Network-Aware Automated Planning

and Plan Execution

Kyle Usbeck

A Thesis Submitted to the Faculty of Drexel University in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

2009-07-07

Department of Computer Science

Drexel UNIVERSITY

Outline

1. Introduction
   - Motivation
   - Background
   - Approach
2. Formalization
   - Problem Statement
3. Technical Approach
   - Planning Agents
   - Execution Agents
   - Monitoring Agents
   - Mixed-initiative UI
4. Experiments
   - Plan Evaluation Benchmarking
   - Network-Aware Agent Combinations
   - Discussion

Department of Computer Science

Drexel UNIVERSITY

Motivation

April 2009:

- 75% of coalition force casualties in Afghanistan are from roadside bombs.
- 40% of coalition force casualties in Iraq are from roadside bombs.

Source: Tom Vanden Brook, USA Today

Motivation

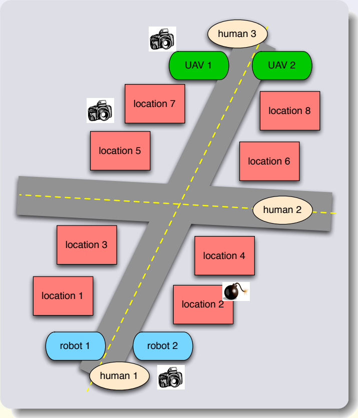
April 2009:

- 75% of coalition force casualties in Afghanistan are from roadside bombs.
- 40% of coalition force casualties in Iraq are from roadside bombs.

Source: Tom Vanden Brook, USA Today

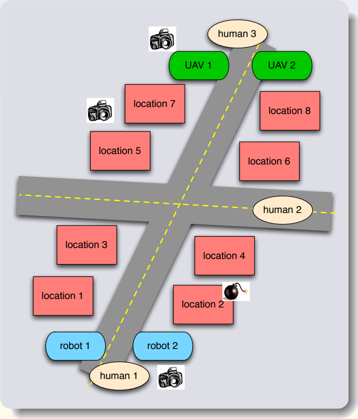
Motivating Scenario

- IED Detection.
- Monitor Locations.
- Techniques.
- Actors.
- Resources.
- Evaluators.



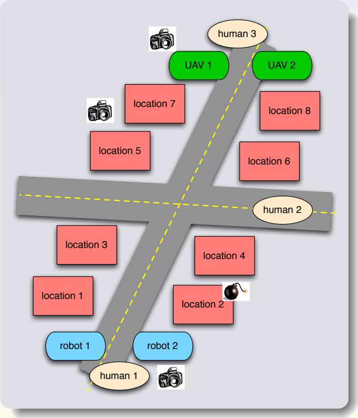
Motivating Scenario

- IED Detection.
- Monitor Locations.
- Techniques.
- Actors.
- Resources.
- Evaluators.



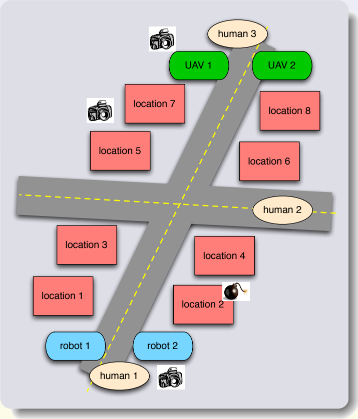
Motivating Scenario

- IED Detection.
- Monitor Locations.
- Techniques.
- Actors.
- Resources.
- Evaluators.



Motivating Scenario

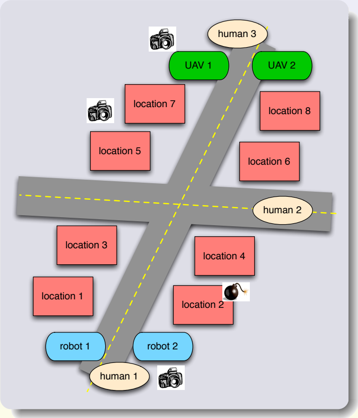
- IED Detection.
- Monitor Locations.
- Techniques.
- Actors.
- Resources.
- Evaluators.



Department of Computer Science

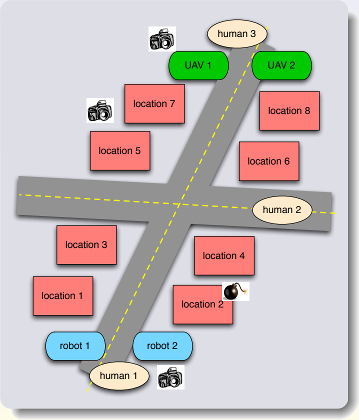
Motivating Scenario

- IED Detection.
- Monitor Locations.
- Techniques.
- Actors.
- Resources.
- Evaluators.



Motivating Scenario

- IED Detection.
- Monitor Locations.
- Techniques.
- Actors.
- Resources.
- Evaluators.



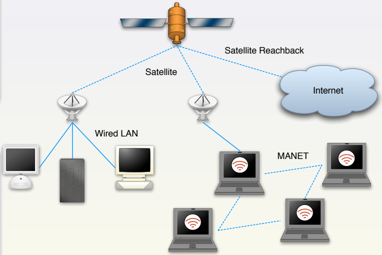
Motivating Scenario

Heterogeneous Network

multiple different network technologies are combined to work together simultaneously.

Network-Centric System

a distributed system where performance is dependent on the quality of the underlying network communication links.



Outline

1. Introduction
   - Motivation
   - Background
   - Approach
2. Formalization
   - Problem Statement
3. Technical Approach
   - Planning Agents
   - Execution Agents
   - Monitoring Agents
   - Mixed-initiative UI
4. Experiments
   - Plan Evaluation Benchmarking
   - Network-Aware Agent Combinations
   - Discussion

Department of
Computer Science

Drexel
UNIVERSITY

Contributions

1. Qualitatively-different plans:
    - Generating plans over a range of evaluation criteria;
    - Visualizing plan evaluations.
    - Improve plan selection.
2. Network-Aware Agents:
    - Classical planning domains for distributed service composition;
    - Measuring the performance and effectiveness of planning, execution, and monitoring agents;
    - Incorporating network-awareness.

Contributions

1. Qualitatively-different plans:
   - Generating plans over a range of evaluation criteria;
   - Visualizing plan evaluations.
   - Improve plan selection.
2. Network-Aware Agents:
   - Classical planning domains for distributed service composition;
   - Measuring the performance and effectiveness of planning, execution, and monitoring agents;
   - Incorporating network-awareness.

Contributions

1. Qualitatively-different plans:
    - Generating plans over a range of evaluation criteria;
    - Visualizing plan evaluations.
    - Improve plan selection.
2. Network-Aware Agents:
    - Classical planning domains for distributed service composition;
    - Measuring the performance and effectiveness of planning, execution, and monitoring agents;
    - Incorporating network-awareness.

Contributions

1. Qualitatively-different plans:
   - Generating plans over a range of evaluation criteria;
   - Visualizing plan evaluations.
   - Improve plan selection.
2. Network-Aware Agents:
   - Classical planning domains for distributed service composition;
   - Measuring the performance and effectiveness of planning, execution, and monitoring agents;
   - Incorporating network-awareness.

Contributions

1. Qualitatively-different plans:
    - Generating plans over a range of evaluation criteria;
    - Visualizing plan evaluations.
    - Improve plan selection.
2. Network-Aware Agents:
    - Classical planning domains for distributed service composition;
    - Measuring the performance and effectiveness of planning, execution, and monitoring agents;
    - Incorporating network-awareness.

Department of Computer Science

Drexel UNIVERSITY

Contributions

1. Qualitatively-different plans:
   - Generating plans over a range of evaluation criteria;
   - Visualizing plan evaluations.
   - Improve plan selection.
2. Network-Aware Agents:
   - Classical planning domains for distributed service composition;
   - Measuring the performance and effectiveness of planning, execution, and monitoring agents;
   - Incorporating network-awareness.

Contributions

1. Qualitatively-different plans:
   - Generating plans over a range of evaluation criteria;
   - Visualizing plan evaluations.
   - Improve plan selection.
2. Network-Aware Agents:
   - Classical planning domains for distributed service composition;
   - Measuring the performance and effectiveness of planning, execution, and monitoring agents;
   - Incorporating network-awareness.

Contributions

1. Qualitatively-different plans:
   - Generating plans over a range of evaluation criteria;
   - Visualizing plan evaluations.
   - Improve plan selection.
2. Network-Aware Agents:
   - Classical planning domains for distributed service composition;
   - Measuring the performance and effectiveness of planning, execution, and monitoring agents;
   - Incorporating network-awareness.

Outline

1. Introduction
   - Motivation
   - Background
   - Approach
2. Formalization
   - Problem Statement
3. Technical Approach
   - Planning Agents
   - Execution Agents
   - Monitoring Agents
   - Mixed-initiative UI
4. Experiments
   - Plan Evaluation Benchmarking
   - Network-Aware Agent Combinations
   - Discussion

Service Composition to Automated Planning

Definition

"Service composition is the linking. . . of existing services so that their aggregate behavior is that of a desired service (the goal)" [Hoffmann *et al.* 09].

- Requires Semantic Web Services [Sirin *et al.* 04].
- QoS Assurance [Gu *et al.* 03].

Assumes static networking.

Service Composition to Automated Planning

Definition

“Service composition is the linking. . . of existing services so that their aggregate behavior is that of a desired service (the goal)” [Hoffmann *et al.* 09].

- Requires Semantic Web Services [Sirin *et al.* 04].
- QoS Assurance [Gu *et al.* 03].

Assumes static networking.

Service Composition to Automated Planning

Definition

“Service composition is the linking. . . of existing services so that their aggregate behavior is that of a desired service (the goal)” [Hoffmann *et al.* 09].

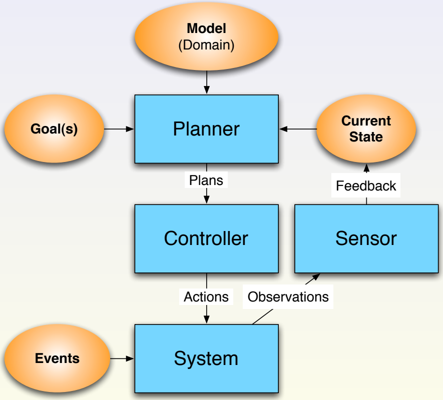
- Requires Semantic Web Services [Sirin *et al.* 04].
- QoS Assurance [Gu *et al.* 03].

