Network-Centric Automated Planning and Execution

A Thesis

Submitted to the Faculty

of

Drexel University

by

Kyle Usbeck

in partial fulfillment of the

requirements for the degree

of

Master of Science in Computer Science

2009

Dedications

This thesis is dedicated to my parents, Frederick and Linda Usbeck. Without their loving support, this work would not have been possible.

Acknowledgements

First, I would like to thank my advisor and mentor, Dr. William C. Regli, for truly starting my interest in Computer Science and patiently teaching me at every step of my academic life. I also thank my committee, Dr. Rachel Greenstadt and Dr. Ani Hsieh, for their insight and advice.

Next, I would like to thank the faculty of Drexel University for my strong foundation in computer science. Also, I give thanks to Prof. Austin Tate, Dr. Gerhard Wickler, and Jeffery Dalton from the University of Edinburgh for providing the intelligent agent framework I use in my implementations and feedback at the beginning of this project.

Also, many people have reviewed portions of this work and offered helpful advice. These people include Ilya Braude, Matthew Chase, Patrick Freestone, Joseph Kopena, Duc Nguyen, Robert Lass, Evan Sultanik and Mary Wertz. I would like to thank them as well as my family and friends for their support during the creation of this thesis.

Finally, I thank the authors of TEX, LATEX, BIBTEX, Vim, and Gnuplot. Without these and other Open-Source software, the presentation quality of this thesis would have suffered.

Table of Contents

List of Tables

List of Figures

Abstract

Network-Centric Automated Planning and Execution

Kyle Usbeck

Advisor: William C. Regli, Ph. D.

Web services provide interoperability to network hosts with different capabilities. Complex tasks can be performed by composing services, assuming sufficient service descriptions are provided. Researchers are just beginning to realize the importance of accounting for network properties during automated service composition. The work presented in this thesis considers dynamic, heterogeneous networks — one type of network-centric environment.

The purpose of this research is to improve network-centric service composition. This is accomplished by converting the service composition problem to an automated planning under uncertainty problem and by reasoning about network properties at various stages of the planning process. This thesis presents a method of improving the agents' ability to construct, execute, and monitor plans in network-centric environments.

There are two main contributions of this thesis: 1) generating qualitatively-different plans and 2) creating network-aware agents. As part of the former contribution, this thesis presents a comparison of methods used to create classical planning domains for distributed service composition problems. The other part of this contribution is an algorithm for guiding a plan-space planner to create qualitatively-different plans based on domain-dependent and network-centric plan evaluations. The second contribution pertains to network-awareness, which agents exhibit by reacting to changes in network conditions. This thesis describes methods of incorporating network-awareness into agents that 1) create plans, 2) execute plans, and 3) monitor plan execution.

Experiments to validate the aforementioned contributions are presented in the context of

an Improvised Explosive Device (IED) detection scenario. Several locations are monitored for IEDs using a variety of techniques including manual searching and visual change detection, as well as a variety of resources including humans, robots, and unmanned aerial vehicles (UAVs). Empirical results indicate that incorporating network-awareness into agents in dynamic, heterogeneous networks improves the overall service composition performance and effectiveness.

1. Introduction

Researchers are just beginning to realize the importance of accounting for network properties during automated service composition. In heterogeneous networks, such as the one depicted in Figure 1.1, there are multiple different network technologies — satellite communication, mobile ad-hoc networks (MANETs), wired Ethernet, *etc.* — combined to work together simultaneously [78]. One method for performing automated service composition in a heterogeneous network is to convert the services to actions in an automated planning problem. By using this method, dynamic network properties are represented in the automated planning world model (domain) and network properties are represented by plan evaluators. These evaluators are functions that express the desirability of a situation or action, and represent the concerns of the end user.

This thesis compares classical planning methods for modeling distributed service composition. Also, it presents extensions necessary for automated planners to reason about service composition in dynamic, heterogeneous networks. The extensions include reusable plan evaluators for network concerns and a novel method of generating qualitatively-different plans over a range of plan evaluators.

There are three stages in the automated planning process: 1) planning, 2) execution, and 3) monitoring. In the planning stage, plans (sets of actions) are created; during execution, the actions of a plan are performed; and in the monitoring stage, execution is overseen to detect errors during action execution. This thesis shows which stage(s) of the automated planning process benefit most from reasoning about network properties.

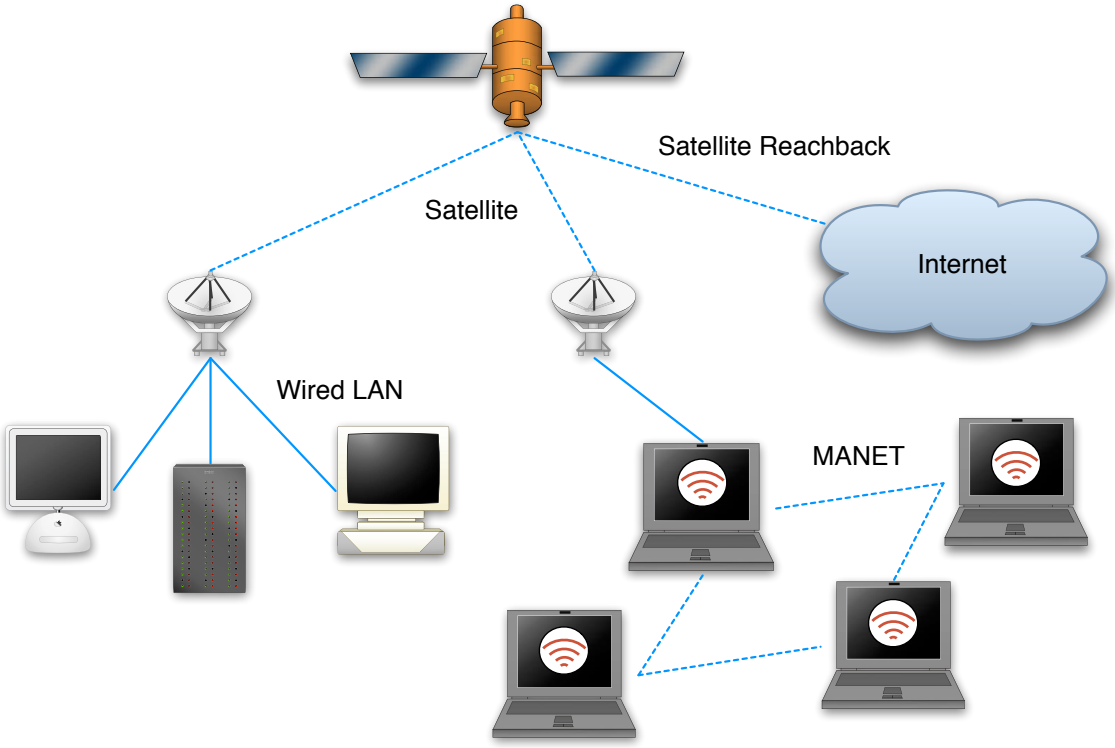

Figure 1.1: Example of a heterogeneous network. In this type of network, multiple different technologies (satellite communication, MANET, and wired Ethernet) are combined.

1.1 Motivation

In current warfare, many casualties occur due to concealed landmines and roadside bombs, called Improvised Explosive Devices (IEDs) [41]. As new techniques of performing IED detection arise, a methodology for choosing among IED detection techniques to perform is necessary.

In the motivating scenario, there are several locations that must be monitored for IEDs. Location monitoring can be accomplished via two different techniques: manual searching and visual change detection. Also, different actors such as humans, robots, and unmanned aerial vehicles (UAVs), are capable of performing the IED monitoring techniques.

Since each monitoring technique requires different resources (time, money, humans,

robots, UAVs, *etc.*) and yields different results, no single technique is always appropriate. The quality of each IED detection method and the combinations of methods used to monitor all of the locations can be quantitatively expressed by a number of evaluators. For instance, an evaluator exists for the amount of time required to complete the location monitoring. Another evaluator exists for the monetary transportation costs associated with moving resources to various locations.

The IED detection scenario is illustrated in Figure 1.2. Red rectangles represent physical locations that require IED monitoring — an IED has been planted in location 2. Humans, robots, and UAVs are the entities that are capable of performing IED detection techniques. Cameras represent the geographical positioning of resources required to perform visual change detection. The goal is to minimize the overall evaluation cost where that cost can be affected not only by the selected monitoring techniques, but also the entities that perform the techniques, the order the techniques are performed, and the resources utilized.

The IED detection scenario can be viewed as an instance of a *web service composition* problem where the monitoring techniques are high-level services. Hoffmann *et al*. [33] defines web service composition as the "linking. . . of existing services so that their aggregate behavior is that of a desired service (the goal)." Each service, when invoked, causes a host to perform a task, such as manually searching for IEDs at a certain location.

Assuring Quality of Service (QoS) over web service compositions is studied to some degree by Gu *et al*. [29], but the authors do not consider varying solution qualities — service QoS is not changing during runtime. Furthermore, Gu *et al*. only consider *availability* and *response time* as QoS metrics, both of which can vary greatly in dynamic networks.

By describing the services using a semantic web service framework, such as OWL-S [77] or WSMO [58], we can determine sets of monitoring combinations to solve this problem using automated composition of semantic web services. Sirin *et al*. [60] and Hoffmann *et al*. [33] map the automated composition of semantic web services to an automated planning

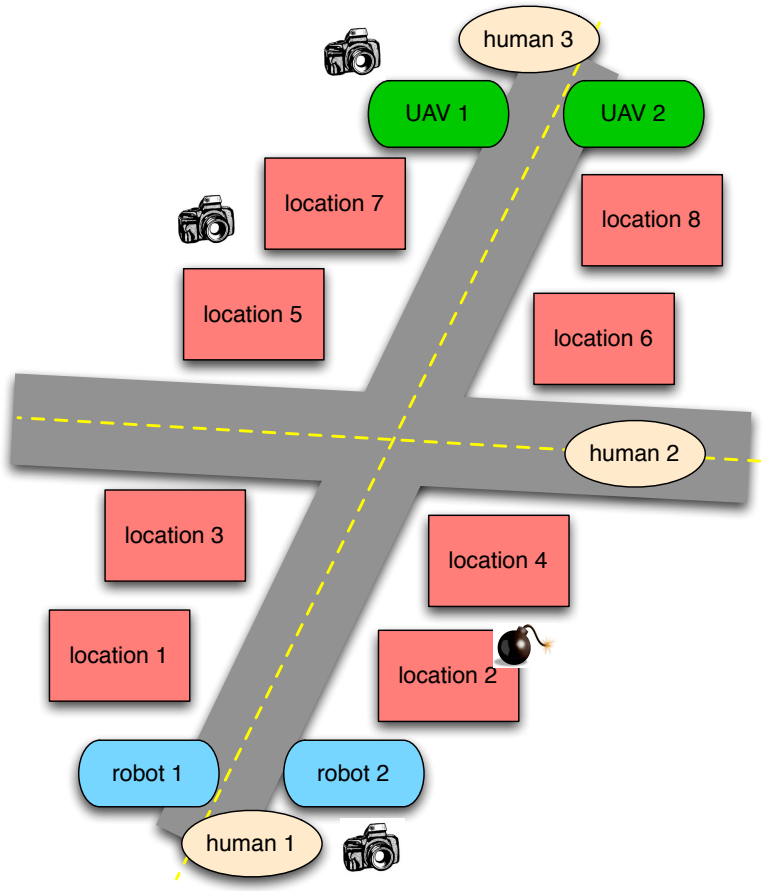

Figure 1.2: The IED detection scenario. Several locations are monitored for IEDs using manual searching or visual change detection.

problem. In what OWL-S labels the "service profile" level and WSMO labels the "service capability" level, web services are described with preconditions and effects, analogous to those in planning operators. The semantic service descriptions provided by OWL-S and WSMO make it possible to perform automated composition of services to accomplish an end user's goal. This still does not solve the problem completely because of the uncertainty in action execution — a host may never receive orders to monitor an area due to communication failures. Also, a current research area is representing the dynamic, heterogeneous network in the automated planning model.

Definition 1. A *network-centric system* is a distributed system where performance is dependent on the quality of the underlying network communication links.

Studies such as [54] show agents that reason over network properties (labeled *network-aware*) in MANET environments can drastically improve system performance over their network-unaware counterparts. Incorporating network-awareness into the automated planning problem would provide a way to represent the dynamic, heterogeneous network, however it is not clear how to accomplish this task or what network properties to model. The network characteristics that Peysakhov *et al*. use for network-awareness are signal strength, signal-to-noise ratio, delay, jitter, and routing. These and other link properties are considered when defining my network-centric model.

This thesis expands on the work of Sirin *et al*. [60] and Hoffmann *et al*. [33] in mapping the service composition problem to an automated planning problem. Furthermore, I define a set of plan evaluations for network-centric systems. The thesis draws on ideas from Myers *et al*. [46] in defining a new method of qualitatively-different plan generation based on plan evaluations. Finally, I adapt the ideas from Peysakhov *et al*. [54] to incorporate network-awareness into the agents involved in creating plans, executing plans, and monitoring plan execution.

1.2 Approach

My approach to solving the network-centric service composition problem is to represent it as an automated planning problem. To conduct automated planning and execution in network-centric environments, I first formalize a distributed, agent-based service composition problem. This formalization includes methods of distributing services in automated planning domains and my approach to guiding the search strategy of a plan-space automated planner. Next, I explain methods of reasoning over network properties in planning, plan execution, and execution monitoring agents. Finally, empirical results are collected to

compare the effects of agent implementations that reason about network properties during different stages of execution.

The contributions of this thesis are as follows:

1. A comparison of methods for creating classical planning domains for distributed service composition problems;
2. Reusable network-centric plan evaluators for planning agents operating in dynamic, heterogeneous networks;
3. Modifications for a plan-space automated planner to generate qualitatively-different plans over a range of plan evaluation criteria; and
4. A comparison of the efficiency and performance of network-aware planning, execution, and monitoring agents.

The purpose of this research is to improve the ability of agents to construct and execute plans in network-centric environments. In doing so, I will explain how to reason over and react to uncertainty at different stages of the planning and plan execution process.

1.3 Organization

The thesis is organized as follows:

- Chapter 2 provides background on research in areas related to the concepts of this thesis;
- Chapter 3 formalizes my problem statement;
- Chapter 4 describes my approach towards improving plan construction, execution, and monitoring agents;
- Chapter 5 describes experimental results, evaluating the modified planning algorithm and comparing each agent's performance and effectiveness in different network environments; and

- Chapter 6 presents conclusions and plans for future work.

2. Background

2.1 Planning Notation

The following notation will be used throughout the thesis. In Section 3.1, the notation is extended.

$\mathcal{P}$ is the planning problem,

Σ is the planning domain,

s_0 is the initial state, and

S_g is the set of goal states.

S is the set of states in Σ;

A is the set of actions, $\{a_0, a_1, \ldots, a_{|A|}\}$ in Σ, where each $a = (\text{precond}(a), \text{effects}^-(a), \text{effects}^+(a))$;

E is the set of system events in Σ;

γ is the state transition function;

$\text{precond}(a)$ returns the preconditions of action, a;

$\text{effects}^+(a)$ returns the positive effects of action, a;

$\text{effects}^-(a)$ returns the negative effects of action, a;

2.2 Introduction to Planning

Planning is the explicit deliberation process that chooses and organizes actions by anticipating their outcomes.

Several factors influence the motivation for planning:

- complexity of tasks or objectives;
- joint activities with other agents; and
- risk or cost associated with tasks or objectives.

Planning is useful in complex situations because the interactions of actions and resources may be too complicated to find an efficient solution otherwise. One such complication, agent coordination, arises very often. For example, if Bill and Sue want cereal for tomorrow's breakfast, they might plan which person is responsible for which items. Because the grocery store is close to Sue's workplace, she picks up milk and cereal on her way home from work. Bill, then, is responsible for making sure there are clean bowls and spoons after dinner tonight.

Also, risk and cost can often be avoided by planning. For instance, when driving cross-country, planning a route can help avoid a wrong turn or a less-than-optimal road (which cost time and money in the form of fuel).

However, since planning is a "very complicated, time-consuming, and costly processes" it is useful to perform a cost/benefit analysis to determine its value [49].

In studying artificial intelligence, planning is the computational study of the general deliberation process. The remainder of this section will discuss the actors in the planning process and formalize automated planning. Furthermore, this section discusses how the planning problem can be extended to include uncertainty in the planning and execution process.

2.2.1 Role of Agents in Planning

Tate [68] discusses the roles of agents in the planning process. Tate defines three key agent roles: *Task Assignment*, *Planning*, and *Execution*. In this notion, the planning agent is responsible for solving a static planning problem and passing the plan to the execution agent. The execution agent interacts with the real system, and in some situations, can react to some action execution failures. The task assignment agent communicates with the planning and execution agents to trigger plan creation and execution respectively.

The agent interactions in [68] form a control theoretic feedback loop. A feedback loop

contains a *controller* which initially accepts a plan and then gives input to some *system*, or plant. A *sensor* component determines the current state of the system to help guide the controller in the remaining execution steps. In this terminology the controller functions as the task assignment and execution agent — it should be noted that these operations can be separated. Figure 2.1 combines Tate's agent roles with the feedback loop of agents in planning systems.

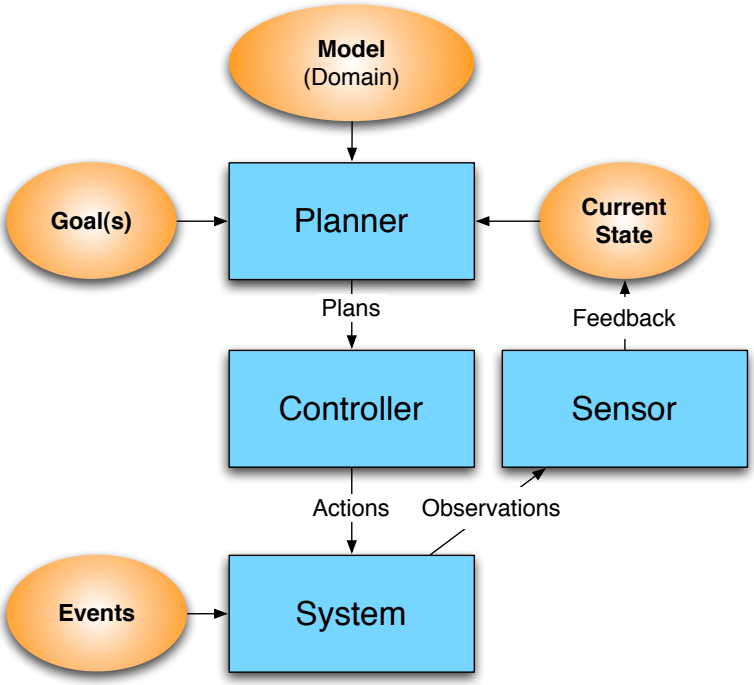

Figure 2.1: The interactions between agents in a planning architecture.

2.2.2 Classical Planning

Classical Planning the generic term for planning using restricted state transition systems. In this section, I formalize classical planning and introduce several representations.

Classical Planning models the planning domain, Σ, as a state transition system such that $\Sigma = (S, A, E, \gamma)$ where S is the set of states, A is the set of actions, E is the set of events, and $\gamma = S \times (A \cup E) \rightarrow 2^S$.

The planning problem, $\mathcal{P}$, can in turn be expressed as the triple (Σ, s_0, S_g) where s_0 is the initial state and S_g is a set of goal states.

According to [49], there are three ways to represent classical planning:

- Set-theoretic Representation,
- Classical Representation, and
- State-variable Representation.

The differences between these representations come in how they choose to model symbols and operators in the domain. These representations are described in the next sections.

Set-theoretic Representation

The set-theoretic representation models symbols as propositions. For example a state where a person is at location might look like the following:

```
{ atLoc1 }
```

Operators are therefore represented in propositional logic written action $a = (\text{precond}(a), \text{effects}^-(a), \text{effects}^+(a))$ where precond defines a set of propositions that must be true for the action to be applied, effects$^-$ lists the propositions that become false after an action is applied, and effects$^+$ lists the propositions that become true after an action is applied. Thus, an action to move the person would then look like:

```
move = ( { atLoc1 }, { atLoc1 }, { atLoc2 } )
```

Classical Representation

In contrast, classical representation uses a derivative of first-order logic and planning operators to model the problem. A state is represented as a list of first-order atoms, for example:

```
at( person1, location1 )
```

A planning operator is described by a name, preconditions, and effects. In this representation, preconditions and effects are sets of literals.

```
move( p, l1, l2 )
  ;; moves a person, p, from location, l1, to location, l2
  precond: at( p, l1 ), adjacent( l1, l2 )
  effects: at( p, l2 )
```

State-variable Representation

An expressively equivalent representation to classical, the state-variable representation uses *functions*, rather than relations. In this modeling scheme, a state might look as follows:

```
{ loc(p) = l1 }
```

An action, then is modeled as follows:

```
move( p, l1, l2 )
  ;; moves a person, p, from location, l1, to location, l2
  precond: at( p ) = l1, adjacent( l1, l2 )
  effects: at( p ) = l2
```

PDDL

The planning domain and problem can be represented in a number of different ways, one of the most common being the Planning Domain Definition Language (PDDL) [25]. PDDL was originally created as a standard for international planning competitions.

PDDL allows for pluggable extentions which enhance the expressivity of the language. Examples include extensions for conditional operators, HTN actions, and temporal planning. The base PDDL specification allows expression of object types, predicates, and actions.

Assumptions

The conceptual model and representations discussed make certain restrictive assumptions. These assumptions are meant to reduce the complexity of the planning problem. The *restrictive model* is the name given to the model that makes all of the assumptions discussed in the rest of this section.

Finite Domain. There are a finite number of states in Σ. This assumption ensures that the state transition graph is finite. If S is finite or the members of S can be enumerated by some algorithm (S is recursively enumerable), then states can be represented in first-order predicate logic, and planning is decidable. Relaxing this assumption allows for new objects to be created during the plan and continuous datasets (e.g. $\mathbb{R}$) to be considered.

Fully Observable Domain. The planner has complete knowledge of the domain and the controller's observability function, η, is the identity function. Sometimes, based on the domain, making this restrictive assumption is impossible. Relaxing the assumption is useful when not every aspect of the domain can be known by every agent.

Deterministic Domain. γ specifies that each action $\in A$ changes the state transition system to a single state. Formally, $\forall s \in S, u \in A \cup E : |\gamma(s,u)| \leq 1$. Relaxing this assumption allows for reasoning over non-deterministic actions.

Static Domain. This assumption states that $E = \emptyset$. In many systems, however, external events are unavoidable. These events cannot be controlled, but may need to be considered when planning in some domains. For example, the weather can restrict applicable actions, but cannot be voluntarily changed.

Restricted Goals. This assumption states that goals are restricted to a set of states such that if the system enters one of these states, it has reached its goal. Relaxing this assumption allows the planner to handle constraints on states, plans, and cost/utility.

Sequential Plans. A solution plan is a linearly ordered, finite sequence of actions. This restriction simplifies the data structure for describing plans. By relaxing this assumption, certain (more complex) plan structures can be better described. Relaxing this assumption allows for concurrent action execution.

Implicit Time. Actions and events have no duration in the state transition system. In other words, state transitions are instantaneous and there is no explicit temporal representation. Relaxing this assumption allows reasoning about action duration and concurrency at the cost of increased planning complexity.

Offline Planning. The planner is not concerned with changes in Σ while it is constructing a plan. This assumption limits the applicability of the plan, since, if the domain changes while the planner is constructing a plan, the plan may no longer be valid. In domains where planning occurs as an online activity, this restriction can be relaxed.

Relaxing Assumptions

The restrictive assumptions are made to reduce the complexity of the planning problem. The restrictive model, however, does not account for many of the real-world problems that might occur. For instance, it is impossible to represent any action with non-deterministic effects.

Relaxing the assumptions in the restrictive model allows for more realistic domain modeling. The caveat is that the realism in the modeling comes at the cost of computational complexity. *Planning under uncertainty* is a branch of planning research that relaxes these assumptions, particularly determinism, full observability, and reachability goals. See Section 2.5 for more information.

2.2.3 Solving the Planning Problem

There are two classical methods for solving the planning problem: state-space and plan-space. The method of exploring the search space is one important factor in the performance and result of the planner [49].

Starting at state s_0, a state-space planner uses γ to add new states to the current list of states. Similarly, a state-space planner can start at a goal state $s_g \in S_g$ and work backward.

Plan-space planning takes a different approach, exploring search space of partial plans. [49] defines partial plans as a subset of actions that keep some "useful part of this structure." This is different from state-space planning in that state-space partial plans are sequentially, totally ordered. To eliminate these constraints, a plan-space planner maintains lists of ordering constraints, causal links (why an action was applied), and variable binding constraints (what values a variable can/cannot take).

The differences between plan-space and state-space algorithms yield some interesting properties. Plan-space planning, unlike state-space planning, is infinite with a finite number of states. Also, plan-space planners do not necessarily produce intermediate states, and

the refinement operations for plan-space planners take significantly longer to compute than state transitions. There are, however, a considerable number of benefits to plan-space planners over state-space. For example, the plans produced by plan-space planners are more flexible for execution. Also, plan-space planners provide a simple approach to handling classical extensions such as time, resource, and information gathering actions. Furthermore, distributed planning and multiagent planning can be addressed very naturally with plan-space partial plan structures.

Neoclassical planning (or disjunctive-refinement approaches to planning) arose as a way to more efficiently solve large planning problems. This approach to planning relies on solutions to similarly difficult problems to simplify or solve the planning problem. One technique, labeled *planning-graph*, uses graph node reachability to solve the planning problem. Others convert the planning problem into propositional satisfiability (SAT) and constraint satisfaction problems (CSP) — using research from these respective fields to solve the problems.

The general search algorithm according to [49] involves four steps: refinement, branching, pruning, and select. If these operations organize and guide the search well, the performance and efficiency of the planner can be drastically increased. To do this, heuristic techniques are used to guide the search. Domain-independent heuristics can be used with any planning domain, whereas domain-dependent (or domain-specific) are "tailor-made" for a particular domain. The extra effort required to create domain-dependent heuristics is justified by the use of domain-specific heuristics, or even domain-specific algorithms in most real-world planners according to [49].

2.2.4 Hierarchical Task Network Planning

Hierarchical Task Network (HTN) planning differs from classical planning in that classical planning's objective is to achieve a set of goals whereas HTN planning's objective is

to perform a set of tasks. Furthermore, tasks can be compositions of subtasks, subtasks can be composed of smaller subtasks, and so on until primitive tasks are reached and the planning operators can be performed directly.

Several real-world examples of domain-independent HTN planning systems are:

- **Nonlin** [67], one of the first HTN planning systems;
- **SIPE-2** [74], used in many application domains;
- **UMCP** [16], the first provably sound and complete HTN planning algorithm;
- **SHOP2** [48], developed at University of Maryland, this planner serves an active base of users in government, industry, and academia;
- **O-Plan** [11, 65], a planner developed throughout the 1980s and 1990s used on many applications, and still available for download and as a web-based planning service; and
- **I-Plan**, based on O-Plan and described in Section 2.3.1.

