Progressive Refinement: More than a Means to Overcome Limited Bandwidth,” *Proceedings of SPIE 7243*, San Jose, CA, 2009.

[49] K. Wagstaff, M. Kocurek, D. Mazzoni, B. Tang, “Progressive Refinement for Support Vector Machines,” *Data Mining and Knowledge Discovery*, 2009.

[50] J. Wang, M. Lewis, J. Gennari, “A Game Engine Based Simulation of the NIST Urban Search and Rescue Arenas,” *Proceedings of the 2003 Winter Simulation Conference*, vol. 1, pp. 1039-1045, 2003.

[51] C. Karr, “Conceptual Modeling in OneSAF Objective System (OOS),” *Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2005*, vol. 2005, no. 1, 2005.

[52] M. Morgan, M. Henrion, M. Small, *Uncertainty: A Guide to Dealing with Uncertainty in Quantitative Risk and Policy Analysis*, 1990, Cambridge University Press, 1990, pp. 277-278.

[53] F. Brooks, Jr., *Mythical Man-Month, The: Essays on Software Engineering, Anniversary Edition*, 1995, p. 118.

[54] B. Mitchell and L. Yilmaz, “Symbiotic Adaptive Multisimulation: An Autonomic Simulation Framework for Real-Time Decision Support Under Uncertainty,” *ACM Trans. Model. Comput. Simul.*, vol. 19, no. 1, pp. 1 – 31, 2008.

[55] Association for Computing Machinery, “The 1998 ACM Computing Classification System,” [Online]. Available: http://www.acm.org/about/class/1998/

[56] D. Shin, E. Chong, H. Siegel, “A Multiconstraint QoS Routing Scheme using the Depth-First Search Method with Limited Crankbacks,” *2001 IEEE Workshop on High Performance Switching and Routing*, pp. 385-389, 2001.

[57] S. Lim, S. Yu, M. Kang, C. Ham, “Robot Palletizing Simulation Using Heuristic Pattern Generation and Trajectory Optimization,” *SICE-ICASE, 2006, International Joint Conference*, Bexco, Busan, Korea, pp. 2227-2232, 2006.

[58] (February 2004), G. Kkyne, J. Carroll, “Resource Description Framework (RDF): Concepts and Abstract Syntax,” *W3C Recommendation* [Online]. Available: http://www.w3.org/TR/rdf-concepts/

[59] (2011, Feb. 12), Stanford Center for Biomedical Ontology, “Protégé” [Online]. Available: http://protege.stanford.edu/.

[60] B. Smith, C. Wety, “Ontology: Towards a New Synthesis,” *Proceedings of the International Conference on Formal Ontology in Information Systems* (FOIS2001). ACM Press, 2001

[61] (2011, Feb. 12), Franz, Inc. [Online]. Available: http://www.franz.com/

[62] (2011, Feb. 12), N. Walsh, “A Technical Introduction to XML,” *O’Reilly XML.com*, October 1998. Available: http://www.xml.com/pub/a/98/10/guide0.html

[63] G. Brassard, P. Bratley, *Fundamentals of Algorithms*. Englewood Cliffs, NJ: Prentice Hall, 1996, pp. 198 – 202.

[64] S. Russell, P. Norvig, *Artificial Intelligence A Modern Approach*. Upper Saddle River, NJ: Prentice Hall, 2003, pp. 94 – 105.

[65] S. Russell, P. Norvig, *Artificial Intelligence A Modern Approach*. Upper Saddle River, NJ: Prentice Hall, 2003, pp. 75 – 76.

[66] A. Bar-Zeev, "Scengraphs: Past, Present and Future," *Reality Prime*, 2003 [Online]. Available: http://www.realityprime.com/scenegraph.php

[67] Wikipedia contributors, "Scene Graph," *Wikipedia, The Free Encyclopedia,* 2007 Feb 1, 11:16 UTC [cited 2007 Feb 2]. Available: http://en.wikipedia.org/w/index.php?title=Scene_graph&oldid=104818809.

[68] P. Fishwick, *Simulation Model Design and Execution*. Englewood Cliffs, NJ: Prentice Hall, 1995, pp. 165-169.

BIOGRAPHICAL SKETCH

Teresa Nieten received her Bachelor of Science degree in computer and information science and engineering at the University of Florida in 1992, and her Master of Science in Computer Engineering at the University of Florida in 2004. She has worked on command and control systems in energy management, spaceflight, and robotics areas; simulation systems for mission planning and robotics; and telecommunications. She is currently a Senior Systems Engineer with Modus Operandi, Inc., in Melbourne, FL working with natural language processing. Teresa is married to fellow PhD student Daniel Nieten and has a son, Brandon.