Assumes static networking.

Department of Computer Science

Drexel UNIVERSITY

Agents in Planning

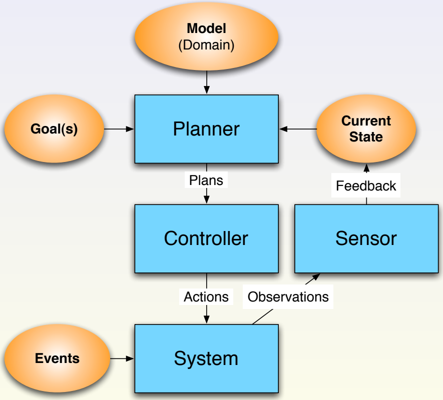


Agents:

- Planning Agent.
- Execution Agent.
- Monitoring Agent.

[Tate 93]

Agents in Planning

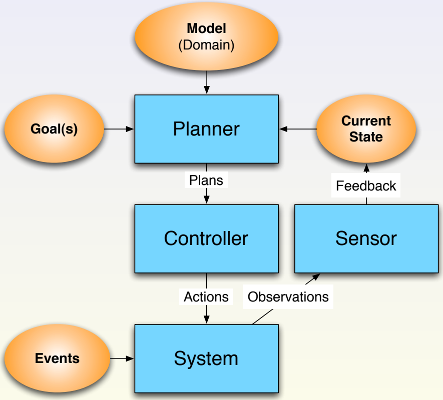


Agents:

- Planning Agent.
- Execution Agent.
- Monitoring Agent.

[Tate 93]

Agents in Planning



Agents:

- Planning Agent.
- Execution Agent.
- Monitoring Agent.

[Tate 93]

Planning Under Uncertainty

Restrictive Assumptions:

- Determinism.
- Full observability.
- Reachability goals.

[Nau *et al.* 04]

Sources of Uncertainty:

- Partial observability.
- Unreliable resources.
- Measurement variance.
- Inherently vague concepts.

Planning Under Uncertainty

Restrictive Assumptions:

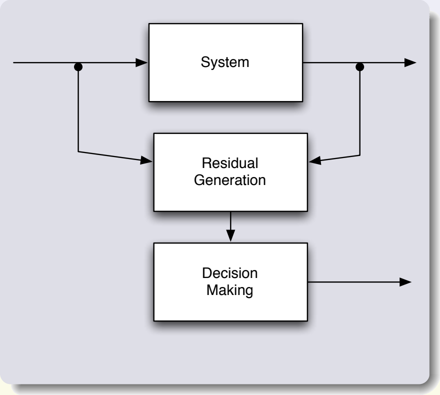
- Determinism.
- Full observability.
- Reachability goals.

[Nau *et al.* 04]

Sources of Uncertainty:

- Partial observability.
- Unreliable resources.
- Measurement variance.
- Inherently vague concepts.

Fault Detection & Isolation (FDI)

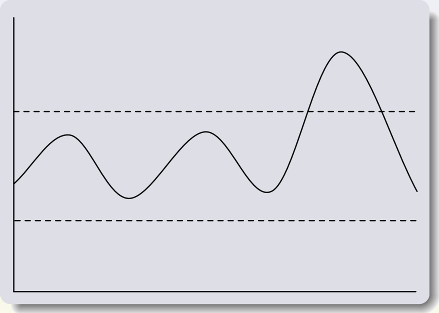


Types of FDI:

- Analytic.
- Data-driven.
- Knowledge-based.

[Pettersson 05]

Fault Detection & Isolation (FDI)



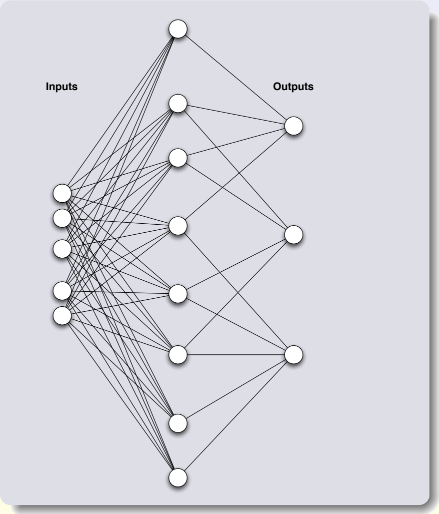
Types of FDI:

- Analytic.
- Data-driven.
- Knowledge-based.

[Pettersson 05]

Department of Computer Science
Drexel UNIVERSITY

Fault Detection & Isolation (FDI)



Types of FDI:

- Analytic.
- Data-driven.
- Knowledge-based.

[Pettersson 05]

Department of Computer Science

Outline

1 Introduction
- Motivation
- Background
- Approach

2 Formalization
- Problem Statement

3 Technical Approach
- Planning Agents
- Execution Agents
- Monitoring Agents
- Mixed-initiative UI

4 Experiments
- Plan Evaluation Benchmarking
- Network-Aware Agent Combinations
- Discussion

Approach

1. Modify planner to improve the quality of the plans it produces based on evaluation criteria.
2. Add network-awareness to planning, execution, and monitoring agents.

Purpose

To improve network-centric automated planning and execution.

Department of Computer Science

Drexel UNIVERSITY

Approach

1. Modify planner to improve the quality of the plans it produces based on evaluation criteria.
2. Add network-awareness to planning, execution, and monitoring agents.

Purpose

To improve network-centric automated planning and execution.

Department of Computer Science

Approach

1. Modify planner to improve the quality of the plans it produces based on evaluation criteria.
2. Add network-awareness to planning, execution, and monitoring agents.

Purpose

To improve network-centric automated planning and execution.

Department of Computer Science

Drexel UNIVERSITY

Outline

1. Introduction
   - Motivation
   - Background
   - Approach
2. Formalization
   - Problem Statement
3. Technical Approach
   - Planning Agents
   - Execution Agents
   - Monitoring Agents
   - Mixed-initiative UI
4. Experiments
   - Plan Evaluation Benchmarking
   - Network-Aware Agent Combinations
   - Discussion

Formal Problem Statement

Σ is the planning domain — the model of the world passed as input to the planner.

Σ is a Tuple

S set of states;

A set of actions;

E set of events;

γ transition function $\gamma : S \times A \to S$.

Formal Problem Statement

Σ is the planning domain — the model of the world passed as input to the planner.

Σ is a Tuple

S set of states;

A set of actions;

E set of events;

γ transition function $\gamma : S \times A \to S$.

Department of Computer Science

Formal Problem Statement

Σ is the planning domain — the model of the world passed as input to the planner.

Σ is a Tuple

S set of states;

A set of actions;

E set of events;

γ transition function $\gamma : S \times A \rightarrow S$.

Department of Computer Science

Drexel UNIVERSITY

Formal Problem Statement

Σ is the planning domain — the model of the world passed as input to the planner.

Σ is a Tuple

S set of states;

A set of actions;

E set of events;

γ transition function $\gamma : S \times A \to S$.

Formal Problem Statement

The functions on planning actions:

For $a \in A$

precond(a) preconditions of a;

effects$^{+}(a)$ positive effects of a;

effects$^{-}(a)$ negative effects of a;

host(a) the single host h from a;

resources(a) the set of resources (parameters) of action a.

Department of Computer Science

Formal Problem Statement

The functions on planning actions:

For $a \in A$

$\text{precond}(a)$ preconditions of a;

$\text{effects}^{+}(a)$ positive effects of a;

$\text{effects}^{-}(a)$ negative effects of a;

$\text{host}(a)$ the single host h from a;

$\text{resources}(a)$ the set of resources (parameters) of action a.

Department of Computer Science

Drexel UNIVERSITY

Formal Problem Statement

The functions on planning actions:

For $a \in A$

$\text{precond}(a)$	preconditions of a;
$\text{effects}^{+}(a)$	positive effects of a;
$\text{effects}^{-}(a)$	negative effects of a;
$\text{host}(a)$	the single host h from a;
$\text{resources}(a)$	the set of resources (parameters) of action a.

Formal Problem Statement

The functions on planning actions:

For $a \in A$

$\text{precond}(a)$ preconditions of a;

$\text{effects}^{+}(a)$ positive effects of a;

$\text{effects}^{-}(a)$ negative effects of a;

$\text{host}(a)$ the single host h from a;

$\text{resources}(a)$ the set of resources (parameters) of action a.

Department of Computer Science

Formal Problem Statement

The functions on planning actions:

For $a \in A$

precond(a)	preconditions of a;
effects$^+(a)$	positive effects of a;
effects$^-(a)$	negative effects of a;
host(a)	the single host h from a;
resources(a)	the set of resources (parameters) of action a.

Formal Problem Statement

The planning agent receives the tuple, I_P, and creates a set of plans, P_I.

I_P is a Tuple

Σ	automated planning domain;
s_0	initial state;
S_g	set of goal state(s);
H	set of hosts (nodes) on the network;
ω_H	host link weighting.

Formal Problem Statement

The planning agent receives the tuple, I_P, and creates a set of plans, P_I.

I_P is a Tuple

Σ automated planning domain;

s_0 initial state;

S_g set of goal state(s);

H set of hosts (nodes) on the network;

ω_H host link weighting.

Formal Problem Statement

The planning agent receives the tuple, I_P, and creates a set of plans, P_I.

I_P is a Tuple

Σ	automated planning domain;
s_0	initial state;
S_g	set of goal state(s);
H	set of hosts (nodes) on the network;
ω_H	host link weighting.

Formal Problem Statement

The planning agent receives the tuple, I_P, and creates a set of plans, P_I.

I_P is a Tuple

Σ automated planning domain;

s_0 initial state;

S_g set of goal state(s);

H set of hosts (nodes) on the network;

ω_H host link weighting.

Formal Problem Statement

The planning agent receives the tuple, I_P, and creates a set of plans, P_I.