2.2.5 Scheduling

Although scheduling is discussed in the same context as planning, the processes have different goals. One can think of planning as the process of selecting and ordering actions from a domain, whereas scheduling is the process of assigning temporal intervals to the actions of a plan. Figure 2.2 shows the agents involved in the scheduling process.

What makes the distinction between planning and scheduling even less clear is that the addition of temporal constraints is handled by the planner rather than the scheduler. The temporal constraints define the level to which actions from the plan must be complete relative to the execution status of other actions. For instance, a temporal constraint may state *action A must be fully completed and action B must be started for action C to begin execution*. Figure 2.3 shows the visualization of temporal constraints on plan actions. In contrast, an example of the scheduler output would be *action C will be executed at time t*.

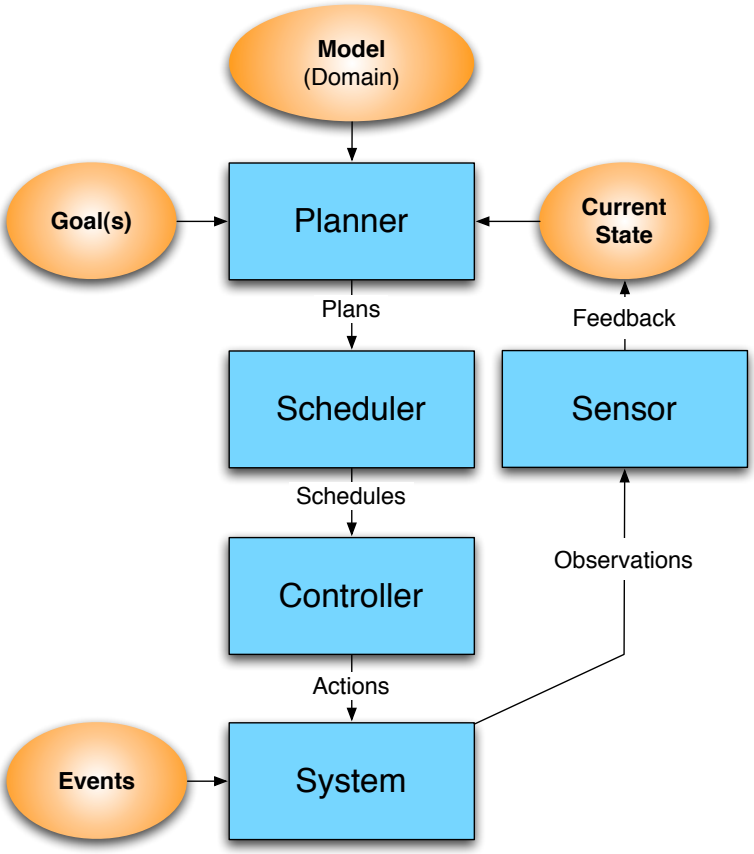

Figure 2.2: The agents involved in the scheduling process.

2.3 Planners: Examples and Architectures

According to [11], examples of heavily studied automated planners include:

- **HSP** [5], performs a forward, state-space, heuristic search;
- **PLANEX** [19], conducts reactive planning by inserting monitoring actions into plan execution;
- **Metric-FF** [32], uses graph-plan plan length as a heuristic in a hill-climbing state-space search;

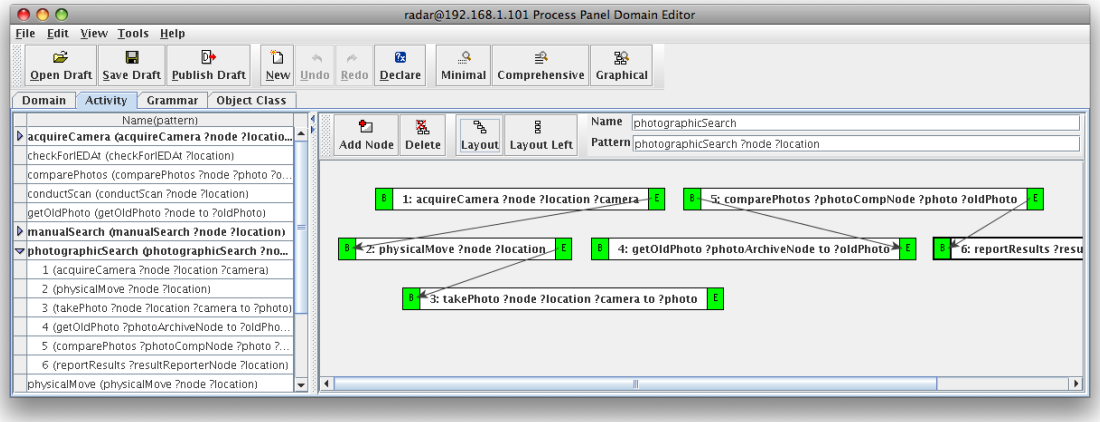

Figure 2.3: Temporal constraints on plan actions as visualized by the "Domain Editor" of I-X.

- **SIPE** [73, 76] and **SIPE-2** [46, 74], provides a GUI for mixed-initiative plan selection;
- **UCPOP** [51], uses a sound and complete partial-order planning algorithm; and
- **Nonlin** [67], conducts domain-independent, partial-order, hierarchical task network planning.

O-Plan is a plan-space HTN planning agent written in Lisp. One contribution of O-Plan was the introduction of the <I-N-OVA> constraint model of activity. <I-N-OVA> stands for *Issues*, *Nodes*, *Orderings/Variables/Auxiliary*, which is a method of representing plans as sets of constraints. During the development of I-X, the <I-N-OVA> model was changed to <I-N-C-A> (discussed in Section 2.3.1).

Another contribution of O-Plan was the separation of planning, task assignment, and execution agents (introduced in Section 2.2.1). This separation led to the development of the I-X framework and a successor to O-Plan, I-Plan.

2.3.1 I-X/I-Plan

As described in [69], I-X is a framework with a number of different aspects intended to create a well-founded approach to allow humans and computer systems to cooperate in the creation or modification of some product such as a design, physical entity or plan — i.e., it supports cooperative synthesis tasks. The I-X approach involves the use of shared models for task-directed cooperation between human and computer agents who are jointly exploring (via some, perhaps dynamically determined, process) a range of alternative options for the synthesis of an artifact such as a design or a plan (termed a product).

I-X represents a product as a set of nodes making up the components of the product model, along with constraints on the relationship between those nodes, and a set of outstanding issues. This representation is labeled <I-N-C-A>, which represents Issues, Nodes, (Critical and Auxiliary) Constraints and Annotations, and is the successor of the <I-N-OVA>model.

Within the I-X framework is an architecture, I-Plan, in which situated agents, such as planning agents, can be created. Based on O-Plan, I-Plan uses the same conceptual separation of command (task assignment), planning, and execution monitoring roles. Another technique shared by both I-Plan and O-Plan is a hierarchical planning system which can produce plans as partial orders on actions.

I-Plan is a plan-space HTN planner which uses the I-X framework to represent the domain and world state. A description of I-Plan's planning algorithm can be found in Section 4.4.1.

2.4 Plan Optimization

Classical planning attempts to find a sequence of actions that *satisfies* the classical planning problem. By relaxing the "Restricted Goals" assumption from Section 2.2.2, cost/utility can be incorporated into the satisfaction problem. This allows a lower-bound

on the utility of a plan to be specified.

In certain cases, however, it is more appropriate to find a plan with the highest utility, rather than a plan that satisfies all the necessary requirements. Changing the planning problem from satisfiability to domain-dependent optimization involves the incorporation of preferences or soft constraints into the planning domain.

2.4.1 Plan Metrics

According to [22], "Plan metrics specify, for the benefit of the planner, the basis on which a plan will be evaluated for a particular problem." Plan metrics were added to the PDDL specification (see Section 2.2.2) in version 2.1.

The addition of plan metrics has several implications on the planning problem. Adding the notion of domain-dependent cost/utility is vital in practical planning since users seek to find the "best" plan rather than any satisfying plan.

One use of plan metrics is as a mechanism for evaluating plans. [46], for example uses domain metatheory as a mechanism for generating qualitatively-different plans. Similarly, [71] uses multi-dimensional plan evaluation criteria as a way to present dominant plans to task assignment agents.

Plan metrics can also be used as a way of evaluating planners themselves. [79] warns against this practice, stating that

> Planning should be viewed as a resource-bounded reasoning activity that provides useful information to an execution architecture in selecting actions. The overall utility of a planner is determined by the effect it has on the agent behavior.

Plan metrics can, however, be used to guide a planner toward "better" solutions. [5] discusses the success of heuristic-search planners in international planning competitions

(IPC/AIPS). Of the planners discussed in Section 2.3, some of the best performing are FF and HSP.

One difficult aspect when using heuristic-search planners is choosing the heuristic function. Several papers [44, 6] aim to extract heuristics from declarative problem representations.

2.4.2 Preference-based Planning

The International Conference on Automated Planning and Scheduling (ICAPS) is a forum for researchers and practitioners in planning and scheduling. In 2006, ICAPS hosted a workshop on "preferences and soft constraints in planning." Until then, "with exception of MDPs, nontrivial user preferences have only recently been integrated into AI automated planning" [2]. Since then Preference-Based Planning (PBP) is a large research area in AI.

In PBP, a "criterion to determine when a plan is preferred to another" is provided. PBP is closely related to oversubscription planning (described in [61]), but more general because it includes soft constraints. The definitions of some key concepts in PBP follow.

Definition 2. *Soft problem goals* are goals that are desirable, but that do not necessarily have to be achieved [2]. For example, if a primary goal is to go to bed, one soft goal might be to obtain another pillow.

Definition 3. *Soft constraints* on plan trajectories are constraints over possible actions in the plan and intermediate states reached by the plan [2]. For example, if two actions can be executed simultaneously, but are preferably executed serially, a soft temporal constraint may be added.

Definition 4. *Preferences* include both soft goals and soft constraints [72]. [26] explains that preferences are goals that don't have to be satisfied, in contrast to plan metrics from PDDL 2.1 which associate weighted expressions with utility (see Section 2.4.1).

State trajectory constraints describe temporal control knowledge and temporally extended goals [15].

PBP is also related to *partial satisfaction planning* (PSP). In PSP, the net benefit of a plan (P) is a function of the goals that it satisfies and the costs incurred.

$$\text{NETBENEFIT}(P) = \sum \text{GOALSREACHED}(P) - \sum \text{COST}(P)$$

This method neglects some of the intricacies of goal/cost interactions, but has the same runtime complexity as classical planning [2].

For PBP, a language that can sufficiently express the definition of preferences as well as the aggregation of the preferences is required. These languages can be quantitative, qualitative, or some hybrid [2]. Quantitative languages include those used in decision-theoretic planning (discussed in [8] and Section 2.5.2), partial satisfaction planning (discussed in Section 2.4.2), and PDDL version 3 (discussed in [22] and Section 2.4.1).

Algorithms for solving PBP are classified along three properties: optimal, k-optimal, and incremental. An optimal algorithm eventually outputs an optimal plan. K-optimality is the property that, given a positive integer k, the algorithm outputs an optimal plan whose `makespan` is at most k, where the `makespan` is the minimum amount of time in which we can perform the plan. An incremental algorithm is one where each plan output is better than the previous. This property does not necessarily imply an "anytime" algorithm unless there are no hard goals.

2.4.3 Dominant Plans

A goal of this project is to help agents choose between multiple plan options. One way to accomplish this is by distinguishing dominant plans from those that are dominated [71].

To define the notion of *plan dominance*, we look to the field of multicriteria decision making. Multi-objective optimization (MOO) examines the problem of choosing a strategy

from a list of possible strategies, $s_i \in \mathcal{S}$, given a set of utility functions $u_j(s) \in U$. The first step in solving MOO is to find the *Pareto points* in the multi-objective problem (MOP). Multiple Pareto points can exist for each combination of utility functions. The final step of MOO chooses between the strategies represented by Pareto points [45].

Definition 5. A plan, p, is **dominant** to other plans, P^- in respect to two or more plan evaluators $e_{1...k} \in E$ when $\forall e \in E, p^- \in P^-[e(p) \geq e(p^-)]$.

This definition corresponds to the Pareto points in the MOP rather than the result of the MOO. The tradeoffs between plan evaluators described by the dominant plans are presented for selection to the execution agent.

2.5 Planning Under Uncertainty

Some of the assumptions from Section 2.2.2 commonly need relaxation to be used in practical planning. In real life, actions do not have completely deterministic effects; agents cannot be completely omniscient; and goals are more complex than sets of states. The consequence of relaxing the corresponding assumptions, determinism, full observability, and reachability goals, is labeled *planning under uncertainty* [49].

The aim of *planning under uncertainty* is to design AI planners for environments where there may be incomplete or faulty information, where actions may not always have the same results and where there may be tradeoffs between the different possible outcomes of a plan [4].

According to [53], there are four sources of uncertainty in planning. These are as follows:

- missing information due to partial observability;
- unreliable resources, for example faulty equipment;
- stochastic phenomena as seen in all measurement variance; and

- inherently vague concepts as a result of poor domain/problem modeling.

There are several categories of approaches towards managing uncertainty. The approaches vary in the times and methods of reasoning over relaxed assumptions. Some approaches, such as probabilistic planning, decision-theoretic planning, and contingency planning, choose to reason over uncertainty at planning-time. Other approaches, such as reactive planning, react to uncertainty at execution-time. Still other approaches interleave the functions of planning and execution agents in an attempt to cope with relaxed assumptions.

2.5.1 Uncertainty in the IED detection scenario

In the IED detection scenario, uncertainty exists in several forms. First, the goal is to find the "best" solution, rather than "any" solution. This requires relaxation of the *reachability goals* assumption. Furthermore, networking conditions (firewalls or MANET partitioning) could limit communication between certain agents. Therefore, the *full observability* assumption is unrealistic and must be relaxed. Yet another unrealistic assumption for this domain is *static domain*. The environment can be changed by a variety of factors, not all of which can be modeled, so the domain can change. A realistic model of the IED detection scenario is possible when these assumptions are relaxed.

The following sections discuss the approaches to planning under uncertainty, as well as their advantages and disadvantages.

2.5.2 Decision-Theoretic Planning

According to [34], "Decision theory is based on the axioms of probability and utility." The purpose of decision theory is to formalize decision making under uncertainty. "Where probability theory provides a framework for coherent assignment of beliefs with incomplete information, utility theory introduces a set of principles for consistency among preferences

and decisions." Here Horvitz *et al*. consider decisions to be ground action instances and preferences represent the decision maker's utility in the outcome states. Note that preferences compare resultant states rather than actions, since the goal of decision theory is to elicit better *outcomes* on average [34].

Feldman and Sproull [17] are one of the first to investigate the combination of decision theory with artificial intelligence. The reason for combining these fields is that decision theory adds expressivity to symbolic reasoning. The intersection of these two fields results in representations with increased expressiveness. Also, decision theory improves the ubiquity of planning, allowing more domains to be represented in traditional planning representations. Lastly, decision theory lays the groundwork for provably optimal planning. Decision-theory, as applied to AI, extends the problem formulations to allow for statements about alternative actions and their resultant valuations.

> Given a probability distribution over the possible outcomes of an action in any state, and a reasonable preference function over outcomes, we can define a utility function on outcomes such that whenever the agent would prefer one plan over another, the preferred plan has higher expected utility. The task of the planner (is) to find the plan with the maximum expected utility (MEU) [17].

2.5.3 Probabilistic Planning

Probabilistic planners use information about the probabilities of the possible uncertain outcomes to construct plans that are likely to succeed. According to [38], probabilistic planning can further be split into approaches that represent plans as Markov Decision Processes (MDPs) and Symbolic Planning Approaches.

Koenig [37] explains that MDPs are useful for solving probabilistic and decision-theoretic planning problems. Markovian decision-theoretic planning algorithms work by producing reaction strategies rather than sequences of actions. The idea is to restrict the

planner's attention to "a set of world states that are likely to be encountered in satisfying the goal" [12]. MDP representations are best utilized by execution agents that can determine world state with some accuracy, whereas symbolic planning is better utilized by stateless execution agents [38].

Symbolic planning is another type of probabilistic planning that varies the way that the domain and planning problem are represented. The algorithms presented in [13, 28] take the probability distributions of world-states as input and return plans that make the goal condition true with no less than a given probability threshold. Other approaches, such as [43, 14] merge plans created from every possible initial-state.

Riley and Veloso [57] examines probabilistic planning as a way to conduct multi-agent planning in a Robo-Soccer domain. The execution agents have very limited observability, so failure detection is conducted via temporal constraint violations. Plans are based on opponent behavior models and the work focuses on using the recognized information for purposes of predicting and adapting to future behavior.

2.5.4 Interleaving Planning and Execution: Contingency Planning

Where decision theoretic planning reasons about uncertainty at plan-time, contingency planning deals with uncertainty by interleaving the planning and execution agents [73, 76].

According to [56, 55], contingency plans are needed when there is uncertainty in the world such that the planner cannot make a decision in advance. Pryor and Collins include decision points in the plans they communicate the the execution agent. Before the decision points, the planner inserts information-gathering actions, whose preconditions include goals to obtain the information needed for the decision step.

This form of planning selects actions based on the results of executing other actions. "It thus effectively splits the plan into a set of branches, one for each possible outcome of the uncertainty" [56]. [13] labels this symbolic planning approach *contingency planning*.

This approach only works if the execution agent has some way of monitoring or sensing the world state.

Combining probabilistic planning and contingency planning is the resulting ability to judge whether it is worth planning for a given contingency [56]. One type of contingency planning, *conformant planning* is the problem of finding a sequence of actions that is guaranteed to achieve the goal for any possible initial state and nondeterministic behavior of the planning domain. This approach is designed for execution agents that have no way of sensing the environment [10].

2.5.5 Reactive Planning

In reactive planning, no specific sequence of actions is planned in advance. Instead the planner produces a set of condition-action rules, for example, Universal Plans [59] or Situated Control Rules (SCRs) [14].

When utilizing reactive planning, more reasoning is required at execution time to use reaction rules than is required to execute a contingency plan. In completely reactive planning, [20] "describes an investigation into reactive planning that takes the extreme position of using no prediction of future states at all. Plan selection is done entirely at execution time and is based only on the situation existing then." Suchman labels this approach *Situation-driven Execution* [64] and Firby created a plan representation for this approach: Reactive Action Packages (RAPs) [21].

Georgeff and Lansky [23] describe an *executable specification language*, the Procedural Reasoning System (PRS). PRS is a means for representing knowledge about procedures, which contains semantics to specify facts about processes and their behaviors. The underlying mechanism behind PRS is an intention graph of plan actions (KAs) and meta-KAs control execution.

2.6 Plan Execution and Monitoring

Section 2.5 discusses the sources of uncertainty. Uncertainty can cause plans to be less-effective, or completely ineffective, but the effects can sometimes be reduced. Certain types of uncertainty can be eliminated using better hardware or engineering the environment, but this method can increase the cost of planning or lessen the environmental applicability. On the other hand, we can reason about uncertainty during planning. This method results in more complex models and therefore higher planning complexity. The last method for decreasing the effects of uncertainty is by tolerating the uncertainty. In this approach, plan execution monitoring helps prepare for failing execution.

Isermann and Balle [35] define **execution monitoring** as "a continuous real-time task of determining the conditions of a physical system, by recording information, recognizing and indicating anomalies in the behavior." The purpose of monitoring plan execution is to find faults, a **fault** being "an anomaly in the behavior of the monitored system."

Plan execution monitoring has been studied extensively by control theorists as the *fault detection and isolation* (FDI) problem [53]. The functional concepts of monitoring systems include:

- **Fault Detection** — an anomaly has occurred in the behavior of the monitored system;
- **Fault Isolation** — classification of the fault (aka fault diagnosis); and
- **Fault Identification** — determines the magnitude of the fault.

Gertler [27] uses three criteria for evaluating execution monitoring systems: reaction speed, robustness, and isolation performance. *Reaction speed* measures the amount of time between the actual fault and its detection. *Robustness* indicates the monitor's ability to operate "in the presence of noise, disturbances, and modeling errors." *Isolation performance* measures the number of correct/incorrect fault type assignments.

Hammond *et al*. [30] promotes the importance of fault isolation by showing that propigating failure information back to the planner can drastically help in the replanning phase.

Hart *et al*. [31] describes a representation for action progress expectations, which Hart *et al*. label *envelopes*. The term comes from performance envelopes in engineering disciplines which describe performance profiles. The point of these envelopes is to avoid wasting time executing costly actions when it is clear that they will fail prior to their completion. Using envelopes allows an agent to:

- modify a failing plan so as to prevent its failure,
- abandon a failing plan,
- retire surplus resources from a succeeding plan,
- improve a plan going unexpectedly well, and/or
- reduce communication between cooperating agents (sharing expectations means they only have to communicate when the expectations are violated).

Chiang *et al*. [9] lists three classifications of execution monitoring. The following sections explain the analytical, data-driven, and knowledge-based approaches in detail.

2.6.1 Analytical Approach

In this approach, the presence of a fault is derived from the difference between two analytically generated quantities. The difference is called *residual* and the algorithm that processes the measurable inputs and outputs to a system is called *residual generation*.

The analytical approach utilizes some decision making algorithm, $d(r)$ where r is a residual, based on residual generation, $r(s)$ where s is a signal from a system. The process relies on the concept of analytical redundancy and is influenced by the residual generation technique. [50] identifies three methods of residual generation: parameter estimation, parity relations, and observers.

Some examples of papers that use analytical methods for fault detection are the Procedural Reasoning System (PRS) [24] and Reactive Action Package (RAP) [21]. An illustration of the analytical approach is shown in Figure 2.4.

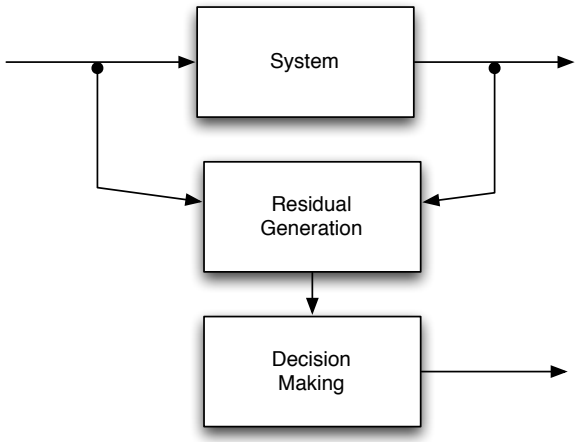

Figure 2.4: The main components of the analytical approach towards fault detection.

Parameter Estimation

In parameter estimation, residuals are calculated as the difference between a running system and a reference model of "normal" system execution. This method requires that a reference model be created and maintained a priori.

Parity Relations

Parity relations are mathematical equations that describe a system based on its parameters. In this method, residuals are derived from consistency checks between a reference model and the currently running parity relations.

Observers

Observer-based methods of residual generation differ from the previously described methods because they do not require system model parameters to be derived *a priori*. The goal of observers is to estimate system outputs during execution of a system. The residual is then the difference between the estimated and the actual system output.

According to [53], most analytical AI research falls into the observer category. One example, the Kalman filter, is used quite often to estimate values from noisy sensor data.

2.6.2 Data-driven Approach

Data-driven approaches do not rely on mathematical models, but instead they are directly derived from sensor data. For example, a fault might be detected if sensor data exceeds some range of deviation from the mean of the previously collected data.

These approaches are classified by the number of variables included in their monitoring. Single-variable approaches are labeled univariate statistical monitoring and all other approaches are labeled multivariate statistical monitoring.

Pettersson has studied data-driven approaches in depth in [53, 52]. These approaches do not require users to construct models of the system. For this reason, they are labeled model-free approaches to execution monitoring. In contrast, model-based methods of execution monitoring run the risk of interpreting modeling errors as faults in system execution. Model-free approaches, however, are only limited by the amount of sensor data used to train the behavior-based monitor.

Although [52] states that most robotics applications use model-based execution monitoring, [7] is one example of a research effort that utilizes model-free execution monitoring. In [7], Bouguerra *et al*. encode semantic information as *description logics* and use these to derive expectations for robot behavior.

2.6.3 Knowledge-based Approach

Knowledge-based approaches to execution monitoring are designed to simulate human problem-solving. They can be model-free, model-based, or some hybrid of other methods. To offset the high cost of human behavior simulation, knowledge-based monitoring systems perform fault isolation in addition to fault detection. [53] lists three categories of knowledge-based approaches: causal analysis, expert systems, neural networks.

Causal Analysis

Causal analysis methods model fault-symptom relationships to perform fault isolation. Of the causal analysis approaches, several papers [36, 75] discuss multi-agent execution monitoring.

The work in [36] focuses on ”overhearing” team members. Overhearing, in this context uses models of social relationships to monitor task execution.

Wilkins *et al*. [75] uses ”Execution Assistants” to perform multi-agent, causal analysis execution monitoring in dynamic domains. Unlike [36], Wilkins *et al*. rely on agent communications to transfer agent state changes. This state information is used to populate agent beliefs in the BDI model.

Expert Systems and Neural Networks

Expert systems and neural networks are used to directly model human/expert behavior in execution monitoring. Expert system approaches are model-based, whereas the strengths of neural networks are exhibited by ”learning” from sensor data, thus making them appropriate for model-free monitoring.

The Saphira architecture [40] is one example of an execution monitoring system implemented using an expert system. [52] explores the possibility of using simulation data to train an artificial neural network.

2.7 Qualitatively Different Plans

There are two high-level techniques of finding qualitatively-different plans: domain-independent, and domain-dependent.

2.7.1 Domain-independent

One advantage to using domain-independent methods of finding qualitatively-different plans is that they require no meta-data about the domain to derive different plans. One method of finding qualitatively-different plans without additional information gathers high-level preferences from the user. This approach is labeled, *mixed-initiative*.

Although TRIPS and TRAINS are considered mixed-initiative planning assistants, the authors explicitly say, "traditional planning technology does not play a major role in the system" [18]. The systems are mixed-initiative in that they help to repair initial tasks, however their approach aims to perform plan repair on an existing plan rather than generate unique plans.

Srivastava *et al.* [63] investigate methods to find inter-related plans. They use a function, DISTANCE$(plan1, plan2)$, to represent the similarity/diversity of two plans. The function could use any combination of the following three domain-independent criteria:

- the actions present in the plan,
- the set of stages (or states) that execution takes, and
- the causal chains that support plan goals.

Tate *et al.* [66] use a mixed-initiative approach, which is largely driven by the task assigner agent. The agent selects assumptions on the top-level activities, and the planner is then responsible for refining the lower-level plan activities.

2.7.2 Domain-dependent

The advantage of using domain-dependent methods of finding qualitatively-different plans is that they incorporate domain information into the inter-related plan measurements. This is accomplished by adding new (domain-dependent) criteria to definition of the DISTANCE function. In [46] and [47], Myers *et al.* use the concept of domain metatheory to evaluate plans. In addition to providing a mechanism for comparing sets of plans, metatheory also provides capabilities to summarize plans. The conceptual components of metatheory are as follows:

- template features (which allow us to differentiate between functionally equivalent alternatives),
- task features (which compose a typing system), and
- roles (which describe the capacity to which an individual resource is used).