I_P is a Tuple

Σ	automated planning domain;
s_0	initial state;
S_g	set of goal state(s);
H	set of hosts (nodes) on the network;
ω_H	host link weighting.

Formal Problem Statement

Problem

To find and execute $p_I \in P_I$ where $p_I = \{a_0, a_1, \ldots, a_{|p_I|}\}$ and execution of p_I yields the best domain-dependent and network-centric evaluations.

Network-Awareness

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

Formal Problem Statement

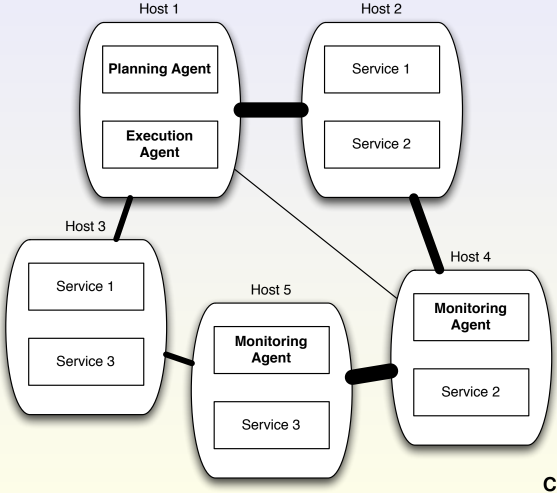
Problem

To find and execute $p_I \in P_I$ where $p_I = \{a_0, a_1, \ldots, a_{|p_I|}\}$ and execution of p_I yields the best domain-dependent and network-centric evaluations.

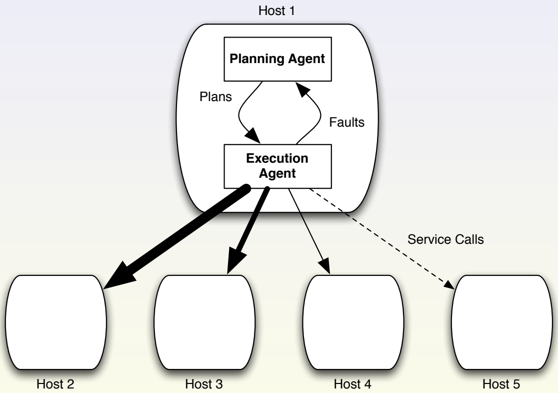
Network-Awareness

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

Formal Problem Statement

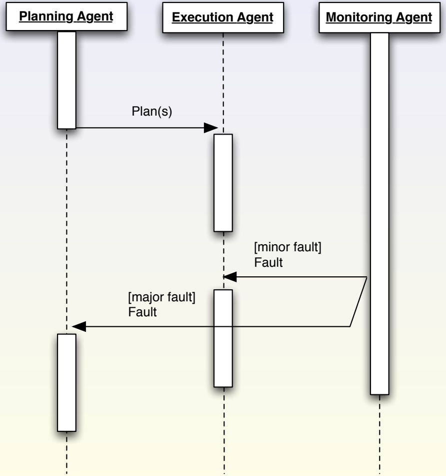


Formal Problem Statement



Department of Computer Science
Drexel University

Formal Problem Statement



Department of Computer Science

Outline

1. Introduction
   - Motivation
   - Background
   - Approach
2. Formalization
   - Problem Statement
3. Technical Approach
   - Planning Agents
   - Execution Agents
   - Monitoring Agents
   - Mixed-initiative UI
4. Experiments
   - Plan Evaluation Benchmarking
   - Network-Aware Agent Combinations
   - Discussion

Department of Computer Science

Drexel UNIVERSITY

Planning Domain Extensions

Operator distribution

- *e.g.*, NODE1ACTION(*parameters*)
- Implicit constraints.

Resource distribution

- *e.g.*, ACTION(*node1*, *parameters*)
- $s_0 \leftarrow s_0 \cup \{\textsc{Type}(node1) = \textsc{NetworkNode}\}$
- $s_0 \leftarrow s_0 \cup \{\textsc{Action}(node1) = \textbf{true}\}$

Planning Domain Extensions

Operator distribution

- *e.g.*, NODE1ACTION(*parameters*)
- Implicit constraints.

Resource distribution

- *e.g.*, ACTION(*node1*, *parameters*)
- $s_0 \leftarrow s_0 \cup \{\textsc{Type}(node1) = \textsc{NetworkNode}\}$
- $s_0 \leftarrow s_0 \cup \{\textsc{Action}(node1) = \textbf{true}\}$

Planning Domain Extensions

Operator distribution

- *e.g.*, NODE1ACTION(*parameters*)
- Implicit constraints.

Resource distribution

- *e.g.*, ACTION(*node1*, *parameters*)
- $s_0 \leftarrow s_0 \cup \{\text{TYPE}(node1) = \text{NETWORKNODE}\}$
- $s_0 \leftarrow s_0 \cup \{\text{ACTION}(node1) = \textbf{true}\}$

Planning Domain Extensions

Operator distribution

- *e.g.*, NODE1ACTION(*parameters*)
- Implicit constraints.

Resource distribution

- *e.g.*, ACTION(*node1*, *parameters*)
- $s_0 \leftarrow s_0 \cup \{\text{TYPE}(node1) = \text{NETWORKNODE}\}$
- $s_0 \leftarrow s_0 \cup \{\text{ACTION}(node1) = \textbf{true}\}$

Department of Computer Science

Drexel UNIVERSITY

Planning Domain Extensions

Operator distribution

- *e.g.*, Node1Action(*parameters*)
- Implicit constraints.

Resource distribution

- *e.g.*, Action(*node1*, *parameters*)
- $s_0 \leftarrow s_0 \cup \{\textsc{Type}(node1) = \textsc{NetworkNode}\}$
- $s_0 \leftarrow s_0 \cup \{\textsc{Action}(node1) = \textbf{true}\}$

Department of Computer Science

Drexel UNIVERSITY

Planning Domain Extensions

Operator distribution

- *e.g.*, NODE1ACTION(*parameters*)
- Impl

Resourc

- *e.g.*,
- $s_0 \leftarrow$
- $s_0 \leftarrow$

Complexity

- Operator distribution increases the number of actions in Σ to $|H| \times |A|$ in the worst case.
- Resource distributed increases the number of constraints in the world-state.

Department of Computer Science

Drexel UNIVERSITY

Planning Agents

Agent Types:

- Domain-Independent.
- Random.
- Guided.

Plan Evaluators:

- Steps.
- Alternatives.
- Longest temporally ordered path.
- Duplicate plans.

Department of Computer Science

Drexel UNIVERSITY

Planning Agents

Agent Types:

- Domain-Independent.
- Random.
- Guided.

Plan Evaluators:

- (none).

Department of Computer Science
Drexel UNIVERSITY

Planning Agents

Plan Evaluators:

- IED detection accuracy.
- Plan execution time.
- Network link quality.
- Network bandwidth usage.

Agent Types:

- Domain-Independent.
- Random.
- Guided.

Domain-Independent Planning Agent

- Uses I-Plan's default strategy.

I-Plan

University of Edinburgh, Tate *et al.* 's plan-space HTN planner which is built on an intelligent agent framework, I-X.

Process

1. Traverses search space depth-first.
2. Encounter an alternative whose constraints cannot be satisfied.
3. Backtracks using an A* search.

Domain-Independent Planning Agent

- Uses I-Plan's default strategy.

I-Plan

University of Edinburgh, Tate *et al.* 's plan-space HTN planner which is built on an intelligent agent framework, I-X.

Process

1. Traverses search space depth-first.
2. Encounter an alternative whose constraints cannot be satisfied.
3. Backtracks using an A* search.

Domain-Independent Planning Agent

- Uses I-Plan's default strategy.

I-Plan

University of Edinburgh, Tate *et al.* 's plan-space HTN planner which is built on an intelligent agent framework, I-X.

Process

1. Traverses search space depth-first.
2. Encounter an alternative whose constraints cannot be satisfied.
3. Backtracks using an A* search.

Domain-Independent Planning Agent

- Uses I-Plan’s default strategy.

I-Plan

University of Edinburgh, Tate *et al*. ’s plan-space HTN planner which is built on an intelligent agent framework, I-X.

Process

1. Traverses search space depth-first.
2. Encounter an alternative whose constraints cannot be satisfied.
3. Backtracks using an A* search.

Domain-Independent Planning Agent

- Uses I-Plan’s default strategy.

I-Plan

University of Edinburgh, Tate *et al*. ’s plan-space HTN planner which is built on an intelligent agent framework, I-X.

Process

1. Traverses search space depth-first.
2. Encounter an alternative whose constraints cannot be satisfied.
3. Backtracks using an A* search.

Computer Science

Random Planning Agent

- DFS with random branching.

Process

```
ConstructRandomPlan($I_P$)
toVisit.push($s_0$)
while $\neg$ toVisit.empty() $\wedge$ $\neg$ solution(toVisit.peek()) do
    $v \leftarrow$ toVisit.pop()
    if $v \notin$ visited then
        visited.add($v$)
        $r \leftarrow$ randomize($v$.children())
        toVisit.push($r$)
    end if
end while
return toVisit.peek()
```

Guided Planning Agent

Generates qualitatively-different plans over:

- Domain-dependent criteria, and
- Network-centric criteria.

Process

1. A priority queue exists for each evaluator.
2. Every partial-plan is evaluated by all evaluators and placed in their respective priority queues.
3. The partial-plan at the head of each priority queue is used for the next step.

Guided Planning Agent

Generates qualitatively-different plans over:

- Domain-dependent criteria, and
- Network-centric criteria.

Process

1. A priority queue exists for each evaluator.
2. Every partial-plan is evaluated by all evaluators and placed in their respective priority queues.
3. The partial-plan at the head of each priority queue is used for the next step.

Guided Planning Agent

Generates qualitatively-different plans over:

- Domain-dependent criteria, and
- Network-centric criteria.

Process

1. A priority queue exists for each evaluator.
2. Every partial-plan is evaluated by all evaluators and placed in their respective priority queues.
3. The partial-plan at the head of each priority queue is used for the next step.