Myers uses these components to direct planners towards solutions with distinct semantic traits.

2.8 Measuring Planners

As mentioned in Section 2.4.1, [79] explains the difficulties in evaluating planners. Because there are fundamental differences in the purposes of modern planners, comparing them can be extremely complex. Furthermore, there are multiple dimensions on which planners can be categorized. These factors include:

- correctness,
- quality of solution,
- failure rate,
- resources required to generate plans,

- robustness of planner,
- graceful degradation, and
- user friendliness.

Zilberstein also explains that comparison to human performance is complicated due to complex human behaviors. However, by evaluating the system based on the improvement in execution, we can compare different types of planning. "The overall utility of a planner is determined by the effect it has on the agent behavior."

2.8.1 Goals-Question-Metric

Basili *et al*. [3] present the Goal-Question-Metric (GQM) approach for deriving metrics from project requirements. In GQM, *Goals* specify issues, objects, and viewpoints. Strictly structured *Questions* are used to quantify the degree to which goals are met, and some goals may require several questions to assess. Finally, *Metrics* are data used to answer the questions and assess the state of the goals.

The GQM approach deals with two criteria: performance and effectiveness. Effectiveness attempts to capture how well the system accomplishes its goals, whereas performance deals with the utilization of resources during system execution.

2.9 Network-Centric Systems and Network-Awareness

The definition of *network-centric systems* states that the performance of the system is dependent on the quality of underlying network links. Consider a network file system for example. If the quality of the communication link between the client and server diminishes, the read and write speeds to the network file system will suffer as a result.

Static, homogeneous networks lack variation in quality of network links by definition. Therefore, the concentration is on dynamic networks, such as MANETs, and heterogeneous networks as described by Wu *et al*. [78].

According to [70], there are many aspects of network-centric efficiency that can be measured. In this work, the authors examine Quality of Service (QoS) metrics such as:

- RSSI,
- packet loss,
- latency, and
- throughput.

Furthermore, the authors recognize the importance of end-user experience by measuring Quality of Experience (QoE) metrics such as:

- situational awareness application latency, and
- VoIP Mean Opinion Scores (MOS).

Mahambre *et al*. [42] discusses QOS-aware, adaptive event-dissemination middleware. In doing so, the authors define characteristics of middleware. One such characteristic is *underlay awareness*. According to Mahambre *et al*. middleware exhibits underlay awareness when its

> "construction strategy considers some of the underlying physical network's properties, including path disjointedness (absence of common nodes or physical links in different overlay paths), hop count (number of physical nodes between a pair of overlay nodes), bandwidth, physical distance information, and failure statistics. This makes the overlay sensitive to changes in the underlying physical network — that is, the overlay dynamically adapts to resource-based adaptation triggers" [42].

I make use of this definition when I define the concept of *network-awareness* in Definition 6.

Another contribution of Mahambre *et al*. [42] is splitting underlay awareness into two categories: *underlay-proximity-aware* and *underlay-quality-aware*. Underlay-proximity-aware middleware uses the physical nodes' network proximity, such as a neighbor list provided by a routing protocol. A yet unimplemented category of underlay awareness, underlay-quality-aware, addresses "physical link quality, failure probabilities, node degree, physical network diameter, and QoS guarantees" in the middleware layer.

2.10 Scenario Background

The motivation for this project is a service composition scenario, particularly in Improvised Explosive Device (IED) adaptive change detection. Papers that use planning to accomplish service composition include [60, 62] and visual IED change detection is documented in [41].

2.11 Related Work

I have conducted research in the following areas:

- Mapping service composition to automated planning;
- The classic automated planning problem and its variants;
- The role of agents in automated planning problems;
- Specific constraint models used by I-X and I-Plan;
- The effect of uncertainty on solving automated planning problems;
- Types of execution monitoring in planning systems;
- Qualitatively different plan generation techniques;
- Methods for measuring planning, execution, and monitoring agents;

- Network-centric systems and considering network properties in the application layer.

The motivating scenario described in Section 1.1 is an instance of a *web service composition* problem as described by [33]. Gu *et al*. [29] assure QoS over web service compositions, but their solution does not allow domain-specific evaluation criteria.

Sirin *et al*. [60] and Hoffmann *et al*. [33] provide techniques for mapping the automated composition of semantic web services to an automated planning problem, where domain-specific evaluations have been studied in [46, 47]. The IED detection scenario exhibits certain characteristics that cause uncertainty at plan-time. Thus, I investigate planning under uncertainty and determine ways to cope with uncertainty.

Once the service composition problem is stated as (a variant of) an automated planning problem, plan evaluations are used as heuristics to guide the planning search and choose among sets of plans. Next, I investigate methods of executing plans and monitoring the execution using FDI. Finally, a study of network-centric systems and network-awareness gives insight into methods of incorporating network properties into planning, execution, and monitoring agents.

3. Formalization

This chapter establishes a formal problem statement which builds on the planning notation established in Section 2.1. Next, Section 3.2 states my hypothesis and Section 3.3 gives a detailed explanation of the motivating IED detection scenario.

3.1 Formal Problem Statement

$\mathcal{P}$ the planning problem $\mathcal{P} = \langle \Sigma, s_0, S_G \rangle$

Σ the domain $\Sigma = \langle S, A, E, \gamma \rangle$

S the set of states

s_0 the initial state

S_g the set of goal states

A the set of actions, $A = \{a_0, a_1, \ldots, a_{|A|}\}$, where each $a = (\text{precond}(a), \text{effects}^-(a), \text{effects}^+(a))$;

E the set of system events

γ the state transition function, $\gamma : S \times A \rightarrow S$

P the set of plans, $P = \{p_0, p_1, \ldots, p_{|P|}\}$ where each plan $p \in P$ is a totally ordered sets of actions

H the network hosts $H = \{h_0, h_1, \ldots, h_{|H|}\}$

ω_H the host link weighting, $\omega_H : H \times H \rightarrow (0, 1]$. The value represents the quality of the link, 0 being the lowest and 1 the highest. This link is only conceptual and the actual data route may go between other hosts or routers

precond(a) returns the preconditions of action, a

effects$^+(a)$ returns the positive effects of action, a

effects$^-(a)$ returns the negative effects of action, a

offers(a) returns the precondition(s) stating that a host offers the service represented by action, a

host(a) returns the single host $h \in H$ on which action a can be run, or $\emptyset$ if the action is ungrounded

resources(a) returns the set of resources (parameters) of action a, or $\{\}$ if the action is ungrounded

Most of this notation is derived from the formalization of the classical planning problem

from [49]. My problem formalization contributions are H, ω_H, offers(a), host(a), and resources(a).

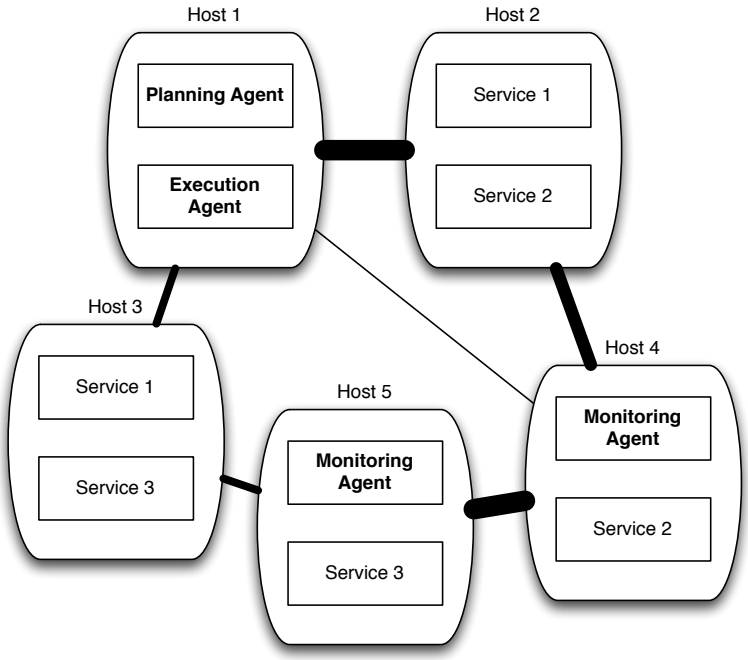

Figure 3.1: The conceptual diagram of the formal problem statement. Rounded rectangles represent hosts $h \in H$, which are capable of executing a set of services. Planning, execution, and monitoring agents reside on hosts. Lines between hosts represent communication links between the hosts and the thickness of the line is a representation of link quality, ω_H.

Figure 3.1 shows the conceptual diagram of the formal problem statement. Hosts, $h \in H$ are connected via a network, with links from host h_i to h_j. Each link also has a relative quality, ω_H, illustrated as the thickness of the connecting lines.

Figure 3.2 shows the role of agents in the formal problem statement. There are three types of agents: planning, execution, and monitoring. Each agent runs on a host and the agents communicate with each other over the network.

Also, Figure 3.2 shows the data flow between agents in the formal problem statement.

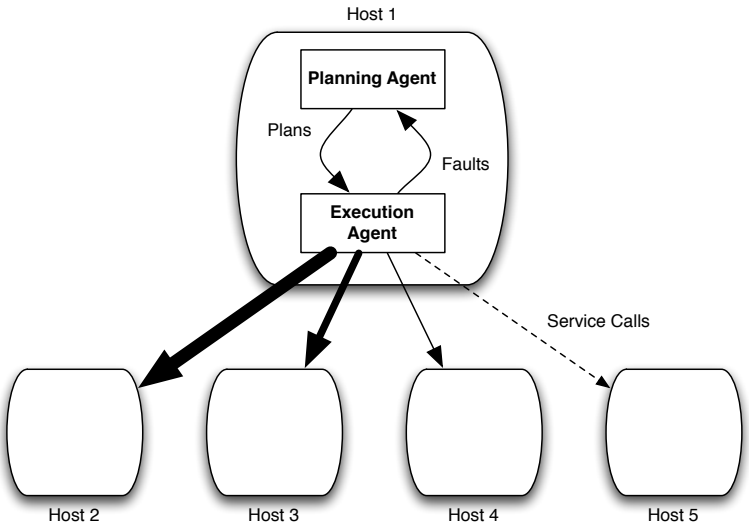

Figure 3.2: The data flow and role of agents in the formal problem statement. There are three types of agents: planning, execution, and monitoring. Each agent runs on a host and the agents communicate with each other over the network. The planning agent is responsible for creating plans and passing them to the execution agent. The execution agent invokes services on the hosts according to the plan. The monitoring agents report execution faults to the planning and execution agent as needed. Each network link has a relative quality, ω_H, illustrated as the thickness of the lines connecting hosts. Service-specific, inter-host communications are not illustrated in this diagram.

The planning agent is given the tuple $I_P = \langle \Sigma, s_0, S_g, H, \omega_H \rangle$, and constructs a plan $p_I \in P$. The execution agent accepts $I_E = \langle p_I, H, \omega_H \rangle$, and invokes the totally ordered service calls described by the actions in p_I. The plans can be augmented with temporal constraints to better describe the sequence of service invocations.

Figure 3.3 shows the sequence diagram for the plan execution process. Plans are created by the planning agent and passed to the execution agent. The execution agent causes changes to a system, which can be sensed by monitoring agents. When a monitoring agent detects a fault, it isolates and identifies the fault (see Section 2.6). Based on the magnitude of the fault and the capabilities of the agents, the monitoring agent reports the fault to the execution agent (for plan repair) or planning agent (for replanning).

The problem is to find and execute $p_I \in P$ where $p_I = \{a_0, a_1, \ldots, a_{|p_I|}\}$ and the

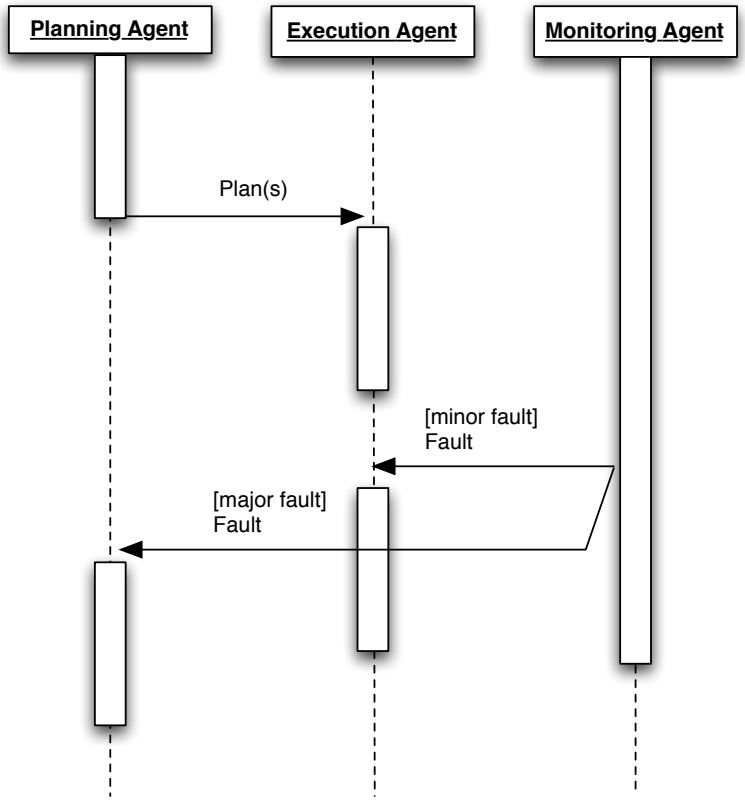

Figure 3.3: The sequence diagram for the plan execution process. The planning agent is responsible for creating plans and passing them to the execution agent. The execution agent invokes services on the hosts according to the plan. The monitoring agents report execution faults to the planning and execution agent as needed.

execution of p_I yields the best domain-dependent and network-centric evaluations.

Definition 6. An agent exhibits **network-awareness** if changes to ω_H cause the agent's output to change while all other inputs remain constant.

3.2 Hypothesis

In intelligent multiagent systems, agents conduct reasoning and coordination within the system. Coordination, however, is difficult when the cost and reliability of communications is not fixed. Peysakhov *et al*. [54] shows that incorporating "network awareness" into the

agent's reasoning improves the performance of an intelligent system.

Therefore, I hypothesize that incorporating network-centric system aspects into plan construction agents, execution agents, and monitoring agents improves system performance and effectiveness. Furthermore, incorporating network-awareness into different stages of the planning process affects performance and effectiveness differently depending on the dynamism of the network. Applications running on mostly-static networks benefit most from network-awareness at plan-time, whereas applications running on highly-dynamic networks receive the greatest benefit from network-awareness at runtime.

3.3 Motivating Scenario

The motivating scenario models service composition on a constrained network as an HTN planning problem where simple tasks represent service calls and complex tasks model service compositions. I adopt the OWL-S process ontology to describe the composition of web services as in [60].[1] Sirin *et al*. describe this behavior as an "action or process metaphor... primitive and complex actions with preconditions and effects." I then map the web service composition problem to the automated planning problem [49] by representing services as actions.

In this scenario, there are several locations that must be monitored for Improvised Explosive Devices (IEDs). Monitoring can be accomplished using services provided on nodes throughout the network — via manual (human) searching or visual change detection. Each combination of IED detection method, resource(s) used, and order of execution yields different plan evaluations.

A static model of communication properties dictates the values of network-centric plan evaluations. Evaluators for *bandwidth usage* and *link quality* represent network constraints. Bandwidth is consumed by plan actions which require non-local data or services. I model

[1] `http://www.w3.org/Submission/OWL-S/`

bandwidth to show preference to local information rather than reach-back interfaces. Link quality evaluations are used to favor low-latency network communications in local area networks. As with other plan evaluators, network-centric evaluators are affected by planning actions and their constraints.

I seek to exploit the tradeoffs between the evaluation criteria to return qualitatively-different plans to the task assignment agent. The scenario is illustrated in Figure 3.4. Red rectangles represent physical locations that require IED monitoring — an IED has been planted in location 2. Humans, robots, and UAVs are the entities that are capable of performing IED detection techniques. Cameras represent the geographical positioning of resources required to perform visual change detection. The goal is to minimize the overall plan cost where that cost can be affected by not only plan actions, but also their ordering constraints and the resources they utilize.

The following are the planning actions for the IED detection domain:

- SWEEPFORIEDS: defines the list of locations to be searched.
- CHECKFORIEDAT: satisfied by manualSearch or photographicSearch.
- MANUALSEARCH: complex task for a human search of a location.
- PHOTOGRAPHICSEARCH: complex task for a change detection search of a location.
- CONDUCTSCAN: a human scans a location for an IED.
- ACQUIRECAMERA: a resource acquires a camera (a requirement for a change detection task).
- TAKEPHOTO: take a photo of a location using a camera.
- GETOLDPHOTO: get the last photo taken of a location (a requirement for comparing photos).
- COMPAREPHOTOS: compares two or more photos for a change that would indicate an IED is present at a location.

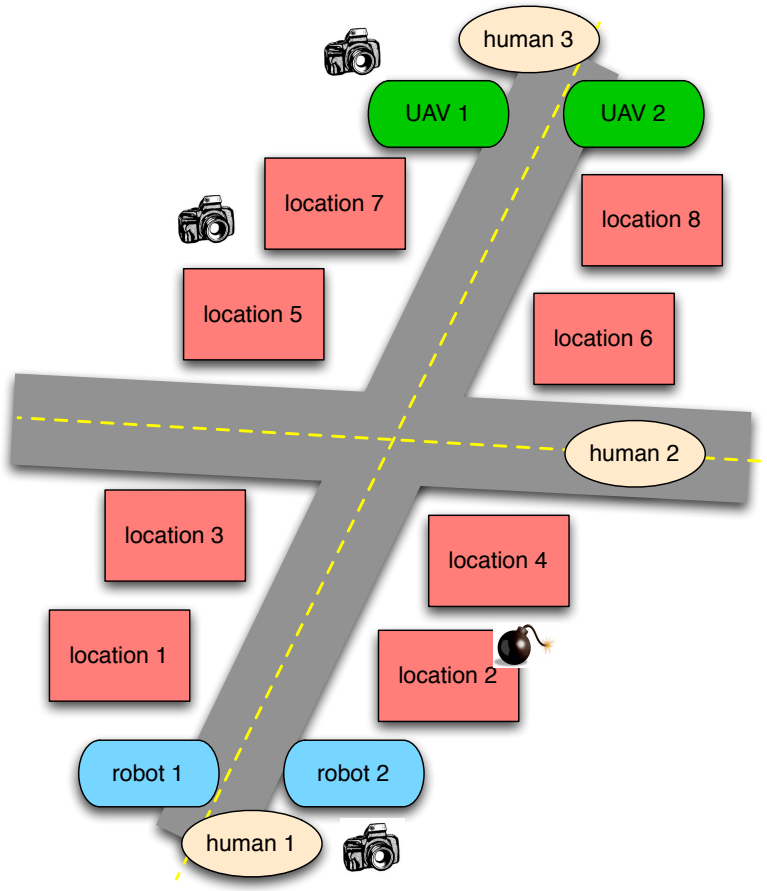

Figure 3.4: Conceptual diagram of the IED detection scenario. Several locations are monitored for IEDs using manual searching or visual change detection.

- PHYSICALMOVE: move a resource from one location to another.
- REPORTRESULTS: the results of a scan are reported to a central authority.

Also, there are high-level resources with types: net-nodes, locations, and cameras. Net-nodes here represent anything that can move in physical space or communicate over the network. Thus, both humans and UAVs are represented by net-nodes. To distinguish between these types of net-nodes, I use net-node properties. For example, a UAV net-node has an average speed around 370 kph versus a human net-node which is about 95 kph (traveling in a car). Locations are geographical places where net-nodes can move, and cameras

are resources required for creation of photographs.

4. Technical Approach

As discussed in the background section, there are three types of agents involved in the plan execution process: planning agents, execution agents, and monitoring agents. My approach to adding awareness to network-centric systems is to reason over network properties in each agent.

Each agent benefits differently from knowledge of different network properties. Network-aware planning agents can use network-based plan evaluations to guide a heuristic plan-space search. Execution agents exhibit network-awareness in plan repair and resource grounding. Network-aware monitoring agents monitor network properties to find failures in plan execution.

This chapter formalizes the purpose of each agent and discusses methods of adding network-awareness to each agent. Also, this chapter explains the implementations of agents that are used in the experiments.

4.1 Planning Agents

The planning agent is responsible for producing a plan. The goal of the plan is to give advice to the execution agent. The main implementation differences between the planning agents stem from the technique used to guide the planner's search strategy.

Following the notation introduced in Section 3.1, the purpose of a planning agent is to accept the tuple I_P as input and produce a plan in the form of an ordered service composition p_I.

4.1.1 Domain-independent Planning Agent

The domain-independent planning agent is based on I-Plan's default search strategy, which uses a combination of exploration and optimization to return different plans.[1] As the planner traverses the search space, it switches between a depth-first exploration strategy and an A* optimization strategy using the number of activities in the partial plan as its admissible heuristic. The planner starts by traversing the space in a depth-first manner and when it encounters an alternative whose constraints cannot be satisfied, it backtracks using the A* search.

The domain-independent planning agent utilizes domain-independent heuristics to guide the planner's search strategy. Included in these heuristics are the following:

- the number of steps the planner took to generate a plan;
- the number of alternatives the planner uncovered along its way;
- the number of options below the revealed alternatives;
- the number of alternatives left unexplored;
- the longest path along temporal ordering constraints; and
- the number of duplicate plans found before returning a new plan.

4.1.2 Random Planning Agent

The random planning agent conducts a depth-first search, selecting randomly among branch-point alternatives in the plan-space search. Algorithm 1 shows the process of the random planning agent.

[1] All planning agents presented in this thesis were implemented using Tate *et al.*'s plan-space HTN planner, I-Plan [11].

Algorithm 1 CONSTRUCTRANDOMPLAN($\mathcal{P}$)

Require: $\mathcal{P}$ is the planning problem for which this algorithm constructs a plan to solve.
Ensure: s_0 is the initial state of $\mathcal{P}$, toVisit is a deque of branch-point alternatives in the plan-space, visited is a list of alternatives already traversed, randomize(z) is a function that returns the ordered set z with all its original elements in a random order, and solution(x) is a boolean function that returns true if x meets solution criteria set by the planner.

```
1:  toVisit.push(s_0)
2:  while ¬ toVisit.empty() ∧ ¬ solution(toVisit.peek()) do
3:      v ← toVisit.pop()
4:      if v ∉ visited then
5:          visited.add(v)
6:          r ← randomize(v.children())
7:          toVisit.push(r)
8:      end if
9:  end while
10: return toVisit.peek()
```

4.1.3 Plan Evaluation Guided Planning Agent

The guided planning agent defines domain-dependent plan evaluators for criteria that are important to the end-user. In my case, I create evaluators for network performance as well as domain-dependent criteria. The guided algorithm seeks to exploit the trade-offs between the defined evaluation criteria by finding qualitatively-different plans.

The network-aware algorithm for the guided planning agent is described in detail in Section 4.4.1.

4.1.4 Measuring Planners

Section 2.8 gives some background on measuring planners and the GQM approach [3]. Although it is difficult to compare planners that attempt to accomplish different goals, there are some criteria on which planners can be measured. The following section classifies these effectiveness and performance measures by the time (in the planning process) when they are applicable.

This section describes characteristics of plans prior to their execution. It is further divided into qualities that indicate the planning agent's effectiveness and performance.

Planner Effectiveness

completeness	Does the plan satisfy the goal barring unforeseen changes in the environment?
cost/utility	How close, in terms of a cost/utility function, is a plan relative to other plans?
robustness	Can the plan produced adapt to changing environmental conditions?
size of plan	How large, in bytes, is the final plan?
length of plan	How many actions are in the plan?
domain-dependent metrics	e.g. how many blocks (in blocks-world) are moved to complete the plan?

Planner Performance

time	How long, in CPU cycles (or seconds), did it take to generate the plan?
memory consumed	How much memory in bytes was used to generate the plan?

Discussion

The effectiveness of the planning agent is important during plan execution. A more effective planning agent gives better advice to the execution agent. A planner's performance is important in circumstances when the planning task is on-line. During on-line planning, the world state can change while planning occurs, meaning a plan that was valid when planning began might be invalid at the end of the planning task. Planning performance becomes even more important when the planning agent is running on a resource-constrained device.

4.2 Execution Agents

The execution agent is responsible for executing a set of actions on a system. Different types of execution agents are defined formally in this section.

Following the notation introduced in Section 3.1, the purpose of an execution agent is to accept a plan I_E as input from a planning agent and invoke its services, respecting ordering constraints. The policies for service invocation and error handling are defined by the type of execution agent.

4.2.1 Naïve Execution Agent

The naïve execution agent receives a plan as input and executes its actions blindly. It ignores errors that occur as a result of communication problems and failed action execution.

Table 4.1 shows the summary of the naïve agent's execution policies.

Policy	Description
Service Invocation	Invokes services exactly as described by p_I. The naïve agent requires that $\forall$ actions $a \in p_I$, host$(a) \neq \emptyset \wedge$resources$(a) \neq \{\}$.
Error Handling	Ignores execution errors.

Table 4.1: Description of the Naïve Execution Agent policies using the formalization described in Section 3.1.

4.2.2 Reactive Execution Agent

The reactive execution agent, like the naïve agent, receives a fully-ground plan as input. The reactive agent, however, can react to action execution failures. Furthermore, if the sensing action detects a fault in plan execution, the reactive agent has the opportunity to isolate and correct the fault.

The method of reaction is based on the type of monitoring. A reactive execution agent working with a synchronous monitor (i.e., analytical) adds its own sensing actions after each action in the original plan. Thus, the reactive agent has some notion of plan execution progress, and if a fault is detected, the agent knows which action(s) caused the fault.

A reactive execution agent working in conjunction with an asynchronous monitor (i.e., data-driven) receives fault information asynchronously. Therefore, the action(s) which caused the fault is uncertain to the execution agent. For this reason, passing fault isolation information to the execution agent is extremely useful for reacting to failures.

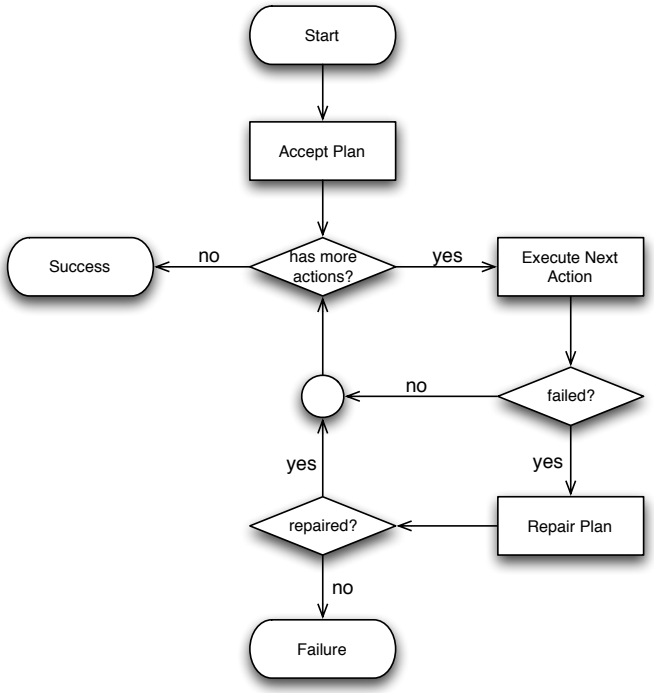

Figure 4.1: Flow chart of the reactive execution agent.