Guided Planning Agent

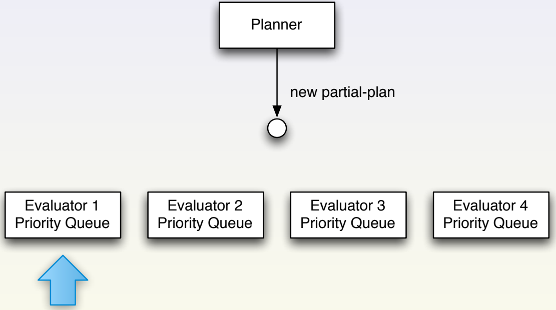
Generates qualitatively-different plans over:

- Domain-dependent criteria, and
- Network-centric criteria.

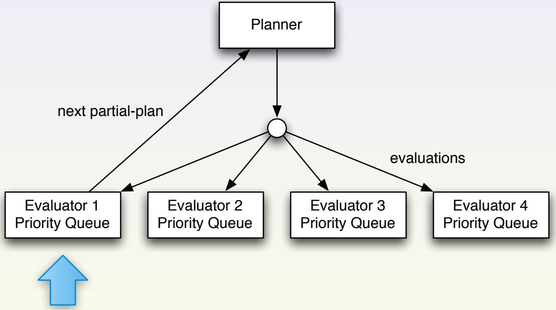
Process

1. A priority queue exists for each evaluator.
2. Every partial-plan is evaluated by all evaluators and placed in their respective priority queues.
3. The partial-plan at the head of each priority queue is used for the next step.

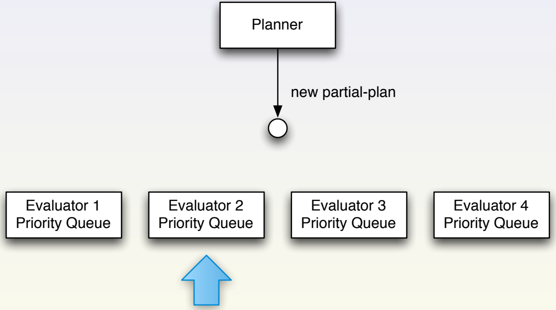
Guided Planning Agent


Planner
new partial-plan
Evaluator 1 Priority Queue
Evaluator 2 Priority Queue
Evaluator 3 Priority Queue
Evaluator 4 Priority Queue

Guided Planning Agent



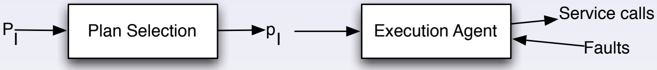
Guided Planning Agent



Outline

1. Introduction
   - Motivation
   - Background
   - Approach
2. Formalization
   - Problem Statement
3. Technical Approach
   - Planning Agents
   - Execution Agents
   - Monitoring Agents
   - Mixed-initiative UI
4. Experiments
   - Plan Evaluation Benchmarking
   - Network-Aware Agent Combinations
   - Discussion

Execution Agents



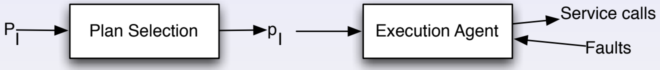
Agent types:

- Naïve.
- Reactive.
- Proactive.

Defined by:

- Service invocation.
- Error handling.

Execution Agents



Agent types:

- Naïve.
- Reactive.
- Proactive.

Defined by:

- Service invocation.
- Error handling.

Naïve Execution Agent

Naïve Execution Agent Properties

Service Invocation Invokes services exactly as described by p_I. The naïve agent requires that $\forall$ actions $a \in p_I, \text{host}(a) \neq \emptyset \wedge \text{resources}(a) \neq \{\}$.

Error Handling Ignores execution errors.

- **Not** network-aware.

Naïve Execution Agent

Naïve Execution Agent Properties

Service Invocation Invokes services exactly as described by p_I. The naïve agent requires that $\forall$ actions $a \in p_I, \text{host}(a) \neq \emptyset \wedge \text{resources}(a) \neq \{\}$.

Error Handling Ignores execution errors.

- **Not** network-aware.

Department of Computer Science

Drexel UNIVERSITY

Naïve Execution Agent

Naïve Execution Agent Properties

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.

- **Not** network-aware.

Department of Computer Science

Drexel UNIVERSITY

Reactive Execution Agent

Reactive Execution Agent Properties

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.

- Network-aware recovery — plan repair.
- Uses routing protocol neighbors & link quality.

Department of Computer Science

Reactive Execution Agent

Reactive Execution Agent Properties

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'.

- Network-aware recovery — plan repair.
- Uses routing protocol neighbors & link quality.

Reactive Execution Agent

Reactive Execution Agent Properties

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.

- Network-aware recovery — plan repair.
- Uses routing protocol neighbors & link quality.

Reactive Execution Agent

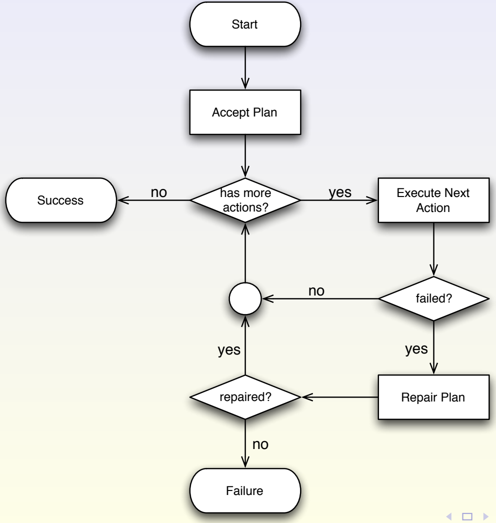
Reactive Execution Agent Properties

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.

- Network-aware recovery — plan repair.
- Uses routing protocol neighbors & link quality.

Reactive Execution Agent



Proactive Execution Agent

Proactive Execution Agent Properties

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 $\forall a \in p_I, \text{host}(a) = \emptyset \wedge \text{resources}(a) = \{\}$

Error Handling Repairs the failed p_I by replacing failed service call(s) with new ones, creating p'_I.

- Network-aware host/resource grounding.

Department of Computer Science
Drexel UNIVERSITY

Proactive Execution Agent

Proactive Execution Agent Properties

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 $\forall a \in p_I, \text{host}(a) = \emptyset \wedge \text{resources}(a) = \{\}$

Error Handling Repairs the failed p_I by replacing failed service call(s) with new ones, creating p'_I.

- Network-aware host/resource grounding.

Proactive Execution Agent

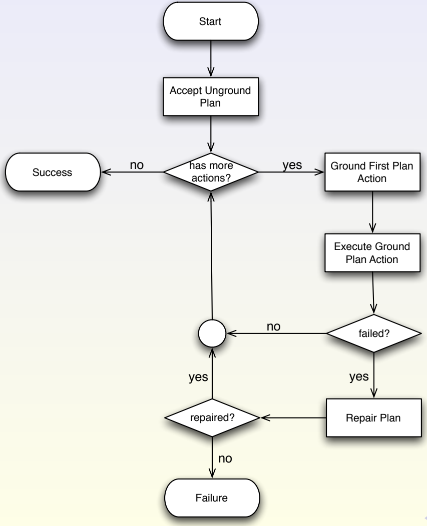
Proactive Execution Agent Properties

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 $\forall a \in p_I, \text{host}(a) = \emptyset \wedge \text{resources}(a) = \{\}$

Error Handling Repairs the failed p_I by replacing failed service call(s) with new ones, creating p'_I.

- Network-aware host/resource grounding.

Proactive Execution Agent



Outline

1. Introduction
   - Motivation
   - Background
   - Approach
2. Formalization
   - Problem Statement
3. Technical Approach
   - Planning Agents
   - Execution Agents
   - Monitoring Agents
   - Mixed-initiative UI
4. Experiments
   - Plan Evaluation Benchmarking
   - Network-Aware Agent Combinations
   - Discussion

Department of Computer Science
Drexel UNIVERSITY

Monitoring Agents

Methods of FDI

1. Analytic.
2. Data-driven.
3. Knowledge-based.

Monitoring Agents

Methods of FDI

1. Analytic. ← Active Monitor
2. Data-driven. ← Passive Monitor
3. Knowledge-based.

Department of Computer Science
Drexel UNIVERSITY

Analytic Monitoring Agent

Given the ordered plan $p_I = \{a_0, a_1, \ldots, a_{|p_I|}\}$

An analytic monitoring agent:

1. Constructs $p_M = \{m_0, m_1, \ldots, m_{|p_I|+1}\}$, an ordered set of monitoring actions;
2. Creates the new execution plan $p_I' = \bigcup_{i=0}^{n} \{m_i, a_i\}$;
3. The result is $p_I' = \{m_0, a_0, m_1, a_1, \ldots, m_{|p_I|}, a_{|p_I|}, m_{|p_I|+1}\}$.
4. Each $m \in p_M$ calculates the residual between expected and actual bytes transferred.

Analytic Monitoring Agent

Given the ordered plan $p_I = \{a_0, a_1, \ldots, a_{|p_I|}\}$

An analytic monitoring agent:

1. Constructs $p_M = \{m_0, m_1, \ldots, m_{|p_I|+1}\}$, an ordered set of monitoring actions;
2. Creates the new execution plan $p'_I = \bigcup_{i=0}^{n}\{m_i, a_i\}$;
3. The result is $p'_I = \{m_0, a_0, m_1, a_1, \ldots, m_{|p_I|}, a_{|p_I|}, m_{|p_I|+1}\}$.
4. Each $m \in p_M$ calculates the residual between expected and actual bytes transferred.

Analytic Monitoring Agent

Given the ordered plan $p_I = \{a_0, a_1, \ldots, a_{|p_I|}\}$

An analytic monitoring agent:

1. Constructs $p_M = \{m_0, m_1, \ldots, m_{|p_I|+1}\}$, an ordered set of monitoring actions;
2. Creates the new execution plan $p_I' = \bigcup_{i=0}^{n} \{m_i, a_i\}$;
3. The result is $p_I' = \{m_0, a_0, m_1, a_1, \ldots, m_{|p_I|}, a_{|p_I|}, m_{|p_I|+1}\}$.
4. Each $m \in p_M$ calculates the residual between expected and actual bytes transferred.

Analytic Monitoring Agent

Given the ordered plan $p_I = \{a_0, a_1, \ldots, a_{|p_I|}\}$

An analytic monitoring agent:

1. Constructs $p_M = \{m_0, m_1, \ldots, m_{|p_I|+1}\}$, an ordered set of monitoring actions;
2. Creates the new execution plan $p'_I = \bigcup_{i=0}^{n} \{m_i, a_i\}$;
3. The result is $p'_I = \{m_0, a_0, m_1, a_1, \ldots, m_{|p_I|}, a_{|p_I|}, m_{|p_I|+1}\}$.
4. Each $m \in p_M$ calculates the residual between expected and actual bytes transferred.

Department of Computer Science
Drexel UNIVERSITY

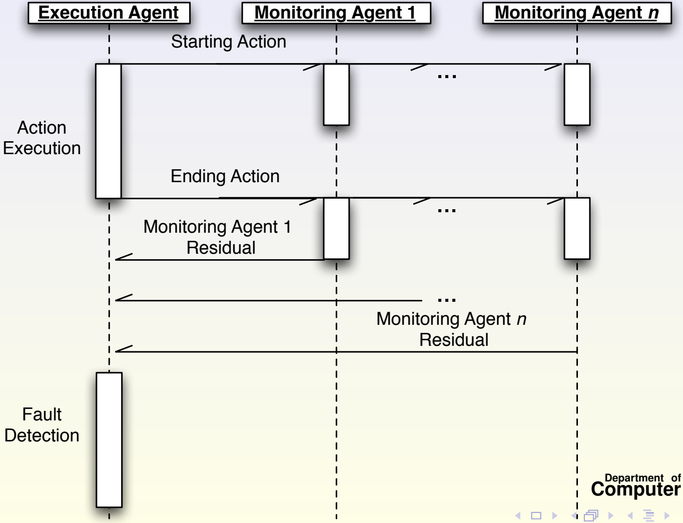
Analytic Monitoring Agent

Given the ordered plan $p_I = \{a_0, a_1, \ldots, a_{|p_I|}\}$

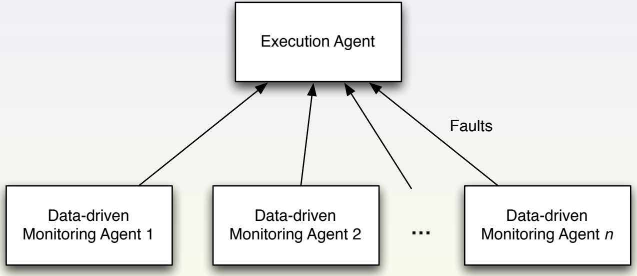
An analytic monitoring agent:

1. Constructs $p_M = \{m_0, m_1, \ldots, m_{|p_I|+1}\}$, an ordered set of monitoring actions;
2. Creates the new execution plan $p_I' = \bigcup_{i=0}^{n}\{m_i, a_i\}$;
3. The result is $p_I' = \{m_0, a_0, m_1, a_1, \ldots, m_{|p_I|}, a_{|p_I|}, m_{|p_I|+1}\}$.
4. Each $m \in p_M$ calculates the residual between expected and actual bytes transferred.

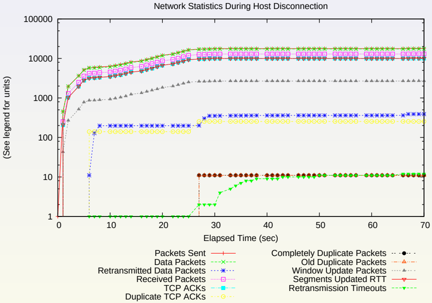
Analytic Monitoring Agent



Data-driven Monitoring Agent


Execution Agent
Faults
Data-driven Monitoring Agent 1
Data-driven Monitoring Agent 2
...
Data-driven Monitoring Agent *n*

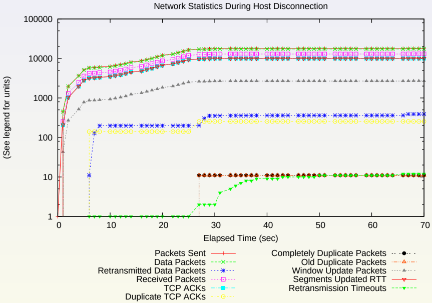
Data-driven Monitoring Agent



- Multivariate monitor.
- Data packets.
- Retransmission timeouts.

Department of Computer Science

Data-driven Monitoring Agent



- Multivariate monitor.
- Data packets.
- Retransmission timeouts.

Outline

1. Introduction
   - Motivation
   - Background
   - Approach
2. Formalization
   - Problem Statement
3. Technical Approach
   - Planning Agents
   - Execution Agents
   - Monitoring Agents
   - Mixed-initiative UI
4. Experiments
   - Plan Evaluation Benchmarking
   - Network-Aware Agent Combinations
   - Discussion

Department of Computer Science

Drexel UNIVERSITY

Plan Evaluation Criteria Statistics

Aspects

- Range (effective and theoretic).
- Direction (minimize or maximize).
- Statistics (e.g., mean, median, mode, standard deviation).

Benefit

Plans can be positioned along an absolute continuum of evaluation values.

Plan Evaluation Criteria Statistics

Aspects

- Range (effective and theoretic).
- Direction (minimize or maximize).
- Statistics (e.g., mean, median, mode, standard deviation).

Benefit

Plans can be positioned along an absolute continuum of evaluation values.

Plan Evaluation Criteria Statistics

Aspects

- Range (effective and theoretic).
- Direction (minimize or maximize).
- Statistics (e.g., mean, median, mode, standard deviation).

Benefit

Plans can be positioned along an absolute continuum of evaluation values.

Plan Evaluation Criteria Statistics

Aspects

- Range (effective and theoretic).
- Direction (minimize or maximize).
- Statistics (e.g., mean, median, mode, standard deviation).

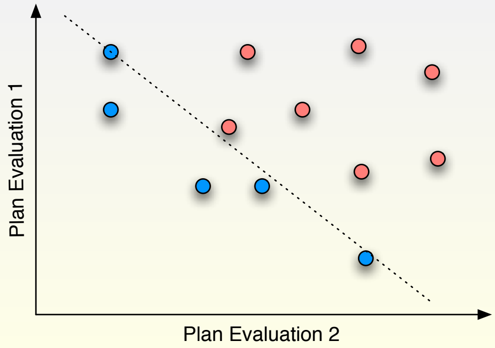
Benefit

Plans can be positioned along an absolute continuum of evaluation values.

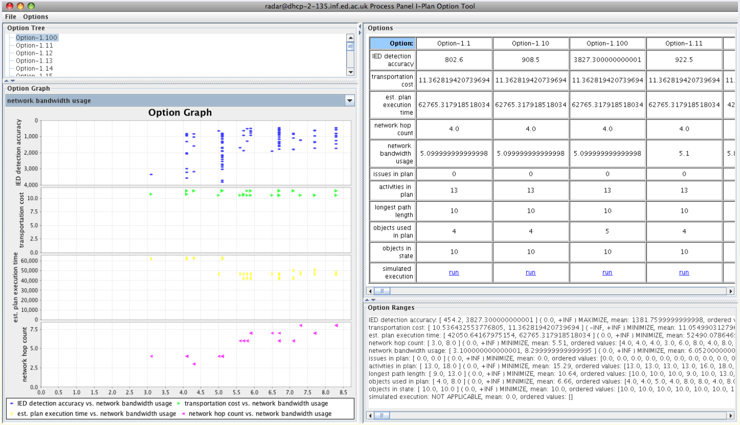
Dominant Plans

Definition

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



Plan Evaluation Visualization



radar@dhcp-2-135.inf.ed.ac.uk Process Panel I-Plan Option Tool

File Options

Option Tree

- Option-1.100
- Option-1.11
- Option-1.12
- Option-1.13
- Option-1.14
- Option-1.15

Option Graph

network bandwidth usage

Options

Option:	Option-1.1	Option-1.10	Option-1.100	Option-1.11	
IED detection accuracy	802.6	908.5	3827.300000000001	922.5	
transportation cost	11.362819420739694	11.362819420739694	11.362819420739694	11.362819420739694	11.
est. plan execution time	62765.317918518034	62765.317918518034	62765.317918518034	62765.317918518034	42
network hop count	4.0	4.0	4.0	4.0	
network bandwidth usage	5.099999999999998	5.099999999999998	5.099999999999998	5.1	5.8
issues in plan	0	0	0	0	
activities in plan	13	13	13	13	
longest path length	10	10	10	10	
objects used in plan	4	4	5	4	
objects in state	10	10	10	10	
simulated execution	run	run	run	run	

Option Ranges

IED detection accuracy: [454.2, 3827.300000000001] (0.0, +INF) MAXIMIZE, mean: 1381.7599999999998, ordered v
transportation cost: [10.536432553776805, 11.362819420739694] (-INF, +INF) MINIMIZE, mean: 11.05499031279
est. plan execution time: [42050.64167975154, 62765.317918518034] (0.0, +INF) MINIMIZE, mean: 52490.078646:
network hop count: [3.0, 8.0] (0.0, +INF) MINIMIZE, mean: 5.51, ordered values: [4.0, 4.0, 4.0, 3.0, 6.0, 8.0, 4.0, 8.0,
network bandwidth usage: [3.100000000000001, 8.299999999999995] (0.0, +INF) MINIMIZE, mean: 6.0520000000
issues in plan: [0.0, 0.0] (0.0, +INF) MINIMIZE, mean: 0.0, ordered values: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0
activities in plan: [13.0, 18.0] (0.0, +INF) MINIMIZE, mean: 15.29, ordered values: [13.0, 13.0, 13.0, 13.0, 16.0, 18.0,
longest path length: [9.0, 13.0] (0.0, +INF) MINIMIZE, mean: 10.64, ordered values: [10.0, 10.0, 10.0, 9.0, 10.0, 13.0,
objects used in plan: [4.0, 8.0] (0.0, +INF) MINIMIZE, mean: 6.66, ordered values: [4.0, 4.0, 5.0, 4.0, 8.0, 8.0, 4.0, 8.0
objects in state: [10.0, 10.0] (0.0, +INF) MINIMIZE, mean: 10.0, ordered values: [10.0, 10.0, 10.0, 10.0, 10.0, 10.0, 1
simulated execution: NOT APPLICABLE, mean: 0.0, ordered values: []