Table 4.2 shows the summary of the reactive agent's execution policies and Figure 4.1 shows the flow chart of the reactive agent's processing.

Policy	Description
Service Invocation	Invokes services exactly as described by p_I. The reactive agent requires that $\forall$ actions $a \in p_I$, host$(a) \neq \emptyset \wedge$resources$(a) \neq \{\}$.
Error Handling	Repairs the failed p_I by replacing failed service call(s) with new ones, creating p'_I.

Table 4.2: Description of the Reactive Execution Agent policies using the formalization described in Section 3.1.

4.2.3 Proactive Execution Agent

The proactive execution agent differs from the naïve and reactive agents in that it does not require fully-ground plans as input. Instead, the proactive agent accepts a list of actions and it conducts reasoning to determine how best to allocate resources at execution time.

This method requires less-intensive planning, but incurs higher cost at execution time because it conducts resource allocation logic. The proactive agent reacts to failures in plan execution in the same way as the reactive execution agent, although fewer failures are expected to occur due to proactive sensing prior to action execution.

Table 4.3 shows the summary of the proactive agent's execution policies and Figure 4.2 shows the flow chart of the proactive agent's processing.

Policy	Description
Service Invocation	Invokes services using network-aware logic to choose the host and resources at execution time. The proactive execution agent uses only service descriptions from actions $a \in p_I$, meaning $\forall a \in p_I$, host$(a) = \emptyset \wedge$ resources$(a) = \{\}$
Error Handling	Repairs the failed p_I by replacing failed service call(s) with new ones, creating p'_I.

Table 4.3: Description of the Proactive Execution Agent policies using the formalization described in Section 3.1.

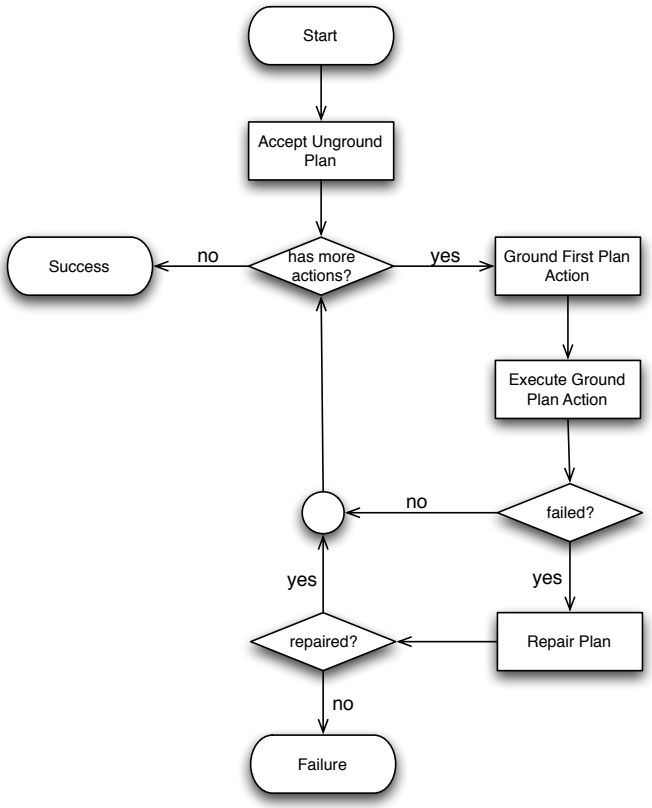

Figure 4.2: Flow chart of the proactive execution agent.

4.2.4 Measuring Execution Agents

This section describes characteristics of execution agents. As in Section 4.1.4, it is divided into qualities that indicate the agent's effectiveness and performance.

Plan Execution Effectiveness

goal achievement	Does the plan achieve the intended goal?
robustness	Does the plan achieve the intended goal under changing environmental conditions?
reactiveness	Does the execution agent monitor/react-to dynamic environments? Does it detect action execution failure? Does it predict failures? Can it modify execution to avoid failures? Can it modify execution to recover from failures without replanning?

Plan Execution Performance

time	How long (in milliseconds) does it take to execute the plan, including the time required to execute individual actions?
memory consumed	How much memory (in bytes) does the execution agent use to execute the plan?

4.3 Monitoring Agents

The monitoring agent is responsible for monitoring the progress and status of plan execution. The monitoring agent performs fault detection and isolation, reporting its results to the execution and/or planning agent(s). In this section several types of network-based monitoring agents are described.

4.3.1 Analytical Monitoring Agent

The analytical monitoring agent is a synchronous monitor based on the model-free observer method. It is synchronous in that the monitor is polled by an external agent. In this method, an action is executed after observing sensor values. When the action completes, another sensor reading is performed and the observed residual is compared to an estimate.

Analytical monitors are particularly useful to systems where actions have predictable effects on a system. For example, after a crane's *grab* action, we expect the crane to be holding an item. If the crane's arm is empty after executing a *grab*, then a fault occurred.

In the case of network-centric monitors, we expect actions to have certain effects on the system's network. An observer monitors the number of packets and bytes transmitted in accordance with each action. A residual is calculated from an estimate based on the action and its resources. Finally, fault-detection logic determines if the residual falls within the expected range.

4.3.2 Data-driven Monitoring Agent

The data-driven monitoring agent is an asynchronous agent that conducts multivariate statistical analysis to ensure that sensor data falls within the limits of a given model. It is asynchronous in that the monitoring agent runs autonomously and informs external agents of faults when they are detected. For this reason, data-driven monitors are useful for systems that operate according to a predictive model.

In my case, I use network factors that should remain relatively static under normal operation such as PER, the number of retransmitted packets, socket timeouts, connection resets, failed connections/datagrams, etc.

4.3.3 Knowledge-based Monitoring Agent

Knowledge-based monitoring agents use advanced reasoning techniques to observe faults in the system. Expert systems and neural networks are two commonly used reasoning techniques in this approach.

In the case of distributed systems, the physical location of the nodes as well as the physical layer and data link layer communication systems have major impact on the quality of the network. Therefore, geographical location and signal strength can serve as knowledge-

based fault indicators.

4.3.4 Measuring Monitoring Agents

For planning systems that interleave planning and execution, a combination of the criteria from Section 4.1.4 and Section 4.2.4 are applicable.

Additionally, there are two types of monitors: passive and active. Passive monitors have little or no effect on the system — the system operates the same with the monitor as it would without the monitor. Active monitors, on the other hand, have some impact on the system. An active monitor might cause the system to act differently than it would if the monitor were not running. Therefore, another metric for monitoring agents considers the effect(s) and impacts of the monitor on the running system.

Another metric for monitoring agents is their accuracy in fault detection. Accuracy for FDI is measured by the number of false-positives (type I error) and false-negatives (type II error). False-positives occur when a monitoring agent signals a fault that did not happen. A false-negative is a fault that goes undetected by monitoring agents. A perfect monitoring agent has no false-positives or false-negatives.

4.4 Network-Aware Agents

This section describes the network properties and reasoning that occurs in the network-aware agents, including the planning, execution, and monitoring agents. First, I define the notion of network-awareness in terms of the formalization introduced in Section 3.1. Then, I compare the differences between network-awareness in each agent type.

In Definition 6, network-awareness is said to occur when changes to ω_H cause an agent's output to change while all other inputs remain constant. In other words, if the evaluations of the problem solutions do not vary over any different networking conditions, then the agent reasoning is not network-aware.

The difference among the network reasoning techniques in each agent type is the time in the plan execution process when the reasoning occurs. Figure 3.3 shows a sequence diagram illustrating when network reasoning occurs in each agent.

4.4.1 Network-Aware Planning Agents

The guided planning agent is the only planner in the experiments that is network-aware. It utilizes network-based plan evaluations and generates qualitatively-different plans over those criteria.

Current work on generating qualitatively-different plans discusses two high-level techniques: domain-independent, and domain-dependent. I use plan evaluation criteria to represent both techniques. The reasoning for using a single mechanism for both techniques is that plan evaluators are sufficiently capable of recognizing domain-independent as well as domain-dependent information about a plan. This topic is discussed further in the section labeled "Plan Evaluation Criteria Statistics."

My approach differs from previous approaches to generating qualitatively-different plans in that it improves mixed-initiative planning by using a combination of domain-dependent and domain-independent plan evaluations. Furthermore, I generate plans with qualitative differences based on the tradeoffs in the multi-objective evaluation criteria.

The idea for biasing the planner is to identify qualitatively-different plans using plan evaluation criteria. A plan evaluator contains the following:

- a complete/fully-ground plan evaluation function, EVALPLAN($Plan$);
- a potentially non-ground partial plan evaluation function, EVALPARTIALPLAN($PartialPlan$); and
- evaluation criterion statistics.

My method for biasing the planner's search strategy based on plan evaluations is to maintain a set of priority queues, $\mathcal{L} = \{Q_0, Q_1, \ldots, Q_{|\mathcal{L}|}\}$ — one for each plan evaluator.

Every time a new partial-plan/backtrack-point is generated, its viability is assessed and it is inserted into each priority queue according to the partial plan evaluation of the priority queue's plan evaluator. Psuedocode is found in Algorithm 2.

Algorithm 2 HANDLEPARTIALPLAN(p)

Require: p is the partial plan accepted as input. $\mathcal{E}$ is the set of plan evaluators. $\mathcal{L}$ is a list of priority queues containing plan evaluations.
Ensure: EVALPARTIALPLAN$(evaluator, p)$ is a function that evaluates partial plan, p, using the partial plan evaluator, $evaluator$.
INSERT(Q, e, p) is a function that inserts a partial plan, p, into the priority queue, Q, according to the evaluation, e.

```
1: for all evaluator ∈ E do
2:     Q ← L[evaluator]
3:     e ← EVALPARTIALPLAN(evaluator, p)
4:     INSERT(Q, e, p)
5: end for
```

This method does not require a user to define "tiers" of the plan evaluations (e.g. expensive, cheap, fast, slow) as is necessary in domain metatheory. It does, however, assume that there exists a plan evaluator for all plan criteria that are important to an end-user.

The planner uses the greedy strategy discussed in [63] to generate multiple plans. The planner iterates over the plan evaluators for each plan it generates. Thus, every new plan represents an alternative from the head of a different evaluator queue.

Myers's concept of domain metatheory [46] is rooted in discrete evaluation ranges. In Myers's example, transportation methods are distinguished by affordability and time-efficiency (among others). The feature, affordability takes the values "extravagant, expensive, moderate, inexpensive, or cheap" defined as categories of features. Time-efficiency is similarly broken into discrete categories.

By eliminating the predefined categorizations of the evaluations, this allows for a higher level of granularity in evaluating partial plans. Using partial plan evaluators as a basis for guiding the planner's search strategy has a greater expressivity than domain metatheory. Domain metatheoretic values can be expressed as partial plan evaluators, but complex interactions of actions and resources are more accurately modeled as partial plan evaluators.

For example, filling a ground vehicle with expensive high-octane fuel results in wasted money. Where domain metatheory might explore this option because the plan offers significantly different features, plan evaluation guides the planner away from this result.

4.4.2 Network-Aware Execution Agents

Where network-aware planners reason about network properties at plan-time, the network-aware execution agents conduct reasoning at execution-time. The reactive and proactive execution agents exhibit network-awareness. They do so by reasoning about the current network topology at different points in their execution process. The main difference between the reactive and proactive execution agents is that the reactive only utilizes network reasoning when a failure occurs during plan execution whereas the proactive execution agent checks the current state of the network prior to every action execution. These algorithms are contrasted to the naïve execution agent, which does not exhibit network-awareness.

The "Repair Plan" and "Ground First Plan Action" logic from the reactive and proactive execution agents (see Figure 4.1 and Figure 4.2) are the network-aware portions of execution. In my implementation, these execution blocks are identical in that they both accept an action and decide how to best accomplish it given the current network state. The difference is that the proactive agent is given plans where $\forall a \in I_E \text{host}(a) = \emptyset \wedge \text{resources}(a) = \{\}$.

The logic for the network-awareness attempts to satisfy the action's causal link to the goal adding as few actions as possible. Furthermore, the network-awareness logic in the reactive and proactive agents gives preference to plans using hosts with the greatest ω_H.

My implementation of the execution agents maintains a model of the planning domain in memory, updating the world state as plan execution occurs. When a network-aware execution block is reached, the execution agent consults the routing protocol to determine ω_H and chooses from a list of repair methods.

4.4.3 Network-Aware Monitoring Agents

Adding network-awareness to monitoring agents could have two very different meanings. One interpretation monitors plan execution using domain-specific criteria with as little network impact as possible. My interpretation monitors network properties to find failures in plan execution. I do, however, investigate each monitoring agent's network impact in its performance analysis.

Figure 4.3 shows the interaction between the execution agent and analytic monitoring agents. The analytic execution agent injects monitoring actions into the plan between each action. Given the ordered plan $I_E = \{a_0, a_1, \ldots, a_{|I_E|}\}$, it constructs $I_M = \{m_0, m_1, \ldots, m_{|I_E|+1}\}$, an ordered set of monitoring actions. Next, it creates the new execution plan I'_E with Equation 4.1. The result is $I'_E = \{m_0, a_0, m_1, a_1, \ldots, m_{|I_E|}, a_{|I_E|}, m_{|I_E|+1}\}$.

$$I'_E = \bigcup_{i=0}^{n} \{m_i, a_i\} \tag{4.1}$$

A monitoring action probes each of the monitoring agents to trigger residual calculation on the number of packets transmitted. Fault detection logic is conducted in the execution agent based on the residuals from each monitoring agent. The number of transmitted packets was chosen as the network property for the analytic agent because a model (of ideal execution) could easily be constructed for each plan action.

The data-driven monitor has a much simpler implementation. Each monitor performs autonomous fault detection and only reports faults to the execution agent if they occur. Figure 4.4 illustrates the data-driven monitoring approach. The network properties used by the multivariate data-driven monitoring agents were the number of packets sent and the number of retransmission timeouts. These properties were chosen because initial modeling showed that disconnections caused retransmission timeouts to increase while the number of packets sent remained constant. A graph from the initial modeling effort appears in Figure 4.5.

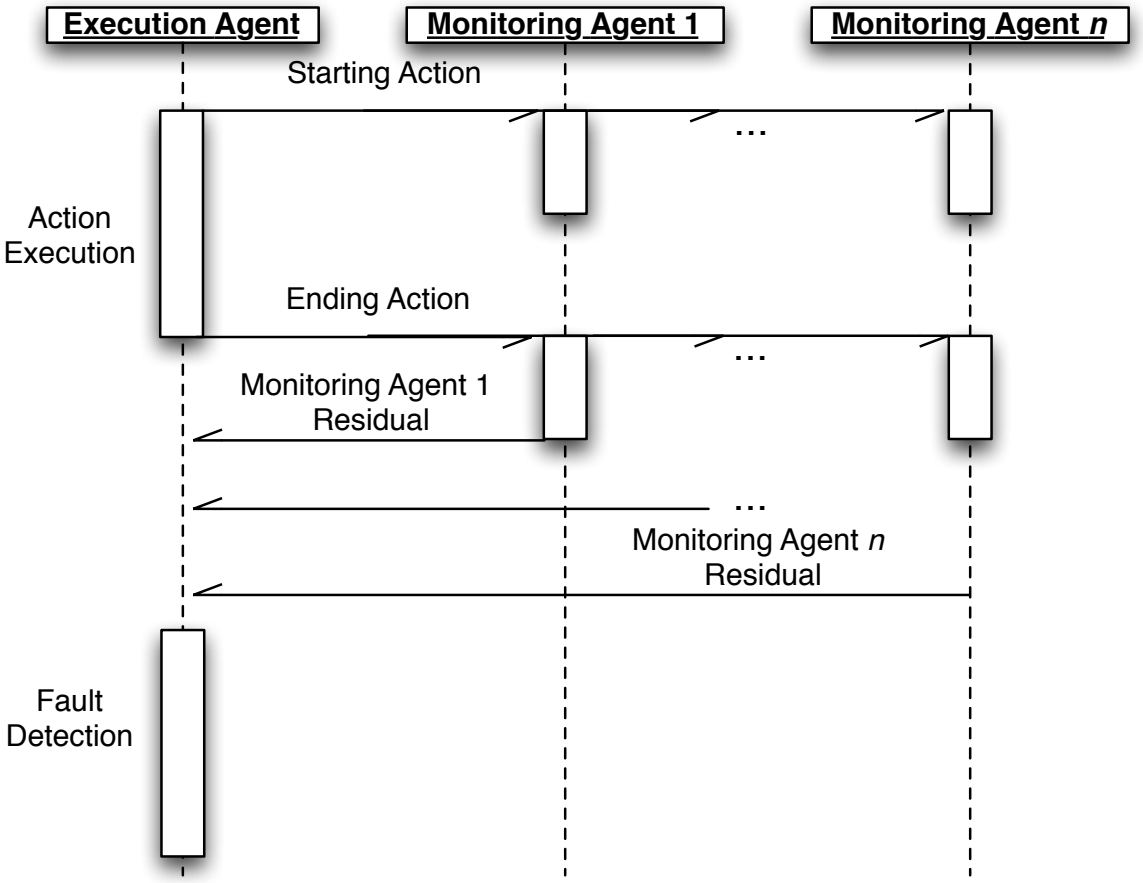

Figure 4.3: Sequence diagram showing the interaction between the execution agent and analytic monitoring agent(s).

4.5 Network-Centric Extensions to the Planning Problem

The challenge in creating network-centric planning extensions is finding a reusable and effective method for incorporating communication properties into the scenario model and planning problem. It is important that this is done properly because a poor representation can cause the planner's search space to increase dramatically. There are two basic ways to model service distribution: The first method, labeled *operator distribution*, represents each service on a node with a different planning action. With this method, actions take the form NODE1ACTION(parameter). The operator distribution process can be implemented using

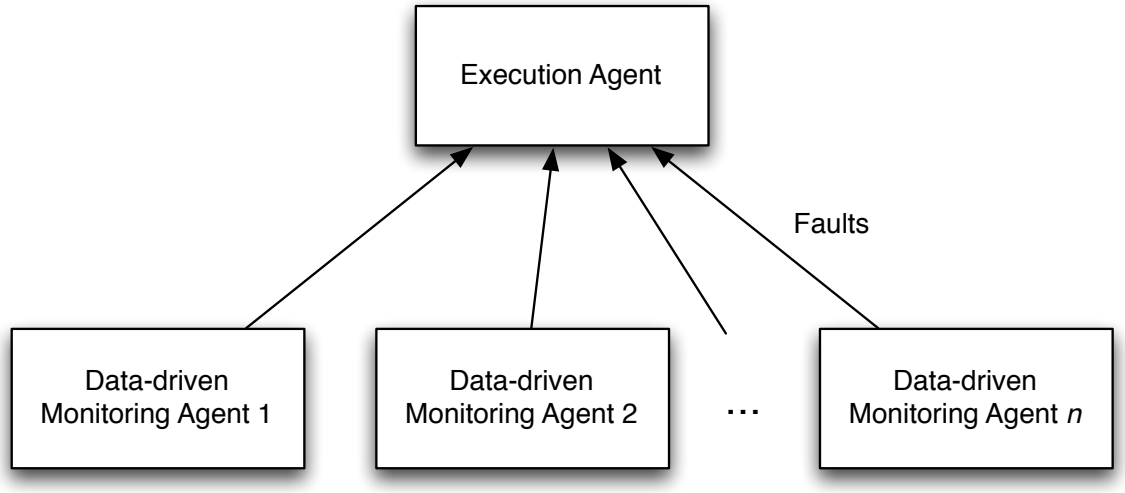

Figure 4.4: Flow diagram showing the interaction between the execution agent and data-driven monitoring agent(s).

the procedure in Algorithm 3.

Algorithm 3 Operator Distribution

Require: A' is the set of all plan actions with network-centric extensions. $\mathcal{N}$ is the set of network-nodes. a_n is the service represented by action, a, executing on network-node, n.
Ensure: NODEOFFERSSERVICE(n, a) is a function that returns TRUE if node, n, offers the service represented by a.

```
1: A' ← {∅}
2: for all a ∈ A do
3:     for all n ∈ 𝒩 do
4:         if NODEOFFERSSERVICE(n, a) then
5:             A' ← A' ∪ {a_n}
6:         end if
7:     end for
8: end for
```

The second, labeled *resource distribution* uses a parameter of the planning action to represent the node on which the service executes. With this method, actions take the form ACTION(node, parameter).

Resource distribution involves an extension to the planning problem for the planner to reason over distributed environments: $\mathcal{P}(\Sigma, s_0, S_g) \rightarrow \mathcal{P}'(\Sigma', s_0', S_g)$ following the notation

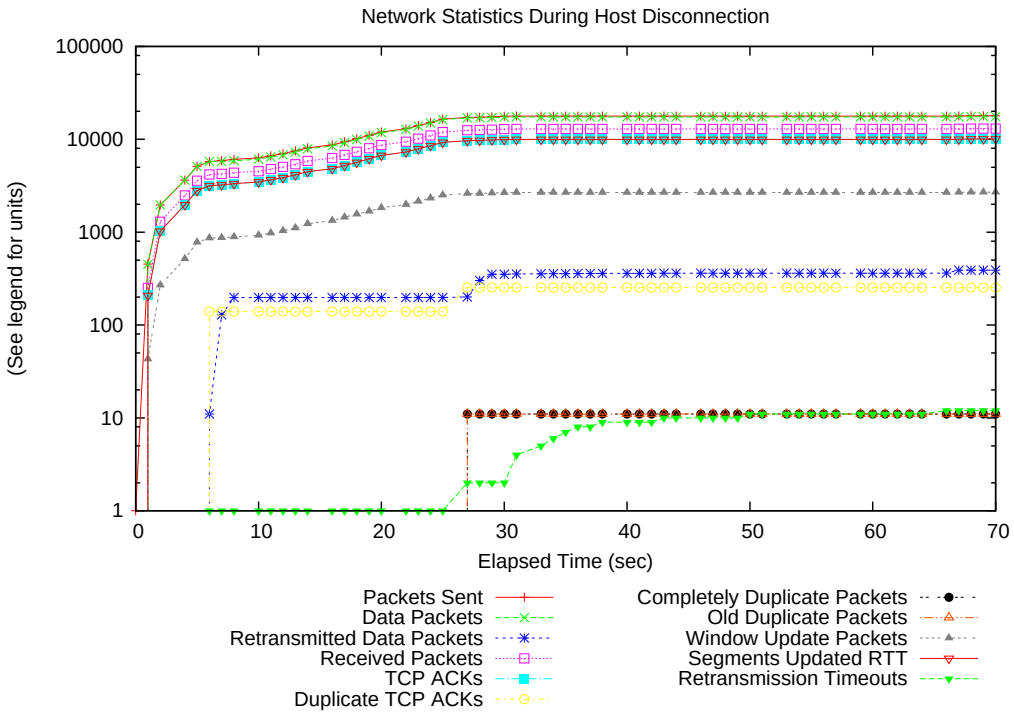

Figure 4.5: Graph illustrating the network criteria tested during the disconnection modeling process for the data-driven monitoring agent. A disconnection occurred roughly at the point on the graph where retransmission timeouts increases sharply.

introduced in Section 2.1.

The actions are modified to include service mapping constraints:

$$A' \leftarrow \{\forall a \in A, a' = ((\text{precond}(a) \cup \text{offers}(a)), \text{effects}^{-}(a), \text{effects}^{+}(a))\}$$

These modifications depend on the world-state to contain constraints for network-node types and the services running on each network-node. The network-node type additions to the world-state, $s_0 \rightarrow s_0'$, are made as shown in Algorithm 4. Further additions to the world-

state are made to map services to network-nodes. The process is shown in Algorithm 5.

Algorithm 4 Resource Distribution — network-node type extensions

Require: s_0 is the initial state of $\mathcal{P}$. s'_0 is s_0 with network-node type extensions. $\mathcal{N}$ is the set of network nodes.
Ensure: NETWORKNODE is the type defined for network node variables in Σ. TYPE is the typing function for plan variables.

$s'_0 \leftarrow s_0$
for all $n \in \mathcal{N}$ **do**
  $s'_0 \leftarrow s'_0 \cup \{\text{TYPE}(n) = \text{NETWORKNODE}\}$
end for

Algorithm 5 Resource Distribution — network-node service mappings

Require: s_0 is the initial state of $\mathcal{P}$. s'_0 is s_0 with network-node service mappings. $\mathcal{S}$ is the set of services on the network.
Ensure: NETWORKNODESOFFERING(s) is a function that returns the set of network nodes that offer service, s.

$s'_0 \leftarrow s_0$
for all $s \in \mathcal{S}$ **do**
  for all $n \in$ NETWORKNODESOFFERING(s) **do**
    $s'_0 \leftarrow s'_0 \cup \{s(n) = \text{TRUE}\}$
  end for
end for

Complexity

Both methods increase the size of the planning search space. Operator distribution adds to the number of actions that can be inserted into a plan. If there are n nodes and each can execute m unique services, then operator distribution increases the number of actions from n in the original domain to $n \times m$ in the worst-case of the distributed service domain. In operator distribution, service availability changes require modifications to the planning domain, whereas resource distribution only modifies the world-state.

Resource distribution keeps the number of actions constant between the original domain and distributed service domain, but increases the difficulty of unification when the planner is searching for applicable actions. Furthermore, it often postpones the grounding

of planning variables until later in the search, which makes it more difficult to calculate role-based heuristics when guiding the planner through the search-space.

4.5.1 Plan Evaluation Criteria Statistics

Alone, plan evaluators as defined here, can distinguish only relative distances between plans. By adding a concept of plan evaluation criteria statistics to plan evaluators, plans can be positioned along an absolute continuum of evaluation values. The aspects of plan evaluation criteria statistics are

- range (effective and theoretic),
- direction (minimize or maximize), and
- statistics (e.g. mean, median, mode, standard deviation).

Plan evaluation statistics specify a theoretic range and monitor the effective range of plan evaluation values. By specifying and tracking the evaluation statistics in this manner, domain metatheory can be implemented within plan evaluators using the theoretic values. For instance, if the theoretic cost of traveling from Philadelphia, PA, USA to Edinburgh, Scotland is [$1:$9999], then three ranges could be evenly divided as:

- Cheap [$1:$3333],
- Moderate [$3334:$6666], and
- Expensive [$6667:$9999].

Another possibility is to dynamically create metatheoretic categories based on the plan evaluations' effective statistics. Following the Philadelphia to Edinburgh travel example, the planner finds flights for $7000, $8000, and $9000, which it labels respectively cheap, moderate, and expensive. When it finds a $5000 flight later, it reassesses the ranges. With this method, the risk is misrepresenting the full range of possibilities, but the categorizations are more realistic.