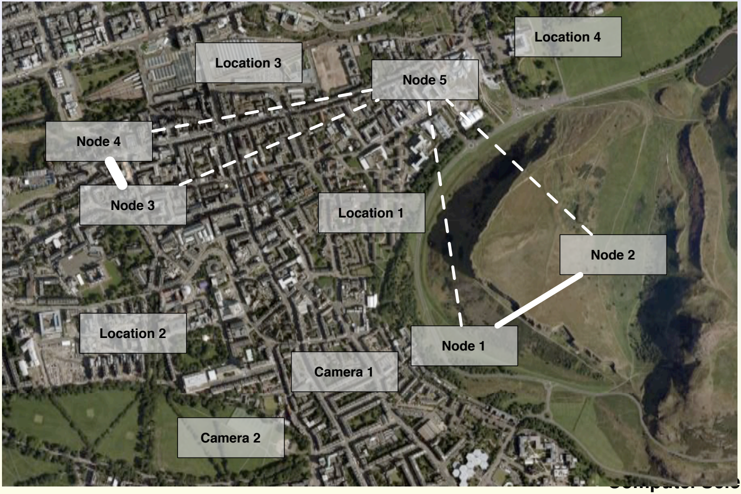
Department of Computer Science

Drexel UNIVERSITY

Outline

1. Introduction
   - Motivation
   - Background
   - Approach
2. Formalization
   - Problem Statement
3. Technical Approach
   - Planning Agents
   - Execution Agents
   - Monitoring Agents
   - Mixed-initiative UI
4. Experiments
   - Plan Evaluation Benchmarking
   - Network-Aware Agent Combinations
   - Discussion

Experiment: Plan Evaluation Benchmarking



Plan Evaluation Benchmarking

Action	Providing Hosts
PHYSICALMOVE	all
ACQUIRECAMERA	all
TAKEPHOTO	all
GETOLDPHOTO	all
RELEASECAMERA	all
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

Plan Evaluation Benchmarking

Camera	Resolution
Camera 1	3.2 MP
Camera 2	8.0 MP

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

Plan Evaluation Benchmarking Results

Each planning algorithm ran in I-Plan for five minutes.

σ Plan Evaluations

	ω_H	Bandwidth	IED Acc.	Time
I-Plan Default	0.949	0.759	291.4	8216
Random	1.647	**1.476**	177.9	7220
Guided	**1.916**	1.141	**392.6**	**14050**

Dominant Plans

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

Plan Evaluation Benchmarking Results

Each planning algorithm ran in I-Plan for five minutes.

σ Plan Evaluations

	ω_H	Bandwidth	IED Acc.	Time
I-Plan Default	0.949	0.759	291.4	8216
Random	1.647	**1.476**	177.9	7220
Guided	**1.916**	1.141	**392.6**	**14050**

Dominant Plans

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

Outline

1 Introduction
- Motivation
- Background
- Approach

2 Formalization
- Problem Statement

3 Technical Approach
- Planning Agents
- Execution Agents
- Monitoring Agents
- Mixed-initiative UI

4 Experiments
- Plan Evaluation Benchmarking
- Network-Aware Agent Combinations
- Discussion

Department of Computer Science
Drexel UNIVERSITY

Experiment: Network-Aware Agent Combinations

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

Department of
Computer Science

Experimental Setup

- Multi-objective Optimization (MOO) Function.
- Implemented agents with I-X and I-Plan.
- Network emulation.
- Mobility models.

MOO function

$$\text{MOO}(p_I) = \text{IEDDetectAcc}(p_I) + 3 \times \text{TranspCost}(p_I) + 5 \times \text{ExecTime}(p_I) + \text{LinkQuality}(p_I) + \text{BandwidthUse}(p_I)$$

Experimental Setup

- Multi-objective Optimization (MOO) Function.
- Implemented agents with I-X and I-Plan.
- Network emulation.
- Mobility models.

MOO function

$$\text{MOO}(p_I) = \text{IEDDetectAcc}(p_I) + 3 \times \text{TranspCost}(p_I) + 5 \times \text{ExecTime}(p_I) + \text{LinkQuality}(p_I) + \text{BandwidthUse}(p_I)$$

Experimental Setup

- Multi-objective Optimization (MOO) Function.
- Implemented agents with I-X and I-Plan.
- Network emulation.
- Mobility models.

MOO function

$$\text{MOO}(p_I) = \text{IEDDetectAcc}(p_I) + 3 \times \text{TranspCost}(p_I) + 5 \times \text{ExecTime}(p_I) + \text{LinkQuality}(p_I) + \text{BandwidthUse}(p_I)$$

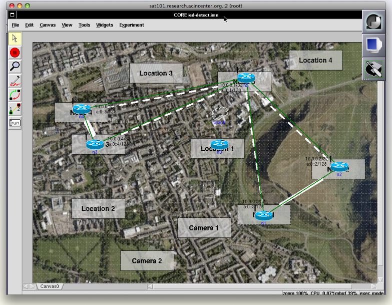
Experimental Setup

- Multi-objective Optimization (MOO) Function.
- Implemented agents with I-X and I-Plan.
- Network emulation.
- Mobility models.

MOO function

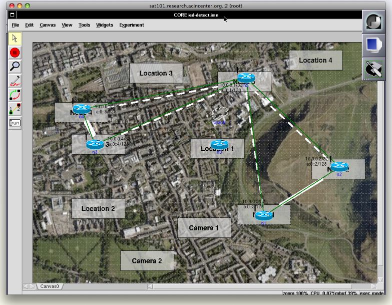
$$\text{MOO}(p_I) = \text{IEDDetectAcc}(p_I) + 3 \times \text{TranspCost}(p_I) + 5 \times \text{ExecTime}(p_I) + \text{LinkQuality}(p_I) + \text{BandwidthUse}(p_I)$$

CORE



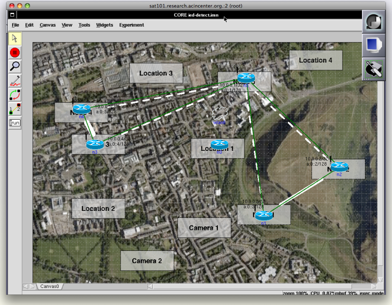
- Boeing's Common Open Research Emulator.
- FreeBSD network stack emulation.
- Simple Multicast Forwarding (SMF).
- Open Shortest Path First (OSPF).

CORE



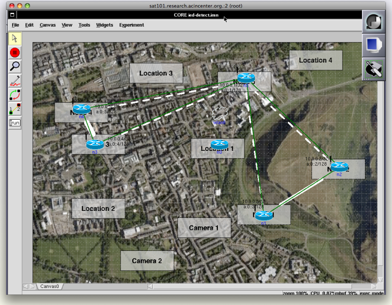
- Boeing's Common Open Research Emulator.
- FreeBSD network stack emulation.
- Simple Multicast Forwarding (SMF).
- Open Shortest Path First (OSPF).

CORE



- Boeing's Common Open Research Emulator.
- FreeBSD network stack emulation.
- Simple Multicast Forwarding (SMF).
- Open Shortest Path First (OSPF).

CORE



- Boeing's Common Open Research Emulator.
- FreeBSD network stack emulation.
- Simple Multicast Forwarding (SMF).
- Open Shortest Path First (OSPF).

Mobility Models

Purpose

- Dictate geographical node locations.
- Dynamic ω_H.

Mobility Patterns

1. Local.
2. Static.
3. Dynamic.
4. Partition-merge.

Department of Computer Science

Drexel UNIVERSITY

Mobility Models

Purpose

- Dictate geographical node locations.
- Dynamic ω_H.

Mobility Patterns

1. Local.
2. Static.
3. Dynamic.
4. Partition-merge.

Department of Computer Science

Drexel UNIVERSITY

Mobility Models

Purpose

- Dictate geographical node locations.
- Dynamic ω_H.

Mobility Patterns

1. Local.
2. Static.
3. Dynamic.
4. Partition-merge.

Department of Computer Science

Mobility Models

Purpose

- Dictate geographical node locations.
- Dynamic ω_H.

Mobility Patterns

1. Local.
2. Static.
3. Dynamic.
4. Partition-merge.

Department of Computer Science

Drexel UNIVERSITY

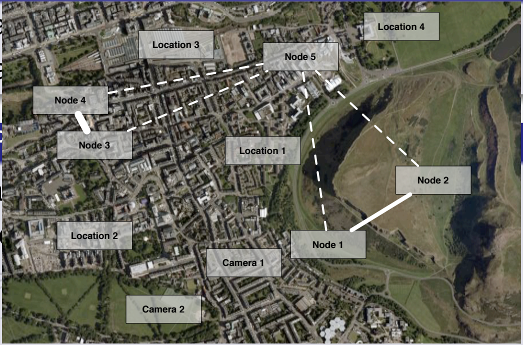
Mobility Models

Purpose

- Dicta
- Dyna

Mobility P

1. Loca
2. Stati
3. Dyna
4. Parti



Mobility Models

Purpose

- Dictate geographical node locations.
- Dynamic ω_H.

Mobility Patterns

1. Local.
2. Static.
3. Dynamic.
4. Partition-merge.

Department of Computer Science

Mobility Models

Purpose

- Dictate geographical node locations.
- Dynamic ω_H.