As discussed in Section 4.4.1, domain metatheory features are rooted in discrete evaluation values. It is implied that the final task assigning agent understand what certain qualities of evaluation criteria are desired. For example, the task assigner must know what level of affordability the user seeks.

The notion of plan evaluators requires that the user specify one evaluator for each concern in the plan. An analogous plan evaluator for the "affordability" feature would be a "monetary cost" plan evaluator which seeks to minimize the overall cost. By specifying the aim to minimize the monetary cost of a plan, we are able to eliminate strictly dominated plans, discussed in Section 2.4.3 and 4.5.2.

4.5.2 Plan Evaluation Visualization

While a task assigning agent might be interested in any number of plan evaluators, the plans whose evaluations dominate other plans in every criteria should be considered. The plan evaluation visualization user interface (viewed by the task assigner) makes a distinction between dominant plans and their dominated counterparts. Dominant plans are defined by Algorithm 6.

Algorithm 6 TEST-PLAN-FOR-DOMINANCE(p, Γ)

Require: Γ is the set of plan evaluation criteria on which to test plan, p, for dominance (modified to seek minimization if necessary). Ψ is the set of all plans (other than the plan being tested, p).
Ensure: Applying a plan to a plan evaluator yields a quantitative evaluation.

```
for all plan ∈ Ψ do
    for all e ∈ Γ do
        return (e(plan) ≥ e(p))
    end for
end for
```

The plan evaluation user interface offers visualization of current evaluation values and each plan evaluator's statistics. The visualization's purpose is to help the task assigning agent to quickly and efficiently understand the options explored by the automated planner

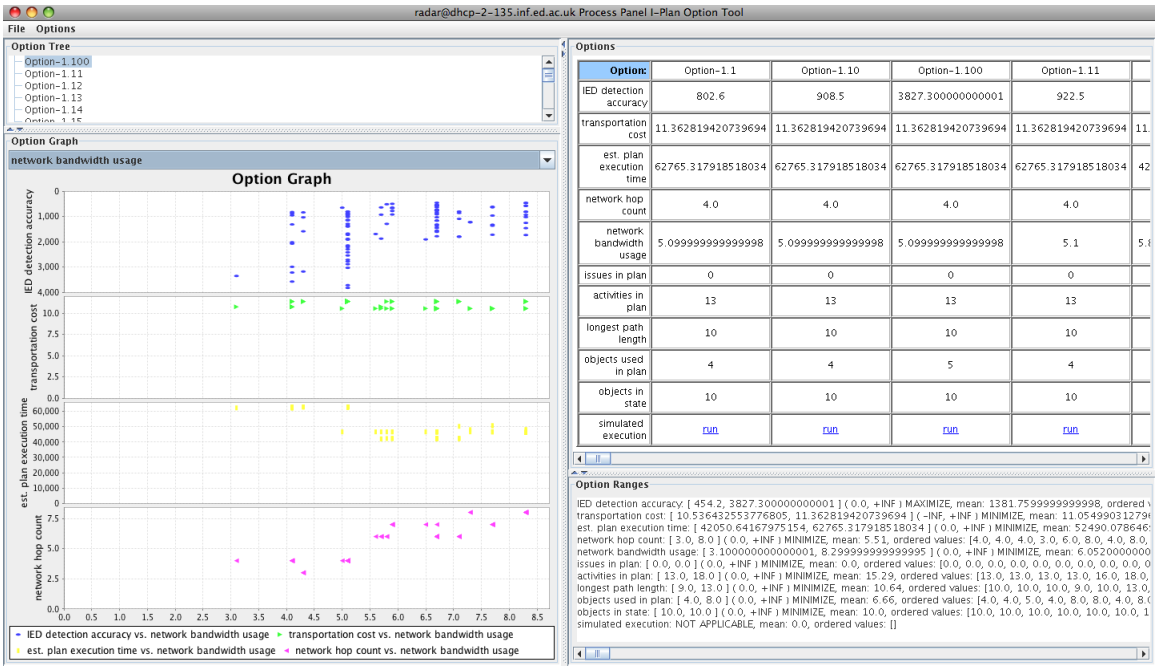

Figure 4.6: A screen capture of the modified I-Plan Option Tool displaying plan evaluation comparisons and statistics. The top-right panel is the *option comparison matrix* which displays textual information about each of the plan options. The bottom-right panel is the *option statistics visualization* which shows the theoretic ranges and direction of the plan evaluators as well as current statistics for each plan evaluator. The bottom-left panel is the *option comparison graph* which shows the effective ranges of the plan evaluators and plots the dominant and dominated plans along these ranges.

in respect to their evaluations. Figure 4.6 shows a screen capture of the tool used to display plan evaluation visualizations.

5. Experiments

There were two groups of experiments created for empirical evaluations. One group determines if my method of incorporating network-awareness into an automated planner yields better results according to plan evaluation criteria. The other quantitatively compares each combination of agent implementation to determine the effects of network-awareness on simulated plan execution.

5.1 Plan Evaluation Benchmarking

The purpose of this experiment is to determine if my method of incorporating network-awareness into an automated planner yields better plan evaluation criteria in a network-centric environment. Figure 5.1 shows the network-centric environment used in the experiment.

Some services are offered by every node in the experiment, these include:

- PHYSICALMOVE;
- ACQUIRECAMERA;
- TAKEPHOTO;
- GETOLDPHOTO; and
- RELEASECAMERA.

Other services are only offered by a few nodes, the remaining services and the nodes that offer are represented in Table 5.1. The resources in the plan evaluation benchmarking experiment exhibit the properties in Table 5.2 and Table 5.3.

See Figures 5.2, 5.3, 5.4, and 5.5 for frequency distributions of plan evaluations of the first forty plans generated by each search strategy: Guided, I-Plan's default, and Random.

Action	Providing Hosts
CHECKFORIEDAT	1, 2, and 5
MANUALSEARCH	1, 2, 3, and 4
PHOTOGRAPHICSEARCH	3, 4, and 5
PHOTOARCHIVE	5
PHOTOCOMPARE	4 and 5
RESULTREPORT	2 and 5

Table 5.1: Actions provided by the hosts in the plan evaluation benchmarking experiment.

Camera	Resolution
Camera 1	3.2
Camera 2	8.0

Table 5.2: Camera properties as resources in the plan evaluation benchmarking experiment.

Node	Speed (max mph)	Transportation Cost (per mile)
Node 1	30	6.0
Node 2	40	6.5
Node 3	20	5.1
Node 4	10	4.9
Node 5	45	6.2

Table 5.3: Network node properties as resources in the plan evaluation benchmarking experiment.

Figure 5.1: Geographical map of the topology of locations, resources, and a network overlay.

As seen in the figures, guiding the search strategy using partial-plan evaluators yielded a greater variety of full-plan evaluations than the random and I-Plan search strategies. For these plan evaluators, the guided search evaluations had comparable or higher standard deviations than both the Random and I-Plan search strategies (see Table 5.4). Network hops served as a link quality evaluator for these experiments.

The first part of the experiments focused on the Guided algorithm's ability to produce qualitatively-different plans. Another desired quality for search algorithms is the ability to produce dominant plans in respect to evaluation criteria. As with the first set of experiments, I used the following plan evaluation criteria:

- Number of Network Hops (full data path),

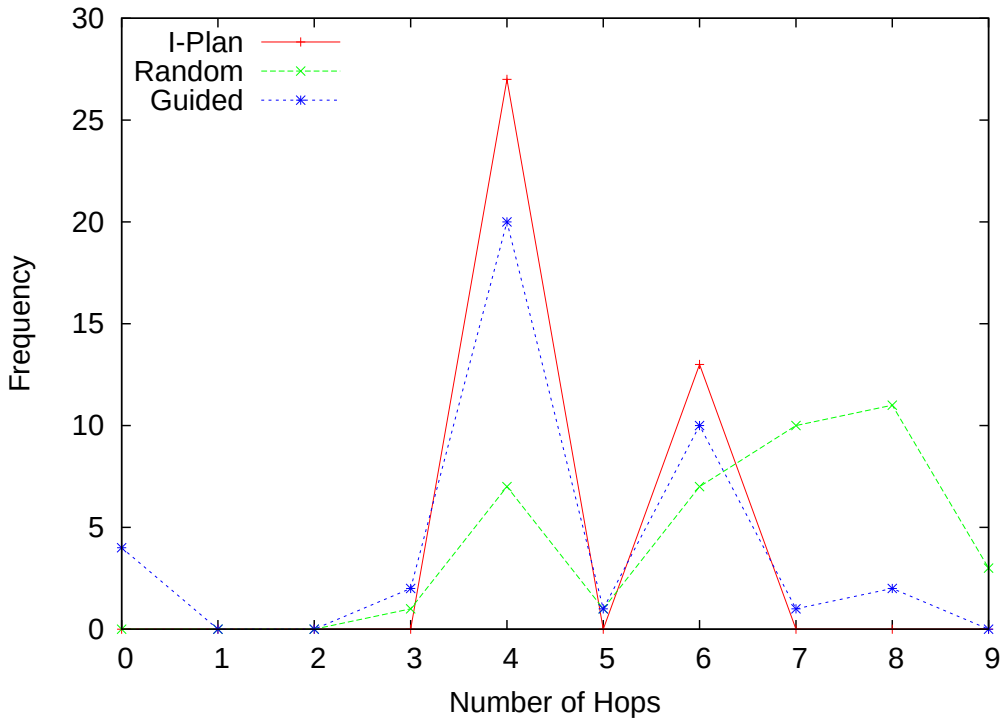

Figure 5.2: Network hop evaluation frequency distribution.

- Network Bandwidth Usage (Mbps),
- IED Detection Accuracy (%), and
- Plan Execution Time.

For the **Dominant Plan** experiment, I generated forty plans using each planning strategy, keeping track of the dominant plans among all strategies. The purpose of this experiment is to show that, while the Guided algorithm can produce qualitatively-different plans, it can also generate dominant plans. In the experiment, 59.3% of unique dominant plans were generated by the Guided algorithm. See Table 5.5 for the results of the Dominant Plan experiment.

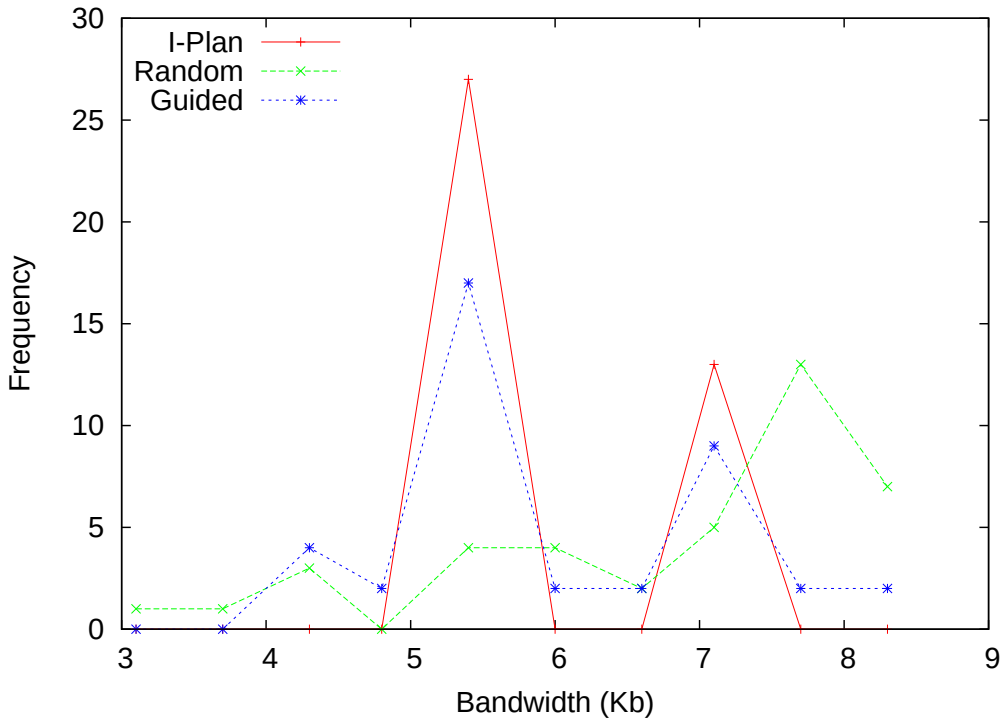

Figure 5.3: Network bandwidth evaluation frequency distribution.

By specifying plan evaluation criteria statistics for each plan evaluator, the notion of dominant plans is exploited. A limited set of dominant plans are presented to the task assigning agent for careful consideration since these are likely to be the best plan options.

The purpose of the empirical validation is to show that guiding the planner's search using multi-objective partial-plan heuristics results in a broad range of (full) plan evaluations. The distinction is between the partial plan evaluations which are guidance heuristics and full plan evaluations which run on fully-ground, complete plans. The standard deviations of each plan evaluation (over the set of plans produced by the planner in its allotted time) in Table 5.4 show that the guided algorithm produced a broad range of plan evaluations.

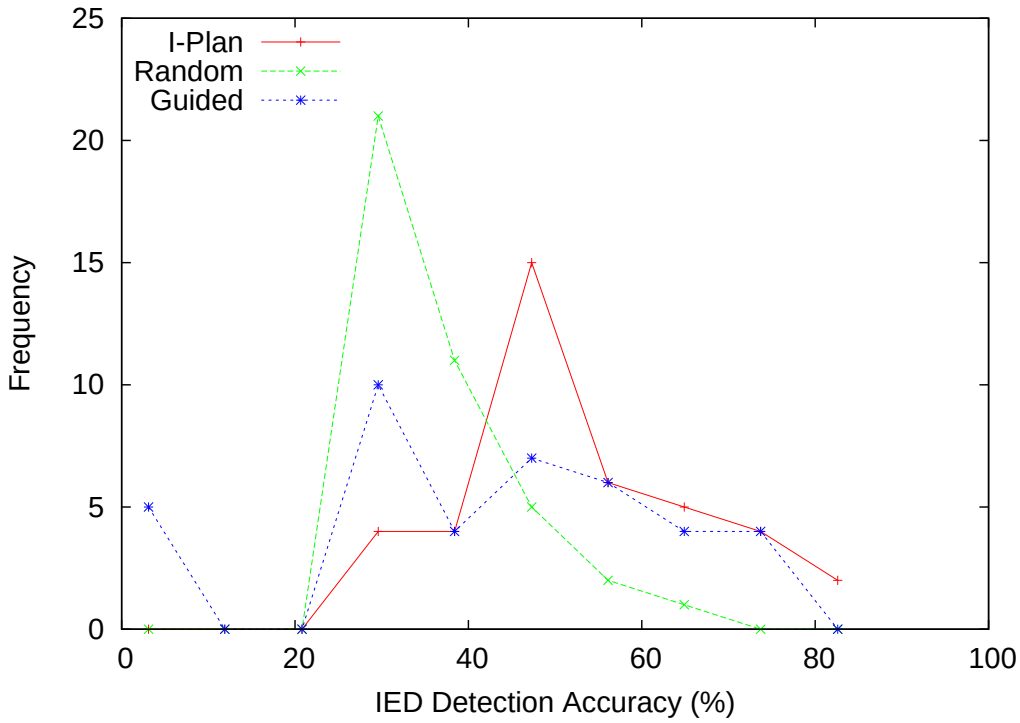

Figure 5.4: IED detection accuracy evaluation frequency distribution.

The percentage of dominant plans produced by each algorithm in Table 5.5 serves as a comparison of algorithm effectiveness.

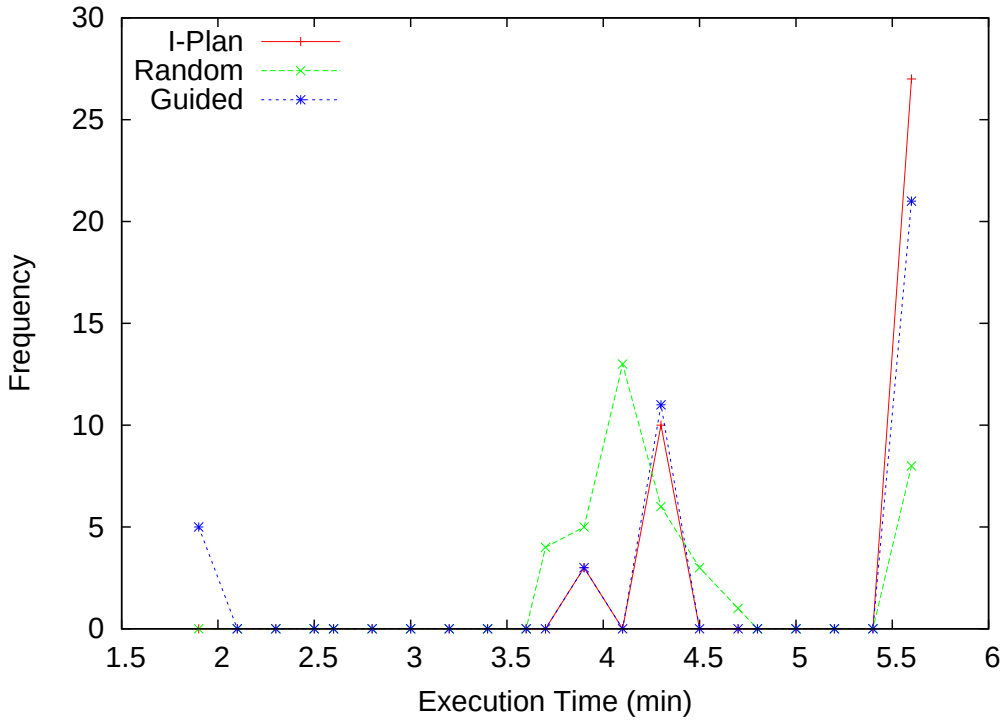

Figure 5.5: Execution time evaluation frequency distribution.

	Network Hops	Bandwidth (Kb)	IED Acc. (%)	Exec Time (sec)
I-Plan Default	0.949	0.759	29.1	0.730
Random	1.647	**1.476**	17.8	0.642
Guided	**1.916**	1.141	**39.3**	**1.249**

Table 5.4: Standard deviations of plan evaluations for each search strategy show that my *Guided* search strategy yields the most qualitatively-different plans in the "Search Guidance Using Plan Evaluation Criteria" experiment.

Search Strategy	**% Dominant Plans Produced**
Guided	59.3%
I-Plan Default	7.4%
Random	33.3%

Table 5.5: The percentage of dominant plans produced by each search strategy in the "Dominant Plans" experiment.

5.2 Network-Aware Agent Combinations

The goal of these experiments is to show which combinations of agent implementations yield the best plan execution. To accomplish this, I ran all combinations of planning, execution, and monitoring agents through the IED detection scenario in a network emulator. The types of agents and their techniques are listed below:

Agent	**Technique**
Planning	Random
	(I-Plan) Domain-independent
	Guided
Execution	Naïve
	Reactive
	Proactive (Sensing)
Monitoring	Data-driven
	Analytical
	(none)

My evaluation of plan execution includes goal satisfaction, execution time, network bandwidth usage, and IED detection accuracy.

5.2.1 Experiment Setup

This section describes the implementation of the agents and the network emulator in which the experiments were run.

CORE

All of the experiments in this section were executed in the Naval Research Laboratory's Common Open Research Emulator (CORE) [1]. CORE is a framework for emulating networks on a single computer. Using FreeBSD network stack virtualization, CORE allows

for heterogeneous networks to be emulated. Furthermore, the mobility of hosts in CORE can be controlled using scenario mobility scripting. Using these features, a large emulation scenario can be deployed and controlled by a single GUI.

All of the core hosts are running Simple Multicast Forwarding (SMF) [39] for multicast packet forwarding and Open Shortest Path First (OSPF) as a unicast routing protocol.[1] Figure 5.6 shows a screenshot of the CORE nodes configured to run the IED detection scenario.

CORE supports loading "mobility models," which dictate the geographical movement of hosts while CORE is running. I implemented three mobility models to show how different types of network dynamism affect plan execution and monitoring. All scenarios started with the hosts positioned as shown in Figure 5.6.

The first mobility model, *static*, had no host movement. The nodes all remained in the positions shown in Figure 5.6. The second, *dynamic*, moved the hosts as dictated by the PHYSICALMOVE and PHYSICALMOVETOCAMERA actions. The last, *partition/merge* or *part-merge*, split the network into two "islands" (as shown in Figure 5.7) and merged the network back together. This mobility model caused a complete disconnect between the two islands.

Experimental Process

All planning agents presented in this thesis were implemented using Tate *et al.* 's plan-space HTN planner, I-Plan [11]. For descriptions of the implementations, see Chapter 4.

After the domain and each of the agent types were implemented, each of the planning agents produced plans for five minutes. From these plans, a MOO function is used to select a single plan from each of the set produced.

The plans are passed to execution agents running in CORE. Other hosts in core are

[1] More information on OSPF at `http://www.ietf.org/html.charters/ospf-charter.html`.

Figure 5.6: Screenshot of the IED detection scenario running in the network emulator, CORE. The blue circles are hosts in CORE and the green lines represent network links. The white rectangles in the background of CORE are used to show the starting locations of the hosts for the IED detection scenario.

running the services that correspond to the plan actions. All hosts that house services also run monitoring agents (unless no monitoring agents are specified by the test). The monitoring agents report faults they detect to the execution agent.

Experimental Trials

Each experimental run, or trial, was differentiated by:

- The implementation of the planning agent used;

Figure 5.7: Screenshot of the partition/merge mobility model running in CORE. The blue circles are the CORE hosts, the white rectangles show the starting positions.

- The implementation of the execution agent used;
- The implementation of the monitoring agents used; and
- The type of network dynamism under which the agents operated.

Because CORE uses FreeBSD network stack emulation, different results were experienced on each run — even when all the above variables were constant. Performance and effectiveness of the trials was judged based on the actions that completed successfully. For example, manually searching all of the locations yields a 90% IED detection accuracy. However, if one manual search does not complete, the IED detection accuracy decreases. Similarly, the

execution time depends on the actions executed, the order in which they are executed, and the resources that they utilize. Additionally, during each of the trials, all of the network traffic was logged to determine the network overhead associated with each unique run.

5.2.2 Planning Agent Comparisons

The first part of the experiment is finding the set of plans that each algorithm produces. The planning agents were allotted five minutes to produce all the plans they could.

Next, a single plan for each algorithm is selected from each set via the MOO function in Equation 5.2.2.1. This function represents the intentions of a human user at the mixed-initiative plan selection interface.

$$\begin{aligned} \text{MOO}(p_I) = &\text{IEDDetectAcc}(p_I) \\ &+ 3 \times \text{TranspCost}(p_I) \\ &+ 5 \times \text{ExecTime}(p_I) \\ &+ \text{HopCount}(p_I) \\ &+ \text{BandwidthUse}(p_I) \end{aligned} \tag{5.2.2.1}$$

Results

The domain-independent planning algorithm. I-Plan's default algorithm uses domain-independent metrics, preferring plans with fewer actions, using fewer resources. Since my method of modeling service distribution represents network nodes as resources, I-Plan's default algorithm favored plans using services on nodes that offered many services, not necessarily the network nodes in the best locations. Also, since manual IED searches require fewer planning actions, the domain-independent heuristics preferred them to photographic searches. An example of a plan generated by the domain-independent search is represented

below.[2] Note the *manualSearch* action being preferred to photograhic search and only *node1* and *node2* being used. Manual searching is preferable to the domain-independent algorithm because it requires fewer planning actions. Only two network nodes are used for the manual searching because the domain-independent algorithm prefers plans that utilize fewer resources.

```
checkForIEDAt  location1
manualSearch   node1 location1
physicalMove   node1 location1
conductScan    node1 location1
physicalMove   node2 location1
reportResults  node2 location1
checkForIEDAt  location2
manualSearch   node1 location2
physicalMove   node1 location2
conductScan    node1 location2
physicalMove   node2 location2
reportResults  node2 location2
checkForIEDAt  location3
manualSearch   node1 location3
physicalMove   node1 location3
conductScan    node1 location3
physicalMove   node2 location3
reportResults  node2 location3
```

The random planning algorithm. The random planning algorithm selects from branches of the search space randomly. The algorithm exhibits no preference toward any plan criteria — including actions and resources. After running the algorithm for five minutes, it produced over 100 plans. The plan that returned the highest MOO value (see Equation 5.2.2.1) is shown below.

None of the plans produced by the random algorithm contained "pointless" actions. I define pointless actions as those whose effects do not contribute to a goal state. The random

[2]The order of the plan actions appears modified for clarity.

algorithm avoided pointless actions because plan-space planners eliminate actions with no causal links to a goal state.

Although the random algorithm avoided pointless actions, it suffered from an inefficient use of resources. This inefficiency causes nodes to travel further and communicate over lower-quality network links. Furthermore, the overall IED detection accuracy suffers as a result of poor camera choice.

```
checkForIEDAt location1
photographicSearch node3 location1
physicalMoveToCamera node3 camera1
acquireCamera node3 location1 camera1
physicalMove node3 location1
getOldPhoto node5 to photo-0
takePhoto node3 location1 camera1 to photo-1
comparePhotos node4 photo-1 photo-0
reportResults node2 location1
checkForIEDAt location2
manualSearch node1 location2
physicalMove node1 location2
conductScan node1 location2
physicalMove node2 location2
reportResults node2 location2
checkForIEDAt location3
manualSearch node1 location3
physicalMove node1 location3
conductScan node1 location3
physicalMove node2 location3
reportResults node2 location3
```

The network-aware guided algorithm. The guided algorithm uses domain-dependent and network-aware plan evaluation criteria to serve as search heuristics. Plans produced by this algorithm reason over planning actions as well as the resources they use.

Of the plans produced by this algorithm in five minutes, the below had the highest MOO value (from Equation 5.2.2.1). The plan uses a combination of manual searching and photographic searching to balance the tradeoff between IED detection accuracy and

execution time.

```
checkForIEDAt location1
photographicSearch node5 location1
physicalMoveToCamera node5 camera2
acquireCamera node5 location1 camera2
physicalMove node5 location1
getOldPhoto node5 to photo-0
takePhoto node5 location1 camera2 to photo-1
comparePhotos node5 photo-1 photo-0
reportResults node5 location1
checkForIEDAt location2
manualSearch node3 location2
physicalMove node3 location2
conductScan node3 location2
physicalMove node5 location2
reportResults node5 location2
checkForIEDAt location3
manualSearch node4 location3
physicalMove node4 location3
conductScan node4 location3
physicalMove node2 location3
reportResults node2 location3
```

Figure 5.8 shows the average of actual execution times of each of the aforementioned plans using naïve and reactive execution agents and analytical and data-driven monitoring agents. For this baseline test, all the agents were running on a single host, so the network was not affecting the execution times. The test showed that all plans take longer to execute with the reactive execution agent and the analytic monitoring agent. This is expected because when using analytic monitoring, the reactive execution agent inserts monitoring plan actions into the plan. On average, the guided algorithm produced the fastest plan, followed closely by the domain-independent (I-Plan) plan and then random. The exception to this ordering is that the guided plan, monitored by the analytic monitor, has more actions and therefore requires more monitoring plan actions to be inserted — increasing the overall execution time slightly.

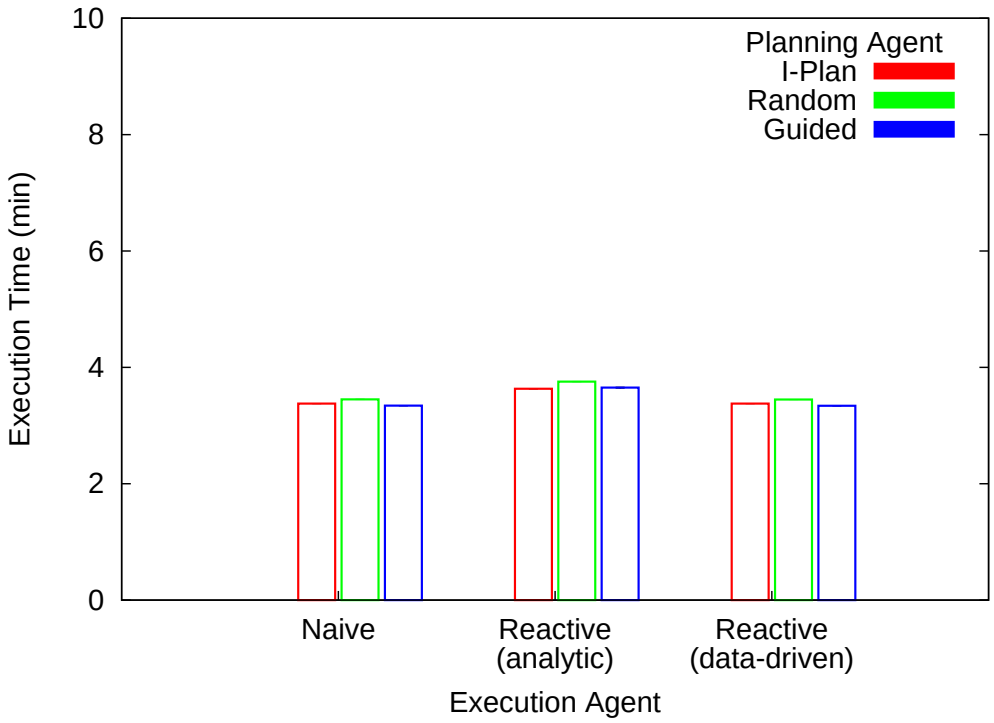

Figure 5.8: Mean plan execution times (in minutes) by plan, execution agent, and monitoring agent types. All agents and services are located on a single host so the network is not affecting the results.

Figure 5.9 illustrates the IED detection accuracy for each plan. The figure shows the IED detection accuracy of each plan assuming it executes to completion with no errors. The domain-independent algorithm (I-Plan) produced a plan with the highest IED detection accuracy, followed by the guided, and finally the random. These results are not surprising because manual searching yields a higher IED detection accuracy (but takes a longer time) than visual change detection. The guided algorithm attempts to exploit a tradeoff between these criteria so it makes sense to lie between the domain-independent and the random algorithms.

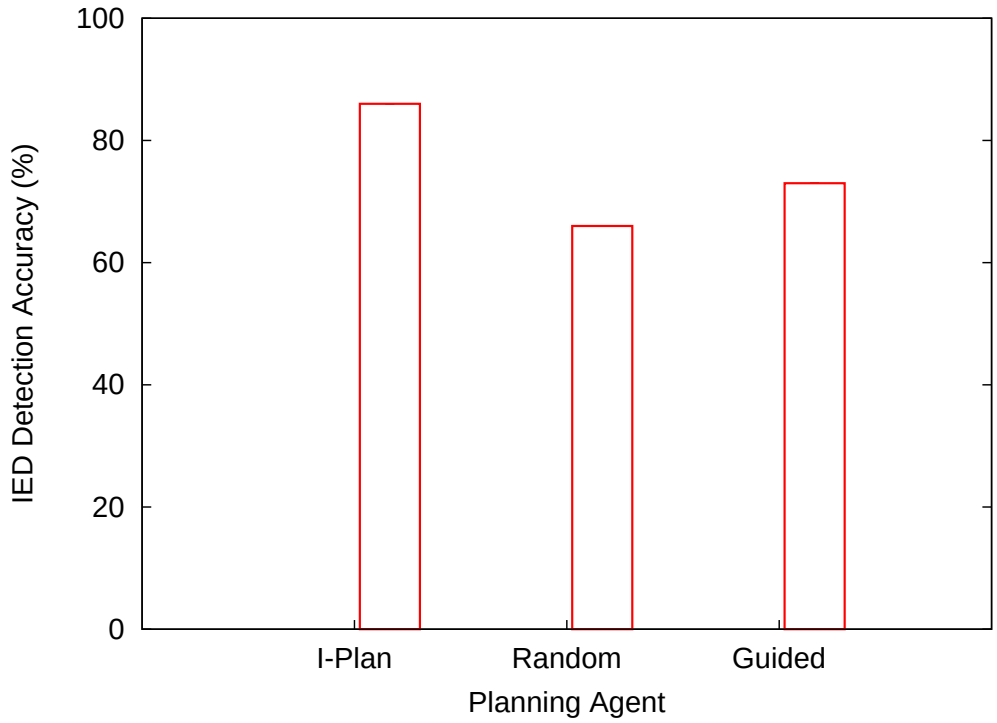

Figure 5.9: IED detection accuracy for each plan. This shows the IED detection accuracy of each plan assuming it executes to completion with no errors.

The proactive execution agent was not used for any of these experiments because all the agents and services were running on the same host. The proactive agent relies on differences in networking conditions to conduct network reasoning, and since all networking conditions are equal, the proactive agent's service choices are meaningless.

Figure 5.10 shows the average plan execution time for plans that executed successfully to completion by planning agent and network dynamism. Static mobility exhibits the least network dynamism, dynamic has varying link weights but never completely partitions the network, and partition-merge (part-merge) separates the network into two islands and then

merges the islands. The IED detection accuracy for each planning agent is not shown because it matches those under ideal conditions in Figure 5.9. Figure 5.10 indicates that dynamic link weights do not greatly affect the plan execution, but full network disruptions (such as those in the partition/merge mobility scenario) have an adverse effect on execution time. Furthermore, in the cases of static and dynamic mobility, the (network-aware) guided planning agent performed comparably to the domain-independent (I-Plan) and random planning agents. In the partition/merge scenario, however, it performed 16.7% faster than the next fastest (the domain-independent) planning agent and 28.8% faster than the random planning agent.

5.2.3 Execution Agent Comparisons

To compare execution agents, the plans from Section 5.2.2 are executed in CORE using each of the execution agents in various network environments.

Figure 5.11, Figure 5.12, and Figure 5.13 show the mean IED detection accuracy versus the mean plan execution time for each execution agent, separated by the domain-independent (I-Plan), random, and guided planning agents respectively. In all of these figures, the naïve execution agent has the lowest IED detection accuracy and plan execution time. This is because many of the plans executed by the naïve agent failed before completing. Also, in all experiments, the reactive and proactive agents achieved the same level of IED detection accuracy (equal to the ideal values). The only factor in their comparative performance was plan execution time.

Figure 5.11 shows the domain-independent planning agent working with each of the execution agents. The reactive execution agent completed slightly faster on average than the proactive agent, however both achieved their ideal IED detection accuracy values. The naïve execution agent, on the other hand, did not complete successfully — yielding a very fast execution time at the expense of a sub-optimal IED detection accuracy.

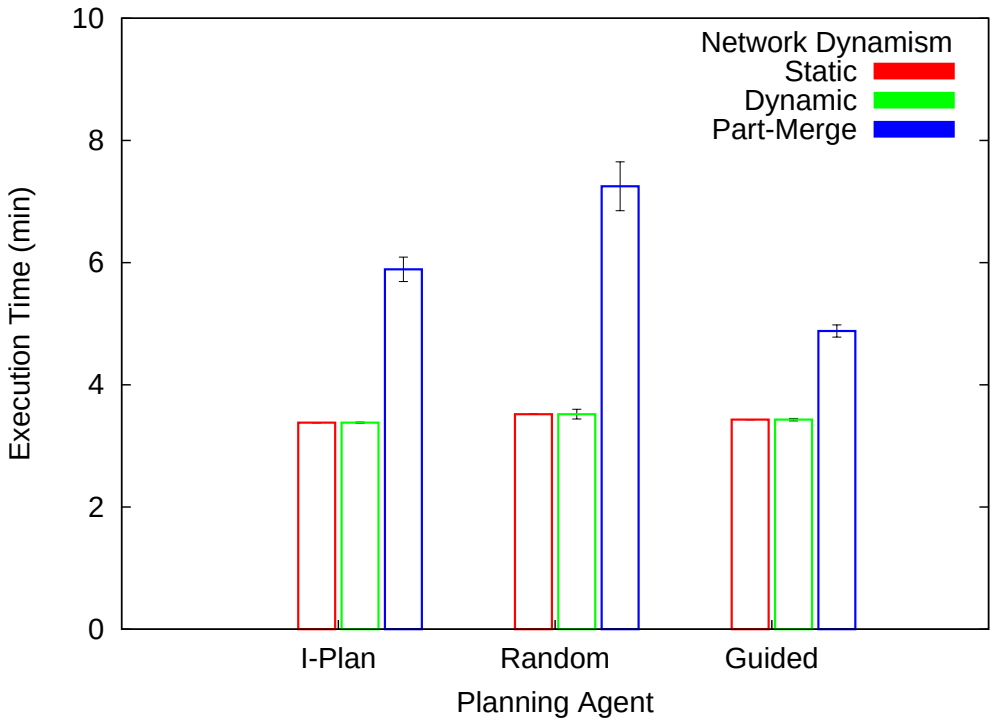

Figure 5.10: Mean plan execution time for plans that executed successfully to completion by planning agent and the network dynamism (as indicated by the mobility scenario). Static mobility exhibits the least network dynamism, dynamic has varying link weights but never completely partitions the network, and partition-merge (part-merge) separates the network into two islands and then merges the islands.

Figure 5.12 shows the random planning agent working with each of the execution agents. Again, the naïve execution agent did not complete successfully, yielding a fast execution time at the expense of a sub-optimal IED detection accuracy. For the random planning agent, however, the proactive execution agent was considerably faster on average than the reactive execution agent. This is probably because the random planning agent did not consider the network conditions at any point of the plan construction, so the reactive execution agent experienced faults for many of the plan's actions. The proactive did not

experience as many faults because it uses network-based logic before each plan action has an opportunity to fail.

Figure 5.13 shows the guided (network-aware) planning agent working with each of the execution agents. Using the guided planning agent with the naïve execution agent finished faster and with a higher IED detection accuracy on average than the domain-independent or random planning agents with the naïve execution agent. Also, the reactive and proactive execution agents completed faster using the guided planning agent than any other planning agent. This supports my results from the planning agent comparison.

Another important factor in Figure 5.13 is that the reactive agent completes considerably faster than the proactive agent. This means that the advice of the guided algorithm significantly helped the execution agent.

One factor to consider for the effectiveness of an execution agent is the impact on the network. An execution agent that uses fewer network resources is preferable so the network resources remain available for other applications. Figure 5.14 shows the average number of packets transmitted over the entire network by execution agent and network dynamism. The results show that the naïve and reactive execution agents behave similarly under connected mobility patterns (static and dynamic). The proactive agent uses slightly more network transmissions on average under connected mobility patterns, probably from the routing protocol conducting unnecessary network probes prior to every action. The naïve execution agent is not shown on the partition/merge scenario because none of the plans it executed in this environment completed successfully. The proactive agent, under partition/merge, sent fewer than half as many packets as the reactive agent. This is probably because of the partial transmissions that fail when the reactive agent attempts to act according to the fully-ground plan. The proactive agent avoids these unnecessary transmissions by updating its view of ω_H before each action execution.

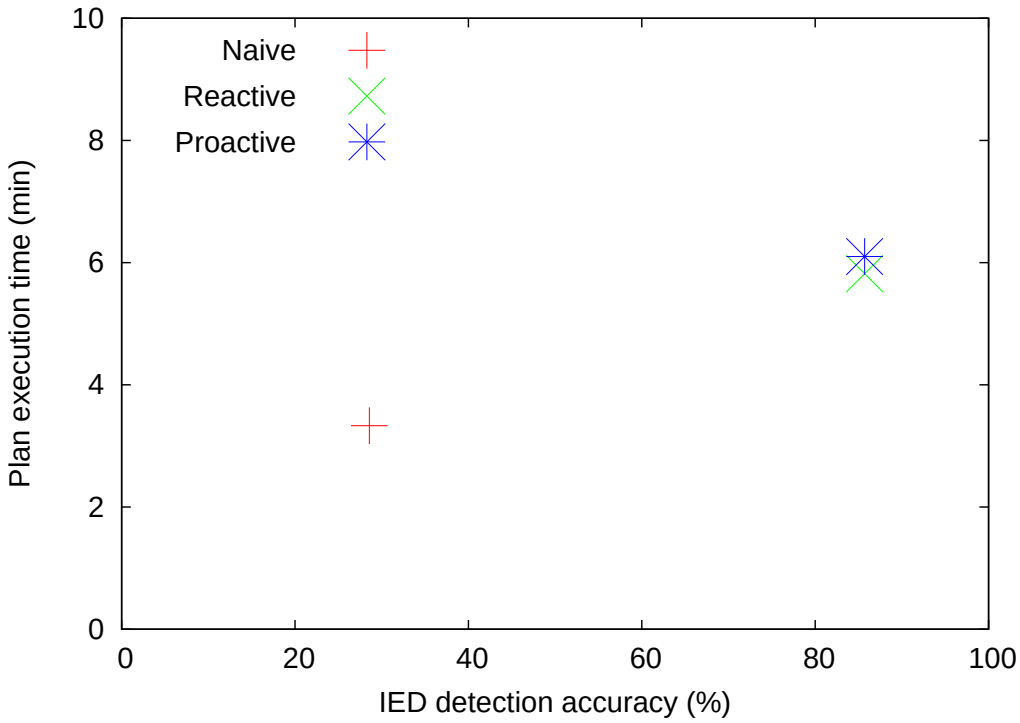

Figure 5.11: Mean IED detection accuracy versus mean plan execution time of the domain-independent planning agent in combination with each execution agent.

5.2.4 Monitoring Agent Comparisons

The analytic monitoring agents triggered a lot of false positives when there were errors in communicating (for computing the residual) between the monitoring agents. Also, the analytic monitor was an active monitor, one that has an (possibly detrimental) effect on the system, while the data-driven monitor is passive, having no effect on the system. Furthermore, implementing the distributed residual calculation was more challenging than implementing the data-driven monitor. Thus, analytic monitors are less suitable in distributed environments than data-driven monitors.

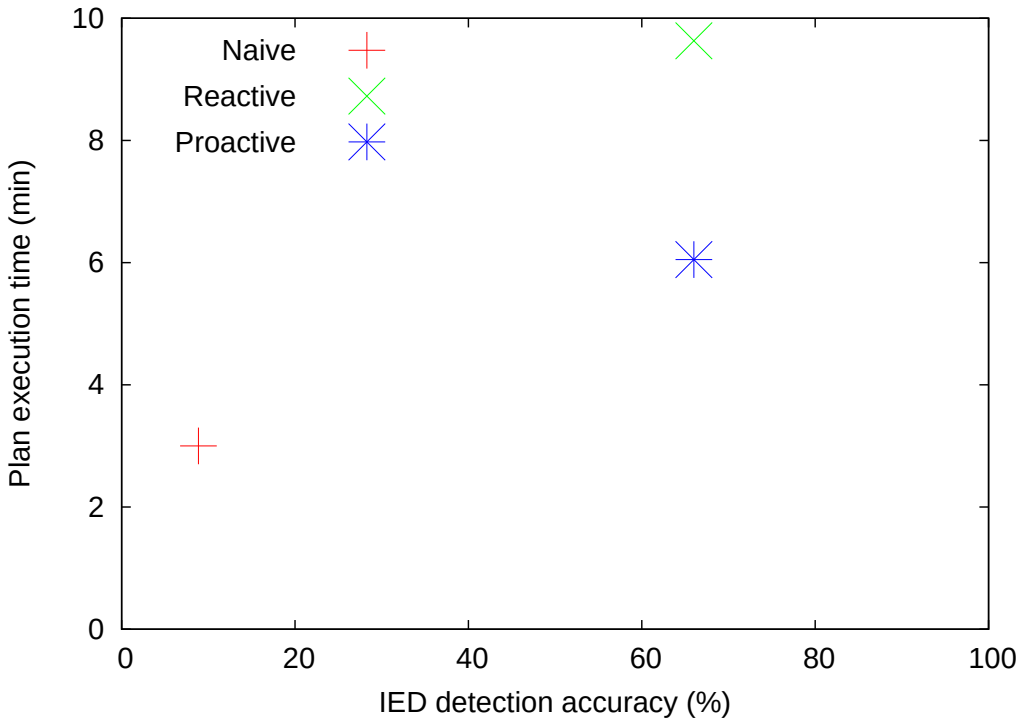

Figure 5.12: Mean IED detection accuracy versus mean plan execution time of the random planning agent in combination with each execution agent.

The data-driven monitors, on the other hand, were useful as indications of network-related plan execution failures. Figure 5.15 shows the execution of a plan that encountered no failures, whereas Figure 5.16 shows the number of retransmitted packets increases while the number of successful outgoing packets remains constant (or near-constant).

Over 54 trials, the data-driven network monitoring agents experienced only a 9.25% false-positive (type I error) rate and a 1.85% false-negative (type II error) rate. False-positives occur more often (five times as frequently) because only one of the monitoring agents has to malfunction for a false-positive to register. For a false-negative to register, a

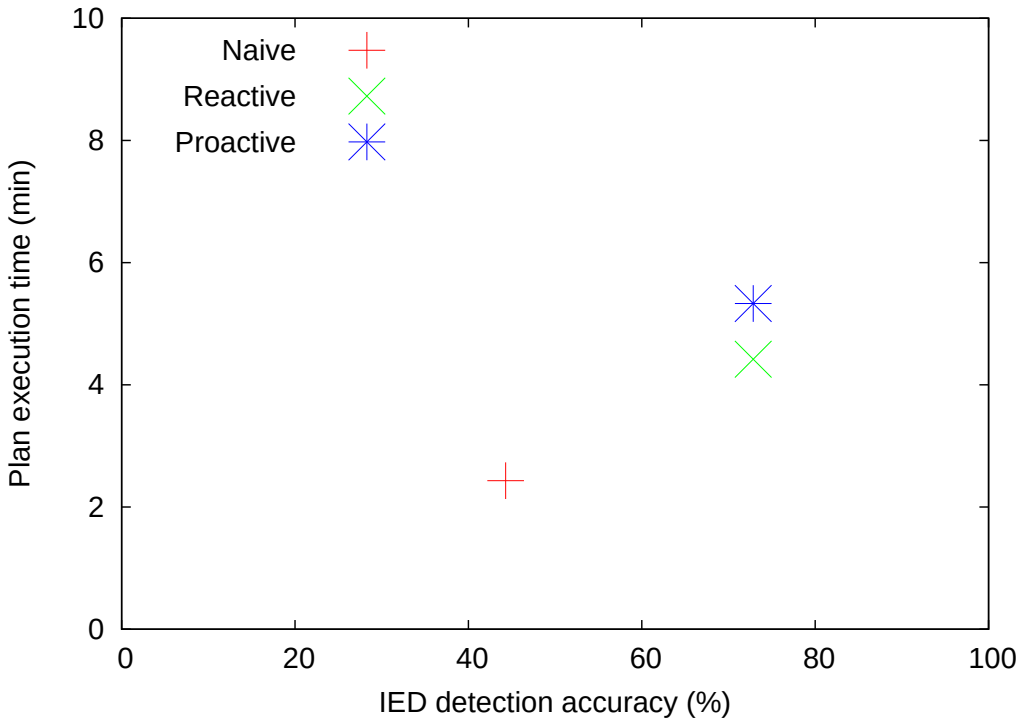

Figure 5.13: Mean IED detection accuracy versus mean plan execution time of the guided planning agent in combination with each execution agent.

network malfunction must go undetected by all of the monitoring agents.

5.3 Experimental Analysis

The goal of the experiments is to validate the following contributions:

- A novel method of generating qualitatively-different plans over a range of plan evaluators; and
- A comparison of the efficiency and performance of network-aware planning, execution, and monitoring agents.

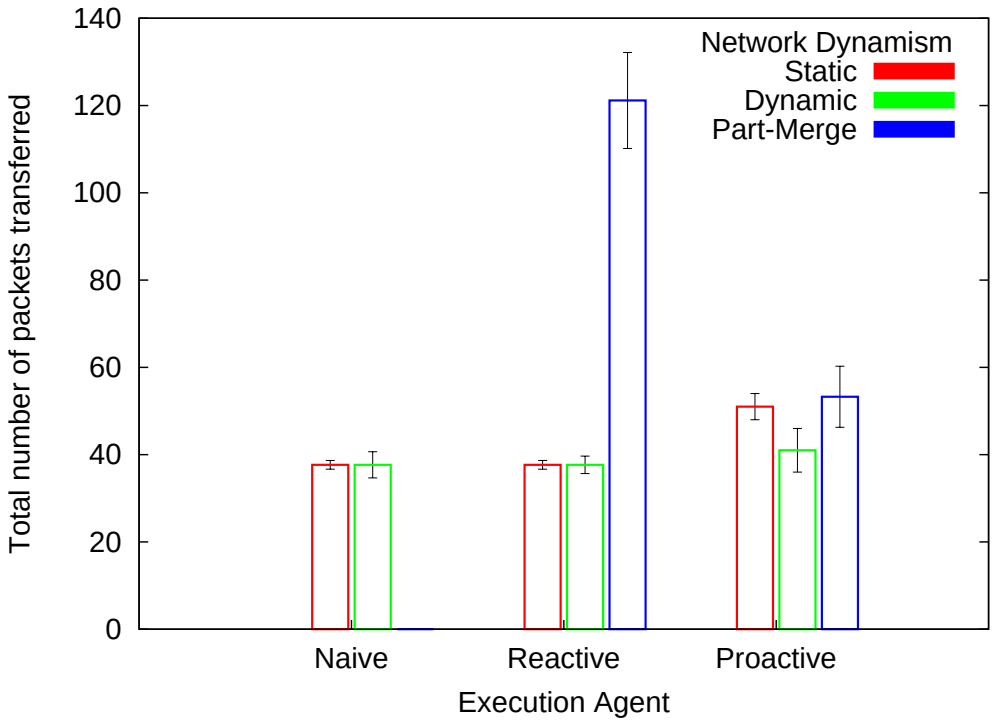

Figure 5.14: Average number of packets transmitted across the entire network for various execution agents under different network dynamics. The naïve execution agent never successfully completed executing a plan in the partition/merge mobility scenario, so it is not included in this figure.

To do so, I show that network awareness in planning, execution, and monitoring agents increased the performance and effectiveness of the agents. The former contribution is validated by showing that network-aware modifications to the planning agent cause it to return a greater diversity of plans and the network-aware plans have higher evaluations on average than those produced by other planning agents. Next, with the help of a network emulator, I show that network-aware plans perform better than their network-unaware peers when executed under network dynamism, in order to validate the later contribution. All experiments

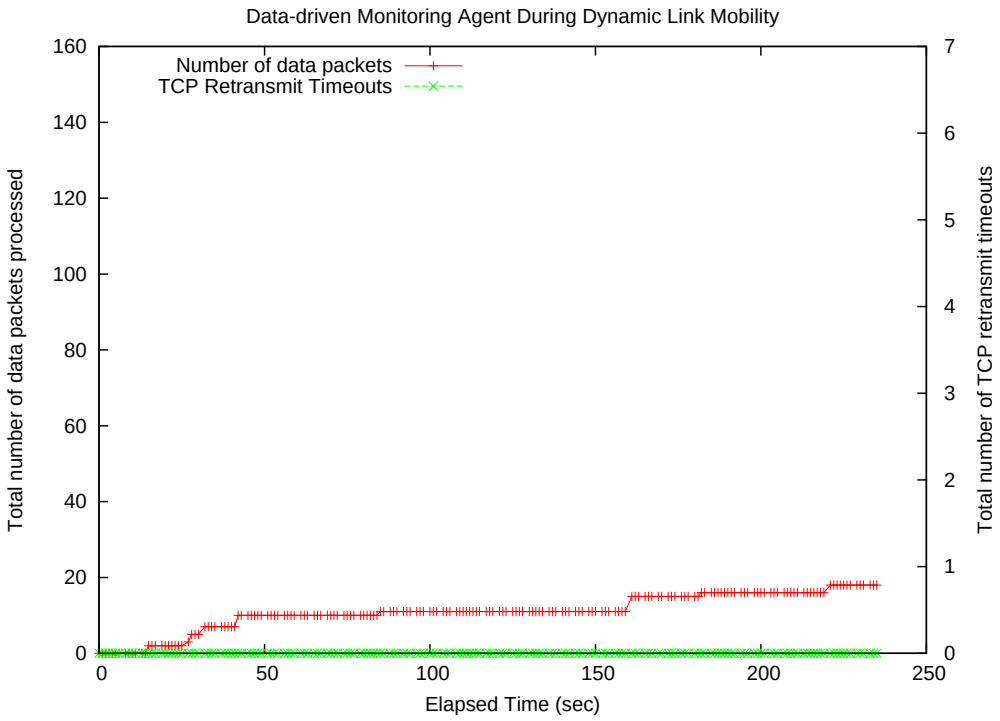

Figure 5.15: Network statistics collected by the data-driven monitoring agent during the dynamic link weight mobility scenario. During this scenario, no faults occurred.

were conducted in the context of the motivating IED detection scenario.

Table 5.4 shows that the novel method of generating qualitatively different plans produced on average a wider variety of plans than random and domain-independent planning algorithms. Table 5.5 shows that, while the guided algorithm produces plans with the greatest variety of plan evaluations, it also generated more dominant plans than any other algorithm. Together with the frequency distributions of plan evaluations in Figure 5.2, Figure 5.3, Figure 5.4, and Figure 5.5, the contribution in qualitatively different plan generation is validated.