Mobility Patterns

1. Local.
2. Static.
3. Dynamic.
4. Partition-merge.

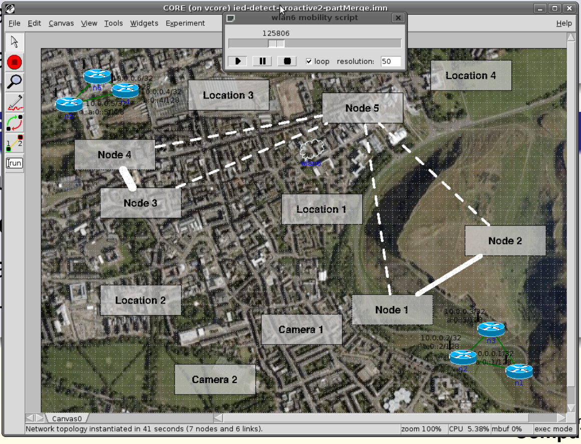
Mobility Models

Purpose

- Dicta
- Dyna

Mobility F

1. Loca
2. Stati
3. Dyna
4. Parti



Network topology instantiated in 41 seconds (7 nodes and 6 links).

nt of
ter Science

Domain-independent Plan

```
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
```

Random Plan

```
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
```

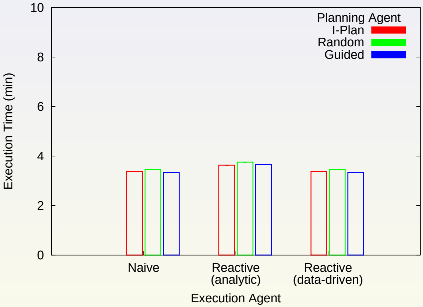
Guided Plan

```
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
```

Department of
Computer Science

Drexel
UNIVERSITY

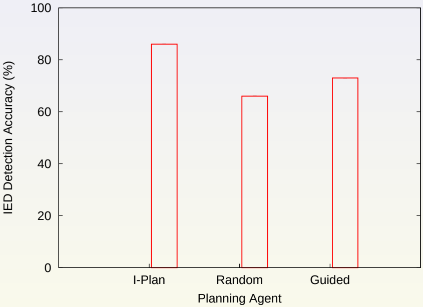
Local Results: Mean Time



- Network **not** a factor.
- Network-awareness did not hurt.

Department of Computer Science

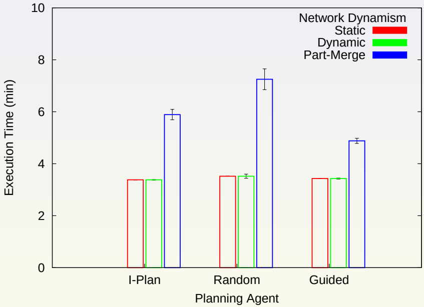
Local Results: Mean IED Detection Accuracy



- Ideal values of IED detection accuracy.

Department of Computer Science

Planning Agent Comparison



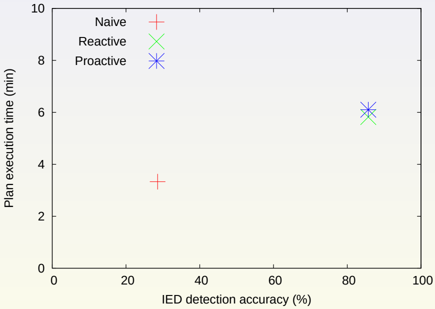
- Network disruptions adversely effect plan execution times.
- Guided was 16.7% faster than I-Plan and 28.8% faster than random in part-merge.

Department of Computer Science

Drexel UNIVERSITY

Execution Agent Effectiveness

Planning Agent: domain-independent (I-Plan default)

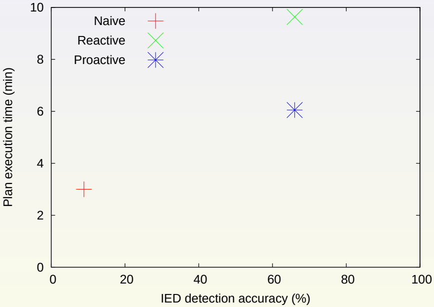


- Naïve agent has the lowest IED detection accuracy and exec. time.
- Reactive and proactive agents achieved ideal IED detection accuracies.

Department of Computer Science

Execution Agent Effectiveness

Planning Agent: random

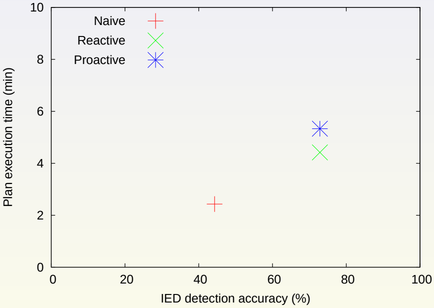


- Naïve agent failed most often.
- Proactive agent finished considerably faster than reactive.

Department of Computer Science

Execution Agent Effectiveness

Planning Agent: guided (network-aware)

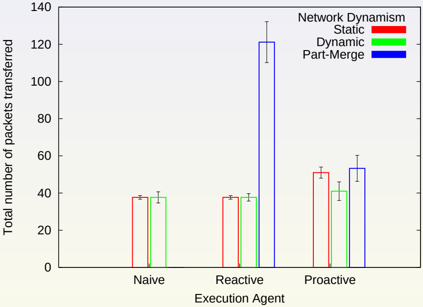


- Naïve agent failed most often.
- The guided algorithm advice significantly helped the execution agent.

Department of Computer Science

Drexel UNIVERSITY

Execution Agent Performance



- Proactive agent uses slightly more network transmissions under connected mobility patterns.
- Under part-merge, the proactive agent sent fewer than half as many packets as the reactive agent.

Department of Computer Science

Drexel UNIVERSITY

Monitoring Agent Comparisons

Analytic Monitoring Agent

- High percentage of false-positives.
- Communication errors → incorrect residuals.
- Active monitor.

- Analytic monitors are less-suitable for network-centric domains.

Department of Computer Science

Monitoring Agent Comparisons

Analytic Monitoring Agent

- High percentage of false-positives.
- Communication errors → incorrect residuals.
- Active monitor.

- Analytic monitors are less-suitable for network-centric domains.

Department of Computer Science

Drexel UNIVERSITY

Monitoring Agent Comparisons

Analytic Monitoring Agent

- High percentage of false-positives.
- Communication errors → incorrect residuals.
- Active monitor.

- Analytic monitors are less-suitable for network-centric domains.

Monitoring Agent Comparisons

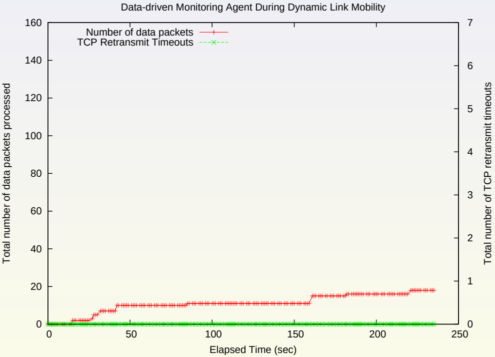
Analytic Monitoring Agent

- High percentage of false-positives.
- Communication errors → incorrect residuals.
- Active monitor.

- Analytic monitors are less-suitable for network-centric domains.

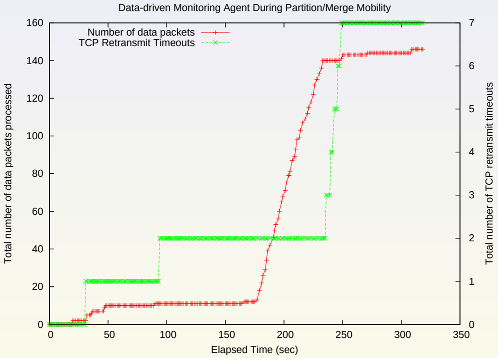
Data-driven Monitoring Agent

Normal execution:



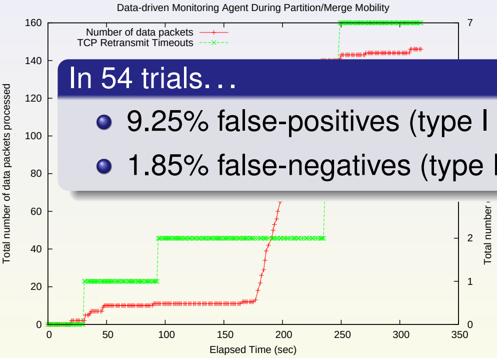
Data-driven Monitoring Agent

Network disconnection:



Data-driven Monitoring Agent

Network disconnection:



In 54 trials. . .

- 9.25% false-positives (type I error).
- 1.85% false-negatives (type II error).

Outline

1 Introduction
- Motivation
- Background
- Approach

2 Formalization
- Problem Statement

3 Technical Approach
- Planning Agents
- Execution Agents
- Monitoring Agents
- Mixed-initiative UI

4 Experiments
- Plan Evaluation Benchmarking
- Network-Aware Agent Combinations
- Discussion

Department of Computer Science

Main Contributions

1. Qualitatively-different plan generation:
    - Qualitatively different plans over a range of plan evaluation criteria.
    - Visualizing plan evaluations.
2. Network-aware agents:
    - Network-aware planning agent.
    - Network-aware execution agents.
    - Network-aware monitoring agents.

Main Contributions

1. Qualitatively-different plan generation:
   - Qualitatively different plans over a range of plan evaluation criteria.
   - Visualizing plan evaluations.
2. Network-aware agents:
   - Network-aware planning agent.
   - Network-aware execution agents.
   - Network-aware monitoring agents.

Main Contributions

1. Qualitatively-different plan generation:
    - Qualitatively different plans over a range of plan evaluation criteria.
    - Visualizing plan evaluations.
2. Network-aware agents:
    - Network-aware planning agent.
    - Network-aware execution agents.
    - Network-aware monitoring agents.

Department of Computer Science

Drexel UNIVERSITY

Main Contributions

1. Qualitatively-different plan generation:
    - Qualitatively different plans over a range of plan evaluation criteria.
    - Visualizing plan evaluations.
2. Network-aware agents:
    - Network-aware planning agent.
    - Network-aware execution agents.
    - Network-aware monitoring agents.

Main Contributions

1. Qualitatively-different plan generation:
    - Qualitatively different plans over a range of plan evaluation criteria.
    - Visualizing plan evaluations.
2. Network-aware agents:
    - Network-aware planning agent.
    - Network-aware execution agents.
    - Network-aware monitoring agents.

Department of Computer Science

Main Contributions

1. Qualitatively-different plan generation:
   - Qualitatively different plans over a range of plan evaluation criteria.
   - Visualizing plan evaluations.
2. Network-aware agents:
   - Network-aware planning agent.
   - Network-aware execution agents.
   - Network-aware monitoring agents.