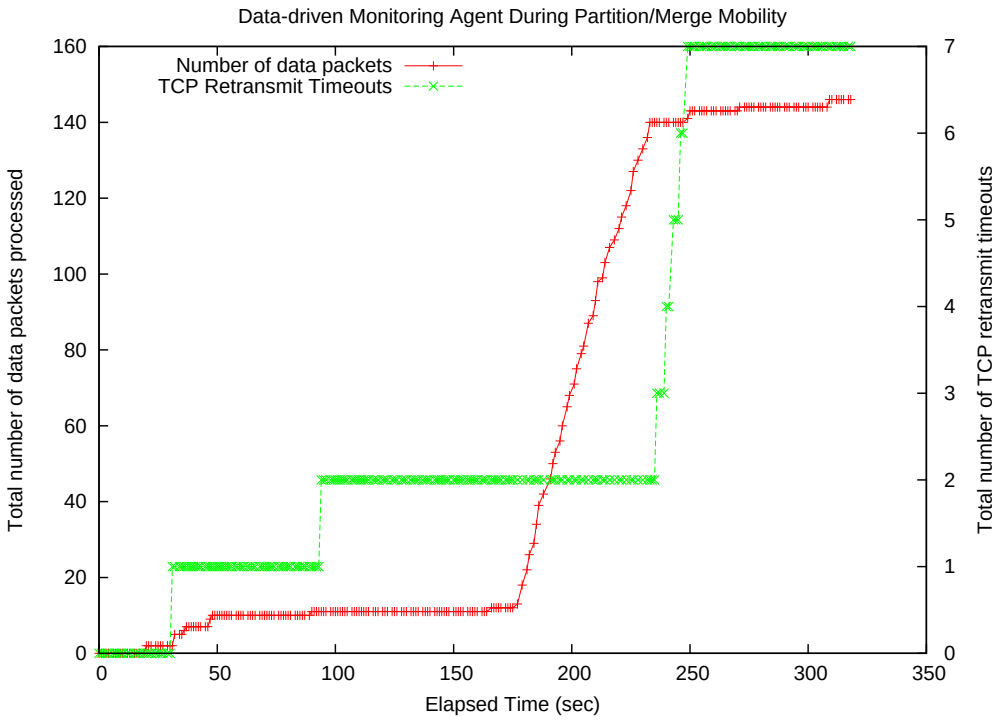

Figure 5.16: Network statistics collected by the data-driven monitoring agent during the partition/merge mobility scenario. During this scenario, failures in plan execution occurred due to lack of network connectivity.

The next contribution validated by empirical results is the comparison of effectiveness and performance of network-aware agents at planning, execution, and monitoring stages of the planning process. Figure 5.8 and Figure 5.14 show that the ideal combination of agents depends on the degree of dynamism in the network. On a mostly-static network, network-awareness in the planning agent showed improvement in performance of the solution. In these cases, the overhead imposed by network-aware execution agents decreased the overall effectiveness. When network links become completely severed, however, adding network-awareness to execution agents improved performance and effectiveness drastically. In these

cases, network-aware planning agents also improved the quality of the solution. Furthermore, experiments show that network-awareness during any stage of the planning process benefits plan execution. Figure 5.11 shows that the naïveexecution agent (which has no network-awareness) failed the most often, producing the lowest IED detection accuracy values; Figure 5.12 indicates that executing network-aware actions outperforms executing random actions; and Figure 5.13 shows that the network-aware planner improved the performance of execution agents.

Also, experiments were performed to determine which type of monitoring agent was effective at finding execution errors based on network conditions. Because the analytic monitor was an active monitor — one that uses the media that it monitors — analytic monitors are less suitable for disruption-prone environments. The data-driven monitors, on the other hand, experienced only 9.25% false-positives and 1.85% false-negatives, meaning they are useful for indications of network-related plan execution failures.

6. Conclusions

The contributions of this thesis are:

1. A comparison of methods for modeling network-centric extensions to the planning problem;
2. Reusable plan evaluators for network-aware planning agents;
3. A novel method of generating qualitatively-different plans over a range of plan evaluators; and
4. A comparison of the efficiency and performance of network-aware planning, execution, and monitoring agents.

The purpose of this thesis is to improve network-centric planning and execution by adding network-awareness to various stages of the planning process.

6.1 Network-Centric Planning Problem Extensions

Two methods of modeling service distribution in classical planning problems are compared. One method, labeled *operator distribution*, represents each service as a separate action in the planning domain. The other method, *resource distribution*, represents network hosts as planning resources. Operator distribution increases the complexity of the planning domain on the order of the number of hosts and unique services, whereas resource distribution increases the number of constraints in the world state on the same order. In choosing a method of modeling service distribution, it is important to consider the performance of the planner — operator distribution works well for planners that handle large numbers of actions, whereas resource distribution is better for planners with efficient resource allocation. Furthermore, duplicate services are better represented by resource distribution.

6.2 Network-Aware Plan Evaluators

Reusable plan evaluators for network-centric planning domains are presented. The evaluators represent end-user concerns when actors are communicating over a dynamic, heterogeneous network. Each evaluator contains two parts: a partial-plan evaluator and a full-plan evaluator. The two parts differ in that the partial-plan evaluator does not assume that temporal constraints are set (actions are ordered). The network-centric plan evaluators presented attempt to minimize bandwidth usage over network links with the highest quality. Using these evaluators for plan-space search guidance improved plan execution time and IED detection accuracy on average in simulated experiments.

6.3 Qualitatively Different Plans

An algorithm for guiding a plan-space planner toward qualitatively-different plan generation is presented. To judge differences between plans, the algorithm relies on users to specify domain-dependent and/or network-centric plan evaluation criteria. Using a GUI for visualizing plan evaluations, I discovered that interesting problems exhibited a natural trade-off between two or more plan evaluation criteria. On average, the algorithm produced a greater range of plan evaluations than domain-independent and random algorithms. Also, the large variety of the solutions was not at the expense of the solution quality. The guided algorithm produced 59.3% of the dominant plans produced by all three algorithms, compared to 33.3% by the random algorithm and 7.4% by the domain-independent algorithm.

6.4 Network-Aware Agents

Also presented are methods of incorporating network-awareness into planning, execution, and monitoring agents. These agents represent the stages of the planning process. Empirical results indicate that incorporating network-awareness into agents in dynamic,

heterogeneous networks improves overall system performance and effectiveness. The ideal combination of agents, however, depends on the degree of dynamism in the network. Empirical validation is presented in the context of a motivating IED detection scenario.

6.4.1 Network-Aware Planning Agent

A network-aware planning agent uses the qualitatively-different plan generation algorithm in conjunction with two network-centric plan evaluation criteria: bandwidth usage and link quality. Although experiences show that these two criteria alone do not produce sufficiently different plans, these criteria are the starting-point for a set of domain-reusable network-centric plan evaluators.

In a static, heterogeneous network, network-aware planning agents improved the performance of the solution by tailoring the plan to utilize lower-cost network links. On average, network-aware planning agents improved plan execution in all network-centric environments, but they show the most improvement in static, heterogeneous networks.

6.4.2 Network-Aware Execution Agents

There are two stages when execution agents can exhibit network-awareness: initial resource grounding (selecting a host on which to execute an action and the resources the action will use) and plan repair (after an execution fault). Two execution agents were created, both of which use network-proximity (obtained from a routing protocol) to augment or repair plan actions. The reactive execution agent only seeks to repair failed plan actions, whereas the proactive execution agent conducts resource grounding prior to executing an action in addition to performing reactive execution.

In static, heterogeneous networks, the overhead imposed by network-aware execution agents slightly decreased the overall effectiveness of the system. However, when the dynamism of the network increases, adding network-awareness to execution agents improved

execution performance and effectiveness drastically. Because network-aware execution agents only improved system effectiveness in highly-dynamic networks, they should only be used when network dynamism is possible.

6.4.3 Network-Aware Monitoring Agents

Two types of monitoring agents represent data-driven and analytic approaches to fault detection and isolation. Passive, data-driven monitoring agents outperformed analytic monitoring agents in disruption-prone environments. The data-driven monitoring agent experienced only 9.25% type I errors and 1.85% type II errors in the experiments.

6.5 Future Work

The experiments presented only used two of the three approaches to FDI. Future work will investigate the use of network-aware, domain-specific, knowledge-based monitoring agents for further improving the performance and effectiveness of monitoring agents in distributed environments.

Another area of future work will incorporate the effects of planning actions into heuristics. For example, a planning action that physically moves a host in a network will have some effect on the network-based plan evaluators. Reasoning about the effects of planning actions should drastically improve upon search guidance in qualitatively-different plan generation.

Bibliography

[1] J. Ahrenholz, C. Danilov, T.R. Henderson, and J.H. Kim. Core: A real-time network emulator. pages 1–7, Nov. 2008.

[2] Jorge A. Baier and Sheila A. McIlraith. Planning with preferences. *AI Magazine*, 29(4):25–36, 2009.

[3] V.R. Basili, G. Caldiera, and H.D. Rombach. The Goal Question Metric approach. *Encyclopedia of Software Engineering*, 1:528–532, 1994.

[4] Jim Blythe. An overview of planning under uncertainty. pages 85–110. 1999.

[5] Blai Bonet. Planning as heuristic search. *Artificial Intelligence*, 129:5–33, 2001.

[6] Blai Bonet, Gábor Loerincs, and Héctor Geffner. A robust and fast action selection mechanism for planning. In *Proceedings of the 14th National Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference (AAAI-97/IAAI-97)*, pages 714–719, Menlo Park, July 1997. AAAI Press.

[7] A. Bouguerra, L. Karlsson, and A. Saffiotti. Semantic knowledge-based execution monitoring for mobile robots. In *Proceedings of the International Conference on Robotics and Automation*, pages 3693–3698, 2007.

[8] C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11(1):94, 1999.

[9] L.H. Chiang, E. Russell, and R.D. Braatz. *Fault detection and diagnosis in industrial systems*. Springer, 2001.

[10] Alessandro Cimatti and Marco Roveri. Conformant planning via model checking. In *Proceedings of the European Conference on Planning*, pages 21–34. Springer-Verlag, 1999.

[11] K. Currie and A. Tate. O-Plan: The Open Planning Architecture. *Artificial Intelligence*, 52(1):49–86, 1991.

[12] Thomas Dean, Leslie Pack Kaelbling, Jak Kirman, and Ann Nicholson. Planning with deadlines in stochastic domains. In *Proceedings of the National Conference on Artificial Intelligence*, pages 574–579, 1993.

[13] Denise Draper, Steve Hanks, and Daniel Weld. Probabilistic planning with information gathering and contingent execution. pages 31–36. AAAI Press, 1994.

[14] M. Drummond and J. Bresina. Anytime synthetic projection: Maximizing the probability of goal satisfaction. In *Proceedings of the Association for the Advancement of Artificial Intelligence*, pages 138–144, Boston, MA, 1990.

[15] S. Edelkamp. Cost-optimal symbolic pattern database planning with state trajectory and preference constraints. In *Proceedings of the Workshop on Preferences and Soft Constraints in Planning at the International Conference on Automated Planning and Scheduling*, 2006.

[16] Kutluhan Erol, James A. Hendler, and Dana S. Nau. UMCP: A sound and complete procedure for hierarchical task-network planning. In *Artificial Intelligence Planning Systems*, pages 249–254, 1994.

[17] Jerome A. Feldman and Robert F. Sproull. Decision theory and artificial intelligence ii: The hungry monkey. *Cognitive Science: A Multidisciplinary Journal*, 1(2):158–192, 1977.

[18] George Ferguson, James F. Allen, and Brad Miller. Trains-95: Towards a mixed-initiative planning assistant. In *Proceedings of the National Conference on Artificial Intelligence*, pages 70–77. AAAI Press, 1996.

[19] Richard E. Fikes, Peter E. Hart, and Nils J. Nilsson. Learning and executing generalized robot plans. pages 485–503, 1993.

[20] R.J. Firby. An investigation into reactive planning in complex domains. In *Proceedings of the National Conference on Artificial Intelligence*, pages 202–206, 1987.

[21] Robert James Firby. Adaptive execution in complex dynamic worlds. Technical report, 1989.

[22] Maria Fox and Derek Long. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20:2003, 2003.

[23] MP Georgeff and AL Lansky. Procedural knowledge. *Proceedings of the IEEE*, 74(10):1383–1398, 1986.

[24] M.P. Georgeff and A.L. Lansky. Reactive reasoning and planning. In *Proceedings of the National Conference on Artificial Intelligence*, pages 677–682. Seattle, WA, 1987.

[25] A. Gerevini and D. Long. Plan constraints and preferences in PDDL3: The language of the fifth international planning competition. *University of Brescia, Italy, Tech. Rep*, 2005.

[26] A. Gerevini and D. Long. Plan constraints and preferences in PDDL3. In *Workshop on Soft Constraints and Preferences in Planning in the International Conference on Automated Planning and Scheduling*, 2006.

[27] J. Gertler. *Fault detection and diagnosis in engineering systems*. CRC Press, 1998.

[28] Robert Goldman and Mark Boddy. Epsilon-safe planning. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 253–26, San Francisco, CA, 1994. Morgan Kaufmann.

[29] X. Gu, K. Nahrstedt, RN Chang, and C. Ward. QoS-assured service composition in managed service overlay networks. In *Proceedings of Distributed Computing Systems*, pages 194–201, 2003.

[30] Kristian J. Hammond. Explaining and repairing plans that fail. *Artificial Intelligence*, 45:173–228, 1990.

[31] David M. Hart, Scott D. Anderson, and Paul R. Cohen. Envelopes as a vehicle for improving the efficiency of plan execution. In *Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling and Control*, pages 71–76. Morgan Kaufmann, 1990.

[32] J. Hoffmann. The Metric-FF planning system: Translating and ignoring delete lists to numeric state variables. *Journal of Artificial Intelligence Research*, 20:291–341, 2003.

[33] Jörg Hoffmann, Piergiorgio Bertoli, Malte Helmert, and Marco Pistore. Message-based web service composition, integrity constraints, and planning under uncertainty: A new connection. *Journal of Artificial Intelligence Research*, 35:49–117, 2009.

[34] E.J. Horvitz, J.S. Breese, and M. Henrion. Decision theory in expert systems and artificial intelligence. *International Journal of Approximate Reasoning*, 2(3):247–302, 1988.

[35] R. Isermann and P. Balle. Trends in the application of model-based fault detection and diagnosis of technical processes. *Control Engineering Practice*, 5(5):709–719, 1997.

[36] Gal A. Kaminka, David V. Pynadath, and Milind Tambe. Monitoring teams by overhearing: A multi-agent plan recognition approach. *Journal of Artificial Intelligence Research*, 17:83–135, 2002.

[37] Sven Koenig. Optimal probabilistic and decision-theoretic planning using markovian decision theory. Technical Report UCB/CSD-92-685, EECS Department, University of California, Berkeley, May 1992.

[38] Nicholas Kushmerick, Steve Hanks, and Daniel S. Weld. An algorithm for probabilistic planning. *Artificial Intelligence*, 76(1-2):239–286, 1995.

[39] Y. Lacharite, Maoyu Wang, L. Lamont, and L. Landmark. A simplified approach to multicast forwarding gateways in MANET. pages 426–430, Oct. 2007.

[40] KB Lamine and F. Kabanza. History checking of temporal fuzzy logic formulas for monitoring behavior-based mobile robots. In *Proceedings of the IEEE International Conference on Tools with Artificial Intelligence*, page 312, Washington, DC, USA, 2000. IEEE Computer Society.

[41] J.A. Macedo and K. Lim. Adaptive Change Detection Methodology for Buried Mine and IED Detection from Space.

[42] SP Mahambre, M. Kumar, and U. Bellur. A taxonomy of QOS-aware, adaptive event-dissemination middleware. *IEEE Internet Computing*, 11(4):35–44, 2007.

[43] T.M. Mansell. A method for planning given uncertain and incomplete information. In *Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence*, pages 350–358, 1993.

[44] D. McDermott. A heuristic estimator for means-ends analysis in planning. In *Proceedings of the International Conference on Artificial Intelligence Planning Systems*, pages 142–149. AAAI Press, 1996.

[45] K. Miettinen. *Nonlinear multiobjective optimization*. Springer, 1999.

[46] K. Myers and T. Lee. Generating qualitatively different plans through metatheoretic biases. In *Proceedings of the National Conference on Artificial Intelligence*. AAAI Press, 1999.

[47] K. L. Myers. Metatheoretic plan summarization and comparison. In *Proceedings of the International Conference on Automated Planning and Scheduling*. AAAI Press, June 2006.

[48] Dana Nau, Tsz-Chiu Au, Okhtay Ilghami, Ugur Kuter, Hector Munoz-Avila, J. William Murdock, Dan Wu, and Fusun Yaman. Applications of SHOP and SHOP2. *IEEE Intelligent Systems*, 20(2):34–41, 2005.

[49] Dana Nau, Malik Ghallab, and Paolo Traverso. *Automated Planning: Theory & Practice*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[50] RJ Patton, CJ Lopez-Toribio, and FJ Uppal. Artificial intelligence approaches to fault diagnosis. In *IEE Colloquium on Condition Monitoring: Machinery, External Structures and Health (Ref. No. 1999/034)*, page 5, 1999.

[51] J.S. Penberthy and D. Weld. UCPOP: A sound, complete, partial order planner for ADL. In *Proceedings of the International Conference on Knowledge Representation and Reasoning*, pages 103–114, 1992.

[52] O. Pettersson, L. Karlsson, and A. Saffiotti. Model-free execution monitoring by learning from simulation. In *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, Helsinki, Finland, 2005.

[53] Ola Pettersson. Execution monitoring in robotics: A survey. *Robotics and Autonomous Systems*, 53:73–88, 2005.

[54] M. Peysakhov, D. Artz, E. Sultanik, and W. Regli. Network awareness for mobile agents on ad hoc networks. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 368–376, Washington, DC, USA, 2004. IEEE Computer Society.

[55] L. Pryor and G. Collins. Cassandra: Planning for contingencies (Technical report No. 41). *The Institute for Learning Sciences, Northwestern University*, 1993.

[56] Louise Pryor and Gregg Collins. Planning for contingencies: A decision-based approach. *Journal of Artificial Intelligence Research*, 4:287–339, 1996.

[57] Patrick Riley and Manuela Veloso. Planning for distributed execution through use of probabilistic opponent models. In *Proceedings of the International Conference on Artificial Intelligence Planning and Scheduling*, 2002.

[58] D. Roman, U. Keller, H. Lausen, J. de Bruijn, R. Lara, and M. Stollberg. Web service modeling ontology. *Applied Ontology*, 1(1):77–106, 2005.

[59] M. J. Schoppers. Universal plans for reactive robots in unpredictable environments. In John Mcdermott, editor, *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1039–1046, Milan, Italy, 1987. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA.

[60] E. Sirin, B. Parsia, D. Wu, J. Hendler, and D. Nau. HTN planning for web service composition using SHOP2. *Web Semantics Journal*, 2004.

[61] D. Smith. Choosing objectives in over-subscription planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, pages 393–401, 2004.

[62] Shirin Sohrabi, Natasha Prokoshyna, and Sheila. A. McIlraith. Web service composition via generic procedures and customizing user preferences. In *Proceedings of the International Semantic Web Conference*, pages 597–611, 2006.

[63] B. Srivastava, S. Kambhampati, M. H. Do, and T. Nguyen. Finding inter-related plans. In *Proceedings of the International Conference on Automated Planning and Scheduling Workshop on Plan Analysis and Management*, 2006.

[64] L.A. Suchman and L.A. Suchman. *Plans and situated actions: The problem of human-machine communication*. Cambridge University Press, 1987.

[65] A. Tate and J. Dalton. O-Plan: a common Lisp planning web service. In *Proceedings of the International Lisp Conference 2003*, pages 12–25, 2003.

[66] A. Tate, J. Dalton, and J. Levine. Generation of multiple qualitatively different plan options. In *Proceedings of International Conference on AI Planning Systems*, pages 27–35, 1998.

[67] Austin Tate. Generating project networks. In *IJCAI*, pages 888–893, 1977.

[68] Austin Tate. Authority management - coordination between task assignment, planning and execution. In *Proceedings of International Joint Conferences on Artificial Intelligence Workshop on Knowledge-based Production Planning, Scheduling and Control*, 1993.

[69] Austin Tate. Coalition task support using I-X and <I-N-C-A>. In Vladimír Marík, Jörg P. Müller, and Michal Pěchouček, editors, *CEEMAS*, volume 2691 of *Lecture Notes in Computer Science*, pages 7–16. Springer, 2003.

[70] K. Usbeck, M. Chase, T. Wambold, C. Rumford, A. Kaplan, and W. Regli. Assessment of WiMAX for command and control applications in urban environments. Technical report, Project ACIN, 2009.

[71] Kyle Usbeck, William C. Regli, Gerhard Wickler, and Austin Tate. Finding dominant plans using plan evaluation criteria. In *Proceedings of Knowledge Systems for Coalition Operations*, pages 36–45, Chilworth Manor, Southampton, UK, MAR 2009.

[72] M. van den Briel, S. Kambhampati, and T. Vossen. Planning with preferences and trajectory constraints by integer programming. In *Proceedings of the Workshop on Preferences and Soft Constraints at the International Conference on Automated Planning and Scheduling*, 2006.

[73] D. E. Wilkins. Recovering from execution errors in SIPE. AI Center Technical Note 346, SRI International, January 1985.

[74] David E. Wilkins. Can AI planners solve practical problems? *Computational Intelligence*, 6(4):232–246, 1990.

[75] David E. Wilkins, Thomas J. Lee, and Pauline Berry. Interactive execution monitoring of agent teams. *Journal of Artificial Intelligence Research*, 18:217–261, 2003.

[76] E. Wilkins, Karen L. Myers, and John D. Lowrance. Planning and reacting in uncertain and dynamic environments. *Journal of Experimental and Theoretical AI*, 7, 1995.

[77] World Wide Web Consortium. Owl-s: Semantic markup for web services, November 2004. http://www.w3.org/Submission/OWL-S/.

[78] G. Wu, M. Mizuno, and PJM Havinga. MIRAI architecture for heterogeneous network. *IEEE Communications Magazine*, 40(2):126–134, 2002.

[79] Shlomo Zilberstein. On the utility of planning. *In M. Pollack (Ed.), SIGART Bulletin Special Issue on Evaluating Plans, Planners, and Planning Systems*, 6:6–1, 1995.

Appendix A. IED Detection Scenario Domain

```
1  ;;; Kyle Usbeck
2  ;;; IED Detection Domain
3
4  (domain (name "IED Detection"))
5
6  ;; sweep multiple locations for IEDs (TODO add more locations)
7  (refinement sweepForIEDs (sweepForIEDs)
8    (nodes
9      (1 (checkForIEDAt location1))
10     (2 (checkForIEDAt location2))
11     (3 (checkForIEDAt location3))
12 ;    (4 (checkForIEDAt location4))
13   )
14   (constraints
15 ;     (world-state condition (type ?node1) = node)
16 ;     (world-state condition (type ?node2) = node)
17 ;     (world-state condition (type ?node3) = node)
18 ;     (world-state condition (type ?node4) = node)
19   )
20   (annotations
21     (comments = "")))
22
23 ;; check for IED at a single location
24 (refinement checkForIEDAt (checkForIEDAt ?location)
25   (variables ?node ?location)
26   (constraints
27     (world-state condition (type ?node) = node)
28 ;    (world-state condition (checkForIEDAt ?node) = true)
29     (world-state condition (type ?location) = location)
```

```
30      (world-state condition (checked ?location) = false)
31      (world-state condition (searched ?location) = true)
32      (world-state effect (checked ?location) = true))
33    (annotations
34      (comments = "")))
35
36  ;; perform a manual search of a location for an IED
37  (refinement manualSearch (manualSearch ?node ?location)
38    (variables ?node ?resultReporterNode ?location)
39    (nodes
40      (1 (physicalMove ?node ?location))
41      (2 (conductScan ?node ?location))
42      (3 (physicalMove ?resultReporterNode ?location))
43      (4 (reportResults ?resultReporterNode ?location)))
44    (constraints
45      (world-state condition (type ?node) = node)
46      (world-state condition (manualSearch ?node) = true)
47      (world-state condition (type ?resultReporterNode) = node)
48      (world-state condition (resultReport ?resultReporterNode) = true)
49      ; manualSearch implies ?node is searchResource
50      (world-state condition (type ?location) = location)
51      (world-state condition (searched ?location) = false)
52      (world-state effect (searched ?location) = true))
53    (annotations
54      (comments = "")))
55
56  ;; perform a photographicSearch of a location for an IED
57  (refinement photographicSearch (photographicSearch ?node ?location)
58    (variables ?node ?photoArchiveNode ?location
59               ?camera ?photo ?oldPhoto ?photoCompNode
60               ?resultReporterNode )
61    (nodes
```

```
62      (1 (acquireCamera ?node ?location ?camera))
63      (2 (physicalMove ?node ?location))
64      (3 (takePhoto ?node ?location ?camera to ?photo))
65      (4 (getOldPhoto ?photoArchiveNode to ?oldPhoto))
66      (5 (comparePhotos ?photoCompNode ?photo ?oldPhoto))
67      (6 (reportResults ?resultReporterNode ?location)))
68    (constraints
69      (world-state condition (type ?node) = node)
70      (world-state condition (photographicSearch ?node) = true)
71      ; photographicSearch implies ?node is a photoResource
72      (world-state condition (type ?location) = location)
73      (world-state condition (searched ?location) = false)
74      (world-state condition (type ?photoArchiveNode) = node)
75      (world-state condition (type ?photoCompNode) = node)
76      (world-state condition (photoArchive ?photoArchiveNode) = true)
77      (world-state condition (photoCompare ?photoCompNode) = true)
78      (world-state effect (searched ?location) = true))
79    (annotations
80      (comments = "")))
81
82 ;; acquire a camera to take a photo of location using photoResource
83 ; Note: this looks wierd on the map because it essentially moves the
84 ; node to the camera at the same time as moving the camera to the
85 ; picture destination, but it works!
86 (refinement acquireCamera (acquireCamera ?node ?location ?camera)
87    (variables ?node ?location ?camera
88               ?camlat ?camlong ?loclat ?loclong)
89    (nodes
90      (1 (physicalMoveToCamera ?node ?camera)))
91    (constraints
92      (world-state condition (type ?node) = node)
93      ; type node implies ?node is a network node
```

```
94      (world-state condition (photographicSearch ?node) = true)
95      ; photographicSearch implies ?node is a photoResource
96      (world-state condition (type ?camera) = camera)
97      (world-state condition (latitude ?camera) = ?camlat)
98      (world-state condition (longitude ?camera) = ?camlong)
99      (world-state condition (latitude ?location) = ?loclat)
100     (world-state condition (longitude ?location) = ?loclong)
101     (world-state condition (inUse ?camera) = false)
102     (world-state effect (latitude ?camera) = ?loclat)
103     (world-state effect (longitude ?camera) = ?loclong)
104     (world-state effect (inUse ?camera) = ?node))
105   (annotations
106     (comments = "")))
107
108 ;; release the camera from use (so others can use it)
109 (refinement releaseCamera (releaseCamera ?camera)
110   (variables ?node ?camera)
111   (constraints
112     (world-state condition (type ?camera) = camera)
113     (world-state condition (inUse ?camera) = ?node)
114     (world-state condition (type ?node) = node)
115     (world-state effect (inUse ?camera) = false))
116   (annotations
117     (comments = "")))
118
119
120 ;; take a photo of a location with a camera
121 (refinement takePhoto (takePhoto ?node ?location ?camera to ?photo)
122   (variables ?node ?camera ?location ?photo
123 ;             ?camlat ?camlong ?loclat ?loclong
124              )
125   (constraints
```

```
126     (world-state condition (type ?node) = node)
127     (world-state condition (photographicSearch ?node) = true)
128     ; implies node is capable of using camera
129     (world-state condition (type ?location) = location)
130     (world-state condition (inUse ?camera) = ?node)
131     ; camera is in use by node
132     ; location of camera...
133     ;(world-state condition (latitude ?camera) = ?camlat)
134     ;(world-state condition (longitude ?camera) = ?camlong)
135     ;(world-state condition (latitude ?location) = ?loclat)
136     ;(world-state condition (longitude ?location) = ?loclong)
137     ;(world-state condition (latitude ?camera) = ?loclat)
138     ;(world-state condition (longitude ?camera) = ?loclong)
139     )
140   (annotations
141     (output-objects = ((?photo photo)))))
142
143 ;; get an old photo from an archive
144 (refinement getOldPhoto (getOldPhoto ?node to ?oldPhoto)
145   (variables ?node ?oldPhoto)
146   (constraints
147     (world-state condition (type ?node) = node)
148     (world-state condition (photoArchive ?node) = true))
149   (annotations
150     (output-objects = ((?oldPhoto photo)))))
151
152 ;; compare a photo and an old photo for IED possibilities
153 (refinement comparePhotos (comparePhotos ?node ?photo ?oldPhoto)
154   (variables ?node ?photo ?oldPhoto)
155   (constraints
156     (world-state condition (type ?node) = node)
157     (world-state condition (photoCompare ?node) = true)
```

```
158     ;(world-state condition (type ?photo) = photo)
159     ;(world-state condition (type ?oldPhoto) = photo)
160     )
161   (annotations
162     (comments = "")))
163
164 ;; do a physical move to location
165 (refinement physicalMove (physicalMove ?node ?location)
166   (variables ?node ?location ?latitude ?longitude)
167   (constraints
168     (world-state condition (type ?node) = node)
169     (world-state condition (type ?location) = location)
170     (world-state condition (latitude ?location) = ?latitude)
171     (world-state condition (longitude ?location) = ?longitude)
172     (world-state effect (latitude ?node) = ?latitude)
173     (world-state effect (longitude ?node) = ?longitude))
174   (annotations
175     (comments = "")))
176
177 ;; do a physical move to a lat/lng
178 (refinement physicalMoveToCamera (physicalMoveToCamera ?node ?camera)
179   (variables ?node ?camera ?latitude ?longitude)
180   (constraints
181     (world-state condition (type ?node) = node)
182     (world-state condition (type ?camera) = camera)
183     (world-state condition (latitude ?camera) = ?latitude)
184     (world-state condition (longitude ?camera) = ?longitude)
185     (world-state effect (latitude ?node) = ?latitude)
186     (world-state effect (longitude ?node) = ?longitude))
187   (annotations
188     (comments = "")))
189
```

```
190 ;; conduct a manual scan of a location
191 (refinement conductScan (conductScan ?node ?location)
192   (variables ?node ?location ?latitude ?longitude)
193   (constraints
194     (world-state condition (type ?node) = node)
195     (world-state condition (manualSearch ?node) = true)
196     (world-state condition (type ?location) = location)
197     (world-state condition (latitude ?location) = ?latitude)
198     (world-state condition (longitude ?location) = ?longitude)
199     (world-state condition (latitude ?node) = ?latitude)
200     (world-state condition (longitude ?node) = ?longitude))
201   (annotations
202     (comments = "")))
203
204 ;; report results of a location test to an authority
205 (refinement reportResults (reportResults ?node ?location)
206   (variables ?node ?location)
207   (constraints
208     (world-state condition (resultReport ?node) = true)
209     (world-state condition (type ?node) = node)
210     (world-state condition (type ?location) = location))
211   (annotations
212     (comments = "Results Reported")))
213
214 (annotations
215   (comments = ""))
```

```
1 <plan xmlns="http://www.aiai.ed.ac.uk/project/ix/">
2   <plan-nodes>
3     <list>
4       <plan-node id="node-1">
5         <activity>
```

```
6                <activity>
7                    <pattern>
8                        <list>
9                            <symbol>sweepForIEDs</symbol>
10                       </list>
11                   </pattern>
12               </activity>
13           </activity>
14       </plan-node>
15   </list>
16 </plan-nodes>
17 <world-state>
18   <list>
19       <!-- Network Nodes -->
20       <!-- Network Node 1-->
21       <pattern-assignment>
22           <pattern>
23               <list>
24                   <symbol>type</symbol>
25                   <symbol>node1</symbol>
26               </list>
27           </pattern>
28           <value>
29               <symbol>node</symbol>
30           </value>
31       </pattern-assignment>
32       <pattern-assignment>
33           <pattern>
34               <list>
35                   <symbol>latitude</symbol>
36                   <symbol>node1</symbol>
37               </list>
```

```
38              </pattern>
39              <value>
40                  <symbol>39.030445</symbol>
41              </value>
42          </pattern-assignment>
43          <pattern-assignment>
44              <pattern>
45                  <list>
46                      <symbol>longitude</symbol>
47                      <symbol>node1</symbol>
48                  </list>
49              </pattern>
50              <value>
51                  <symbol>-74.685936</symbol>
52              </value>
53          </pattern-assignment>
54          <!-- Network Node 2-->
55          <pattern-assignment>
56              <pattern>
57                  <list>
58                      <symbol>type</symbol>
59                      <symbol>node2</symbol>
60                  </list>
61              </pattern>
62              <value>
63                  <symbol>node</symbol>
64              </value>
65          </pattern-assignment>
66          <pattern-assignment>
67              <pattern>
68                  <list>
69                      <symbol>latitude</symbol>
```

```
70                  <symbol>node2</symbol>
71              </list>
72          </pattern>
73          <value>
74              <symbol>39.324776</symbol>
75          </value>
76      </pattern-assignment>
77      <pattern-assignment>
78          <pattern>
79              <list>
80                  <symbol>longitude</symbol>
81                  <symbol>node2</symbol>
82              </list>
83          </pattern>
84          <value>
85              <symbol>-74.404686</symbol>
86          </value>
87      </pattern-assignment>
88      <!-- Network Node 3-->
89      <pattern-assignment>
90          <pattern>
91              <list>
92                  <symbol>type</symbol>
93                  <symbol>node3</symbol>
94              </list>
95          </pattern>
96          <value>
97              <symbol>node</symbol>
98          </value>
99      </pattern-assignment>
100     <pattern-assignment>
101         <pattern>
```

```
102                 <list>
103                     <symbol>latitude</symbol>
104                     <symbol>node3</symbol>
105                 </list>
106             </pattern>
107             <value>
108                 <symbol>39.639534</symbol>
109             </value>
110         </pattern-assignment>
111         <pattern-assignment>
112             <pattern>
113                 <list>
114                     <symbol>longitude</symbol>
115                     <symbol>node3</symbol>
116                 </list>
117             </pattern>
118             <value>
119                 <symbol>-76.45781</symbol>
120             </value>
121         </pattern-assignment>
122         <!-- Network Node 4-->
123         <pattern-assignment>
124             <pattern>
125                 <list>
126                     <symbol>type</symbol>
127                     <symbol>node4</symbol>
128                 </list>
129             </pattern>
130             <value>
131                 <symbol>node</symbol>
132             </value>
133         </pattern-assignment>
```

```
134         <pattern-assignment>
135             <pattern>
136                 <list>
137                     <symbol>latitude</symbol>
138                     <symbol>node4</symbol>
139                 </list>
140             </pattern>
141             <value>
142                 <symbol>39.790974</symbol>
143             </value>
144         </pattern-assignment>
145         <pattern-assignment>
146             <pattern>
147                 <list>
148                     <symbol>longitude</symbol>
149                     <symbol>node4</symbol>
150                 </list>
151             </pattern>
152             <value>
153                 <symbol>-76.09781</symbol>
154             </value>
155         </pattern-assignment>
156         <!-- Network Node 5-->
157         <pattern-assignment>
158             <pattern>
159                 <list>
160                     <symbol>type</symbol>
161                     <symbol>node5</symbol>
162                 </list>
163             </pattern>
164             <value>
165                 <symbol>node</symbol>
```

```
166             </value>
167         </pattern-assignment>
168         <pattern-assignment>
169             <pattern>
170                 <list>
171                     <symbol>latitude</symbol>
172                     <symbol>node5</symbol>
173                 </list>
174             </pattern>
175             <value>
176                 <symbol>40.114372</symbol>
177             </value>
178         </pattern-assignment>
179         <pattern-assignment>
180             <pattern>
181                 <list>
182                     <symbol>longitude</symbol>
183                     <symbol>node5</symbol>
184                 </list>
185             </pattern>
186             <value>
187                 <symbol>-75.12187</symbol>
188             </value>
189         </pattern-assignment>
190         <!-- Node / Service Mapping -->
191         <pattern-assignment>
192             <pattern>
193                 <list>
194                     <symbol>checkForIEDAt</symbol>
195                     <symbol>node1</symbol>
196                 </list>
197             </pattern>
```

```
198             <value>
199                 <symbol>true</symbol>
200             </value>
201         </pattern-assignment>
202         <pattern-assignment>
203             <pattern>
204                 <list>
205                     <symbol>checkForIEDAt</symbol>
206                     <symbol>node2</symbol>
207                 </list>
208             </pattern>
209             <value>
210                 <symbol>true</symbol>
211             </value>
212         </pattern-assignment>
213         <pattern-assignment>
214             <pattern>
215                 <list>
216                     <symbol>checkForIEDAt</symbol>
217                     <symbol>node5</symbol>
218                 </list>
219             </pattern>
220             <value>
221                 <symbol>true</symbol>
222             </value>
223         </pattern-assignment>
224         <pattern-assignment>
225             <pattern>
226                 <list>
227                     <symbol>manualSearch</symbol>
228                     <symbol>node1</symbol>
229                 </list>
```

```
230             </pattern>
231             <value>
232                 <symbol>true</symbol>
233             </value>
234         </pattern-assignment>
235         <pattern-assignment>
236             <pattern>
237                 <list>
238                     <symbol>manualSearch</symbol>
239                     <symbol>node2</symbol>
240                 </list>
241             </pattern>
242             <value>
243                 <symbol>true</symbol>
244             </value>
245         </pattern-assignment>
246         <pattern-assignment>
247             <pattern>
248                 <list>
249                     <symbol>manualSearch</symbol>
250                     <symbol>node3</symbol>
251                 </list>
252             </pattern>
253             <value>
254                 <symbol>true</symbol>
255             </value>
256         </pattern-assignment>
257         <pattern-assignment>
258             <pattern>
259                 <list>
260                     <symbol>manualSearch</symbol>
261                     <symbol>node4</symbol>
```

```
262                 </list>
263             </pattern>
264             <value>
265                 <symbol>true</symbol>
266             </value>
267         </pattern-assignment>
268         <pattern-assignment>
269             <pattern>
270                 <list>
271                     <symbol>photographicSearch</symbol>
272                     <symbol>node3</symbol>
273                 </list>
274             </pattern>
275             <value>
276                 <symbol>true</symbol>
277             </value>
278         </pattern-assignment>
279         <pattern-assignment>
280             <pattern>
281                 <list>
282                     <symbol>photographicSearch</symbol>
283                     <symbol>node4</symbol>
284                 </list>
285             </pattern>
286             <value>
287                 <symbol>true</symbol>
288             </value>
289         </pattern-assignment>
290         <pattern-assignment>
291             <pattern>
292                 <list>
293                     <symbol>photographicSearch</symbol>
```

```
294                     <symbol>node5</symbol>
295                 </list>
296             </pattern>
297             <value>
298                 <symbol>true</symbol>
299             </value>
300         </pattern-assignment>
301         <pattern-assignment>
302             <pattern>
303                 <list>
304                     <symbol>photoArchive</symbol>
305                     <symbol>node5</symbol>
306                 </list>
307             </pattern>
308             <value>
309                 <symbol>true</symbol>
310             </value>
311         </pattern-assignment>
312         <pattern-assignment>
313             <pattern>
314                 <list>
315                     <symbol>photoCompare</symbol>
316                     <symbol>node4</symbol>
317                 </list>
318             </pattern>
319             <value>
320                 <symbol>true</symbol>
321             </value>
322         </pattern-assignment>
323         <pattern-assignment>
324             <pattern>
325                 <list>
```

```
326                 <symbol>photoCompare</symbol>
327                 <symbol>node5</symbol>
328             </list>
329         </pattern>
330         <value>
331             <symbol>true</symbol>
332         </value>
333     </pattern-assignment>
334     <pattern-assignment>
335         <pattern>
336             <list>
337                 <symbol>resultReport</symbol>
338                 <symbol>node2</symbol>
339             </list>
340         </pattern>
341         <value>
342             <symbol>true</symbol>
343         </value>
344     </pattern-assignment>
345     <pattern-assignment>
346         <pattern>
347             <list>
348                 <symbol>resultReport</symbol>
349                 <symbol>node5</symbol>
350             </list>
351         </pattern>
352         <value>
353             <symbol>true</symbol>
354         </value>
355     </pattern-assignment>
356     <!-- Location 1 -->
357     <pattern-assignment>
```

```
358             <pattern>
359                 <list>
360                     <symbol>type</symbol>
361                     <symbol>location1</symbol>
362                 </list>
363             </pattern>
364             <value>
365                 <symbol>location</symbol>
366             </value>
367         </pattern-assignment>
368         <pattern-assignment>
369             <pattern>
370                 <list>
371                     <symbol>checked</symbol>
372                     <symbol>location1</symbol>
373                 </list>
374             </pattern>
375             <value>
376                 <symbol>false</symbol>
377             </value>
378         </pattern-assignment>
379         <pattern-assignment>
380             <pattern>
381                 <list>
382                     <symbol>searched</symbol>
383                     <symbol>location1</symbol>
384                 </list>
385             </pattern>
386             <value>
387                 <symbol>false</symbol>
388             </value>
389         </pattern-assignment>
```

```
390             <pattern-assignment>
391                 <pattern>
392                     <list>
393                         <symbol>latitude</symbol>
394                         <symbol>location1</symbol>
395                     </list>
396                 </pattern>
397                 <value>
398                     <symbol>39.504946</symbol>
399                 </value>
400             </pattern-assignment>
401             <pattern-assignment>
402                 <pattern>
403                     <list>
404                         <symbol>longitude</symbol>
405                         <symbol>location1</symbol>
406                     </list>
407                 </pattern>
408                 <value>
409                     <symbol>-75.12187</symbol>
410                 </value>
411             </pattern-assignment>
412             <!-- Location 2 -->
413             <pattern-assignment>
414                 <pattern>
415                     <list>
416                         <symbol>type</symbol>
417                         <symbol>location2</symbol>
418                     </list>
419                 </pattern>
420                 <value>
421                     <symbol>location</symbol>
```

```
422             </value>
423         </pattern-assignment>
424         <pattern-assignment>
425             <pattern>
426                 <list>
427                     <symbol>checked</symbol>
428                     <symbol>location2</symbol>
429                 </list>
430             </pattern>
431             <value>
432                 <symbol>false</symbol>
433             </value>
434         </pattern-assignment>
435         <pattern-assignment>
436             <pattern>
437                 <list>
438                     <symbol>searched</symbol>
439                     <symbol>location2</symbol>
440                 </list>
441             </pattern>
442             <value>
443                 <symbol>false</symbol>
444             </value>
445         </pattern-assignment>
446         <pattern-assignment>
447             <pattern>
448                 <list>
449                     <symbol>latitude</symbol>
450                     <symbol>location2</symbol>
451                 </list>
452             </pattern>
453             <value>
```

```
454                 <symbol>39.1396</symbol>
455             </value>
456         </pattern-assignment>
457         <pattern-assignment>
458             <pattern>
459                 <list>
460                     <symbol>longitude</symbol>
461                     <symbol>location2</symbol>
462                 </list>
463             </pattern>
464             <value>
465                 <symbol>-76.3875</symbol>
466             </value>
467         </pattern-assignment>
468         <!-- Location 3 -->
469         <pattern-assignment>
470             <pattern>
471                 <list>
472                     <symbol>type</symbol>
473                     <symbol>location3</symbol>
474                 </list>
475             </pattern>
476             <value>
477                 <symbol>location</symbol>
478             </value>
479         </pattern-assignment>
480         <pattern-assignment>
481             <pattern>
482                 <list>
483                     <symbol>checked</symbol>
484                     <symbol>location3</symbol>
485                 </list>
```

```
486             </pattern>
487             <value>
488                 <symbol>false</symbol>
489             </value>
490         </pattern-assignment>
491         <pattern-assignment>
492             <pattern>
493                 <list>
494                     <symbol>searched</symbol>
495                     <symbol>location3</symbol>
496                 </list>
497             </pattern>
498             <value>
499                 <symbol>false</symbol>
500             </value>
501         </pattern-assignment>
502         <pattern-assignment>
503             <pattern>
504                 <list>
505                     <symbol>latitude</symbol>
506                     <symbol>location3</symbol>
507                 </list>
508             </pattern>
509             <value>
510                 <symbol>39.995968</symbol>
511             </value>
512         </pattern-assignment>
513         <pattern-assignment>
514             <pattern>
515                 <list>
516                     <symbol>longitude</symbol>
517                     <symbol>location3</symbol>
```

```
518                 </list>
519             </pattern>
520             <value>
521                 <symbol>-75.993744</symbol>
522             </value>
523         </pattern-assignment>
524         <!-- Location 4 -->
525         <pattern-assignment>
526             <pattern>
527                 <list>
528                     <symbol>type</symbol>
529                     <symbol>location4</symbol>
530                 </list>
531             </pattern>
532             <value>
533                 <symbol>location</symbol>
534             </value>
535         </pattern-assignment>
536         <pattern-assignment>
537             <pattern>
538                 <list>
539                     <symbol>checked</symbol>
540                     <symbol>location4</symbol>
541                 </list>
542             </pattern>
543             <value>
544                 <symbol>false</symbol>
545             </value>
546         </pattern-assignment>
547         <pattern-assignment>
548             <pattern>
549                 <list>
```

```
550                 <symbol>searched</symbol>
551                 <symbol>location4</symbol>
552             </list>
553         </pattern>
554         <value>
555             <symbol>false</symbol>
556         </value>
557     </pattern-assignment>
558     <pattern-assignment>
559         <pattern>
560             <list>
561                 <symbol>latitude</symbol>
562                 <symbol>location4</symbol>
563             </list>
564         </pattern>
565         <value>
566             <symbol>40.264767</symbol>
567         </value>
568     </pattern-assignment>
569     <pattern-assignment>
570         <pattern>
571             <list>
572                 <symbol>longitude</symbol>
573                 <symbol>location4</symbol>
574             </list>
575         </pattern>
576         <value>
577             <symbol>-74.475</symbol>
578         </value>
579     </pattern-assignment>
580     <!-- Camera 1 -->
581     <pattern-assignment>
```

```
582             <pattern>
583                 <list>
584                     <symbol>type</symbol>
585                     <symbol>camera1</symbol>
586                 </list>
587             </pattern>
588             <value>
589                 <symbol>camera</symbol>
590             </value>
591         </pattern-assignment>
592         <pattern-assignment>
593             <pattern>
594                 <list>
595                     <symbol>latitude</symbol>
596                     <symbol>camera1</symbol>
597                 </list>
598             </pattern>
599             <value>
600                 <symbol>38.844494</symbol>
601             </value>
602         </pattern-assignment>
603         <pattern-assignment>
604             <pattern>
605                 <list>
606                     <symbol>longitude</symbol>
607                     <symbol>camera1</symbol>
608                 </list>
609             </pattern>
610             <value>
611                 <symbol>-75.36093</symbol>
612             </value>
613         </pattern-assignment>
```

```
614             <pattern-assignment>
615                 <pattern>
616                     <list>
617                         <symbol>inUse</symbol>
618                         <symbol>camera1</symbol>
619                     </list>
620                 </pattern>
621                 <value>
622                     <symbol>false</symbol>
623                 </value>
624             </pattern-assignment>
625             <!-- Camera 2 -->
626             <pattern-assignment>
627                 <pattern>
628                     <list>
629                         <symbol>type</symbol>
630                         <symbol>camera2</symbol>
631                     </list>
632                 </pattern>
633                 <value>
634                     <symbol>camera</symbol>
635                 </value>
636             </pattern-assignment>
637             <pattern-assignment>
638                 <pattern>
639                     <list>
640                         <symbol>latitude</symbol>
641                         <symbol>camera2</symbol>
642                     </list>
643                 </pattern>
644                 <value>
645                     <symbol>38.504158</symbol>
```

```
646             </value>
647         </pattern-assignment>
648         <pattern-assignment>
649             <pattern>
650                 <list>
651                     <symbol>longitude</symbol>
652                     <symbol>camera2</symbol>
653                 </list>
654             </pattern>
655             <value>
656                 <symbol>-75.78281</symbol>
657             </value>
658         </pattern-assignment>
659         <pattern-assignment>
660             <pattern>
661                 <list>
662                     <symbol>inUse</symbol>
663                     <symbol>camera2</symbol>
664                 </list>
665             </pattern>
666             <value>
667                 <symbol>false</symbol>
668             </value>
669         </pattern-assignment>
670         <!-- Camera 3 -->
671         <!--
672         <pattern-assignment>
673             <pattern>
674                 <list>
675                     <symbol>type</symbol>
676                     <symbol>camera3</symbol>
677                 </list>
```

```
678             </pattern>
679             <value>
680                 <symbol>camera</symbol>
681             </value>
682         </pattern-assignment>
683         <pattern-assignment>
684             <pattern>
685                 <list>
686                     <symbol>latitude</symbol>
687                     <symbol>camera3</symbol>
688                 </list>
689             </pattern>
690             <value>
691                 <symbol>38.0221</symbol>
692             </value>
693         </pattern-assignment>
694         <pattern-assignment>
695             <pattern>
696                 <list>
697                     <symbol>longitude</symbol>
698                     <symbol>camera3</symbol>
699                 </list>
700             </pattern>
701             <value>
702                 <symbol>-76.8882</symbol>
703             </value>
704         </pattern-assignment>
705         <pattern-assignment>
706             <pattern>
707                 <list>
708                     <symbol>inUse</symbol>
709                     <symbol>camera3</symbol>
```

```
                </list>
            </pattern>
            <value>
                <symbol>false</symbol>
            </value>
        </pattern-assignment>
        -->
    </list>
  </world-state>
</plan>
```

```
<?xml version="1.0" encoding="UTF-8"?>

<network-cost>
  <from-node from="node1">
    <transportation>0.00006</transportation>
    <speed>30</speed>
    <to-node to="node1">
      <cost>0.1</cost>
      <hops>0</hops>
    </to-node>
    <to-node to="node2">
      <cost>0.2</cost>
      <hops>1</hops>
    </to-node>
    <to-node to="node3">
      <cost>0.6</cost>
      <hops>2</hops>
    </to-node>
    <to-node to="node4">
      <cost>0.7</cost>
      <hops>2</hops>
```

```
22          </to-node>
23          <to-node to="node5">
24              <cost>0.4</cost>
25              <hops>1</hops>
26          </to-node>
27          <to-node to="node6">
28              <cost>1.0</cost>
29              <hops>3</hops>
30          </to-node>
31      </from-node>
32      <from-node from="node2">
33          <transportation>0.000065</transportation>
34          <speed>40</speed>
35          <to-node to="node1">
36              <cost>0.2</cost>
37              <hops>1</hops>
38          </to-node>
39          <to-node to="node2">
40              <cost>0.1</cost>
41              <hops>0</hops>
42          </to-node>
43          <to-node to="node3">
44              <cost>0.7</cost>
45              <hops>2</hops>
46          </to-node>
47          <to-node to="node4">
48              <cost>0.8</cost>
49              <hops>2</hops>
50          </to-node>
51          <to-node to="node5">
52              <cost>0.4</cost>
53              <hops>1</hops>
```

```
54          </to-node>
55          <to-node to="node6">
56              <cost>1.0</cost>
57              <hops>3</hops>
58          </to-node>
59      </from-node>
60      <from-node from="node3">
61          <transportation>0.000051</transportation>
62          <speed>20</speed>
63          <to-node to="node1">
64              <cost>0.7</cost>
65              <hops>2</hops>
66          </to-node>
67          <to-node to="node2">
68              <cost>0.7</cost>
69              <hops>2</hops>
70          </to-node>
71          <to-node to="node3">
72              <cost>0.1</cost>
73              <hops>0</hops>
74          </to-node>
75          <to-node to="node4">
76              <cost>0.2</cost>
77              <hops>1</hops>
78          </to-node>
79          <to-node to="node5">
80              <cost>0.5</cost>
81              <hops>1</hops>
82          </to-node>
83          <to-node to="node6">
84              <cost>1.0</cost>
85              <hops>3</hops>
```

```
86          </to-node>
87      </from-node>
88      <from-node from="node4">
89          <transportation>0.000049</transportation>
90          <speed>10</speed>
91          <to-node to="node1">
92              <cost>0.7</cost>
93              <hops>2</hops>
94          </to-node>
95          <to-node to="node2">
96              <cost>0.8</cost>
97              <hops>2</hops>
98          </to-node>
99          <to-node to="node3">
100             <cost>0.2</cost>
101             <hops>1</hops>
102         </to-node>
103         <to-node to="node4">
104             <cost>0.1</cost>
105             <hops>0</hops>
106         </to-node>
107         <to-node to="node5">
108             <cost>0.4</cost>
109             <hops>1</hops>
110         </to-node>
111         <to-node to="node6">
112             <cost>1.0</cost>
113             <hops>3</hops>
114         </to-node>
115     </from-node>
116     <from-node from="node5">
117         <transportation>0.000062</transportation>
```

```
118         <speed>45</speed>
119         <to-node to="node1">
120             <cost>0.4</cost>
121             <hops>1</hops>
122         </to-node>
123         <to-node to="node2">
124             <cost>0.5</cost>
125             <hops>1</hops>
126         </to-node>
127         <to-node to="node3">
128             <cost>0.4</cost>
129             <hops>1</hops>
130         </to-node>
131         <to-node to="node4">
132             <cost>0.5</cost>
133             <hops>1</hops>
134         </to-node>
135         <to-node to="node5">
136             <cost>0.1</cost>
137             <hops>0</hops>
138         </to-node>
139         <to-node to="node6">
140             <cost>1.0</cost>
141             <hops>3</hops>
142         </to-node>
143     </from-node>
144     <from-node from="node6">
145         <transportation>0.000053</transportation>
146         <speed>15</speed>
147         <to-node to="node1">
148             <cost>1.0</cost>
149             <hops>3</hops>
```

```
150         </to-node>
151         <to-node to="node2">
152             <cost>1.0</cost>
153             <hops>3</hops>
154         </to-node>
155         <to-node to="node3">
156             <cost>1.0</cost>
157             <hops>3</hops>
158         </to-node>
159         <to-node to="node4">
160             <cost>1.0</cost>
161             <hops>3</hops>
162         </to-node>
163         <to-node to="node5">
164             <cost>1.0</cost>
165             <hops>3</hops>
166         </to-node>
167         <to-node to="node6">
168             <cost>1.0</cost>
169             <hops>3</hops>
170         </to-node>
171     </from-node>
172     <from-node from="camera1">
173         <resolution>3.2</resolution>
174     </from-node>
175     <from-node from="camera2">
176         <resolution>8.0</resolution>
177     </from-node>
178     <from-node from="camera3">
179         <resolution>10.0</resolution>
180     </from-node>
181 </network-cost>
```