Department of Computer Science

Main Contributions

1. Qualitatively-different plan generation:
    - Qualitatively different plans over a range of plan evaluation criteria.
    - Visualizing plan evaluations.
2. Network-aware agents:
    - Network-aware planning agent.
    - Network-aware execution agents.
    - Network-aware monitoring agents.

Future Work

- Knowledge-based monitoring agents.
- Incorporate the effects of planning actions into heuristics.

Future Work

- Knowledge-based monitoring agents.
- Incorporate the effects of planning actions into heuristics.

Department of Computer Science

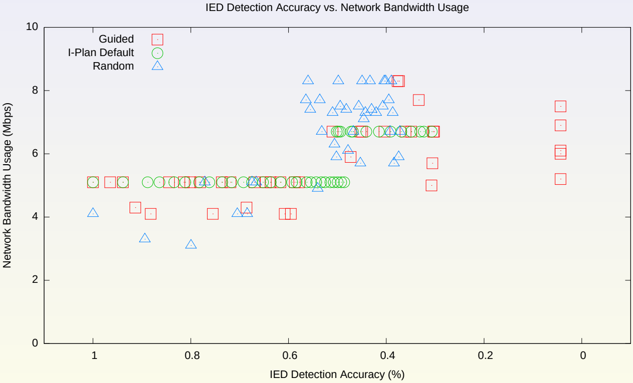
Drexel UNIVERSITY

Acknowledgements

- Advisor:
  - Dr. William C. Regli
- Committee:
  - Dr. Rachel Greenstadt
  - Dr. Ani Hsieh
- Conceptual Contributors:
  - Prof. Austin Tate
  - Dr. Gerhard Wickler
  - Jeff Dalton
- Co-workers
- Critiquors:
  - Ilya Braude
  - Matt Chase
  - Patrick Freestone
  - Joe Kopena
  - Duc Nguyen
  - Rob Lass
  - Evan Sultanik
- Family & Friends
- LaTeX, Vim, opensource software

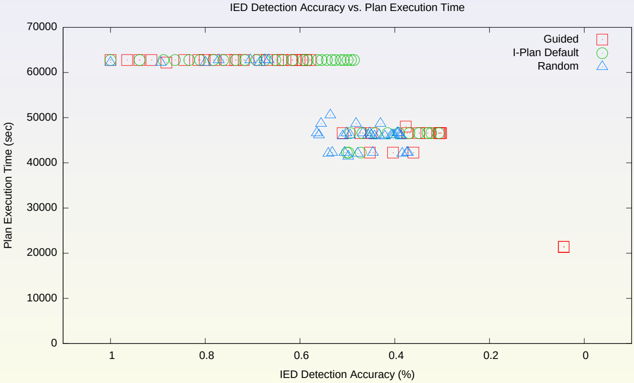
Department of Computer Science

IED Detection Accuracy and Bandwidth Usage



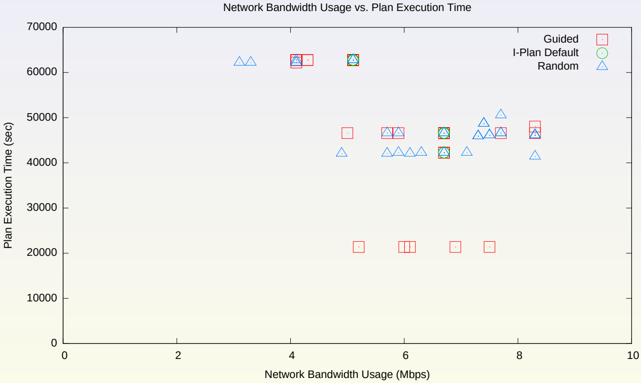
Department of Computer Science

IED Detection Accuracy and Execution Time

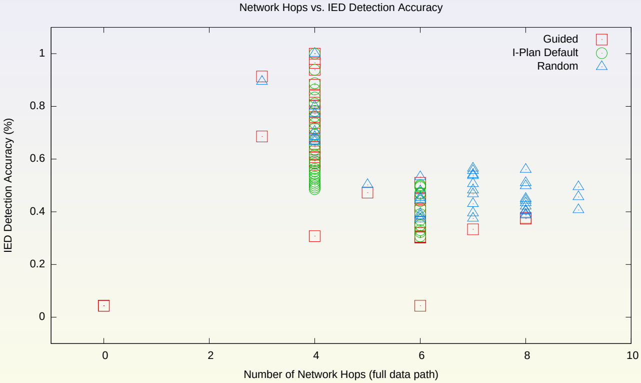


Department of Computer Science

Network Bandwidth Usage and Execution Time

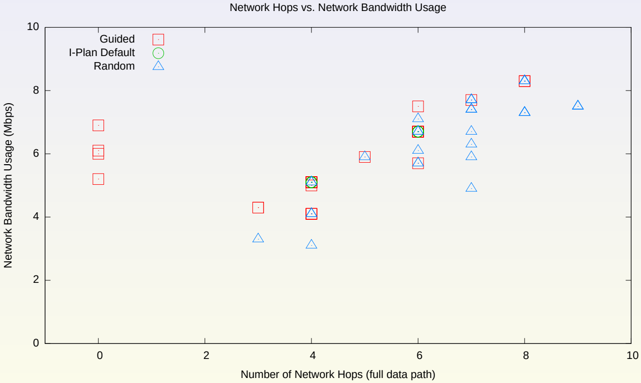


Network Hops and IED Detection Accuracy

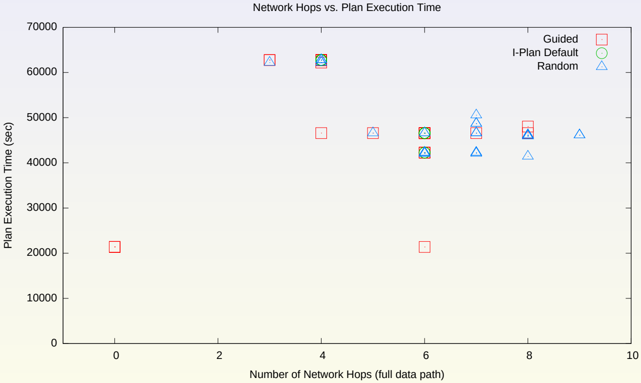


Department of Computer Science

Network Hops and Bandwidth Usage

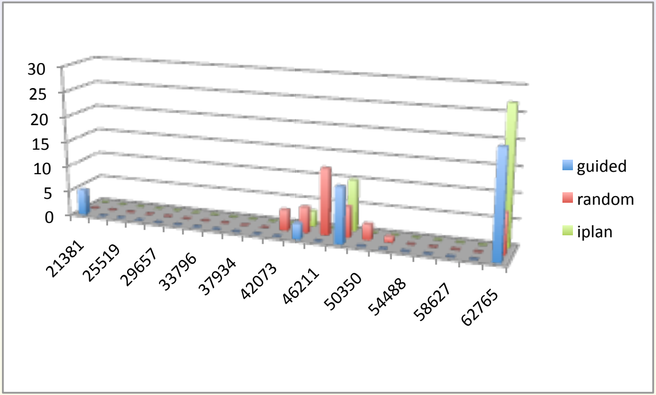


Network Hops and Execution Time



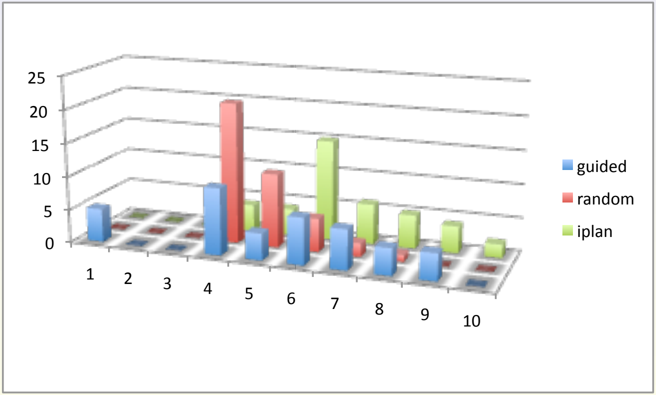
Department of Computer Science

Plan Eval. Benchmarking Execution Time Distribution



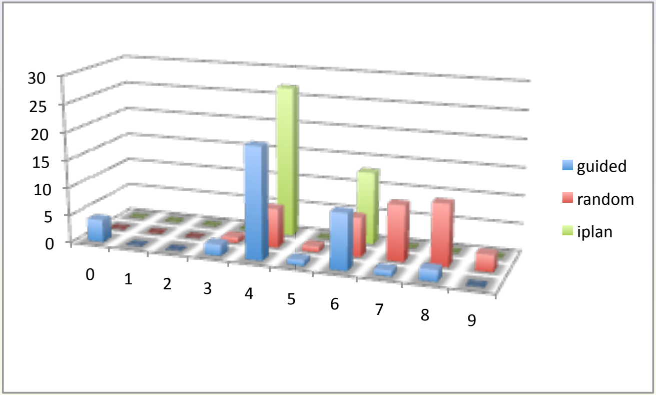
Department of
Computer Science
Drexel
UNIVERSITY

Plan Eval. Benchmarking IED Detect. Acc. Distribution



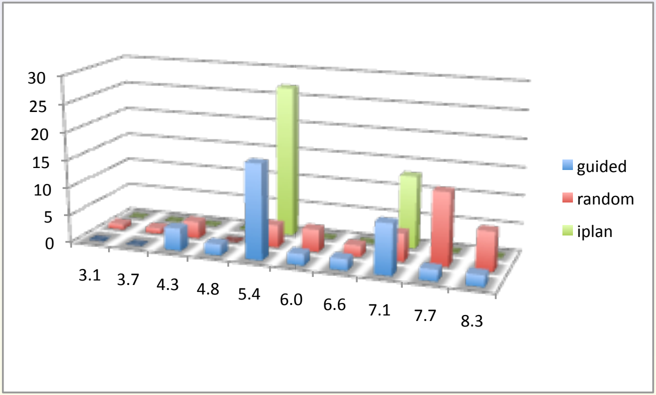
Department of Computer Science

Plan Eval. Benchmarking Link Quality Distribution



Department of
Computer Science

Plan Eval. Benchmarking Bandwidth Usage Distribution


guided
random
iplan