A Framework for Planning in Continuous-time Stochastic Domains

Håkan L. S. Younes
Carnegie Mellon University

David J. Musliner
Honeywell Laboratories

Reid G. Simmons
Carnegie Mellon University
Introduction

- Policy generation for complex domains
  - Uncertainty in outcome and timing of actions and events
  - Time as a continuous quantity
  - Concurrency

- Rich goal formalism
  - Achievement, maintenance, prevention
  - Deadlines
Motivating Example

- Deliver package from CMU to Honeywell
Elements of Uncertainty

- Uncertain duration of flight and taxi ride
- Plane can get full without reservation
- Taxi might not be at airport when arriving in Minneapolis
- Package can get lost at airports
Modeling Uncertainty

- Associate a delay distribution $F(t)$ with each action/event $a$
- $F(t)$ is the cumulative distribution function for the delay from when $a$ is enabled until it triggers

Pittsburgh Taxi

1. Driving
2. At airport

```
U(20,40)
```

arrive
Concurrency

- Concurrent semi-Markov processes

Generalized semi-Markov process

Pittsburgh Taxi

Minneapolis Taxi

Concurrent semi-Markov processes
Rich Goal Formalism

- Goals specified as CSL formulae
  - $\phi ::= true \mid a \mid \phi \land \phi \mid \neg \phi \mid Pr_{\geq \theta}(\rho)$
  - $\rho ::= \phi U^{\leq t} \phi \mid \Diamond^{\leq t} \phi \mid \Box^{\leq t} \phi$
Goal for Motivating Example

- Probability at least 0.9 that the package reaches Honeywell within 300 minutes without getting lost on the way
  - $\Pr_{\geq 0.9}(\neg \text{pkg lost } U\leq 300 \text{ pkg@Honeywell})$
Problem Specification

- **Given:**
  - Complex domain model
    - Stochastic discrete event system
  - Initial state
  - Probabilistic temporally extended goal
    - CSL formula

- **Wanted:**
  - Policy satisfying goal formula in initial state
Generate, Test and Debug

[Simmons 88]

Generate initial policy

Test if policy is good

Debug and repair policy

repeat

good

bad
Generate

- Ways of generating initial policy
  - Generate policy for relaxed problem
  - Use existing policy for similar problem
  - Start with null policy
  - Start with random policy

Not focus of this talk!
Test

- Use **discrete event simulation** to generate sample execution paths
- Use **acceptance sampling** to verify probabilistic CSL goal conditions
Debug

- Analyze sample paths generated in test step to find reasons for failure
- Change policy to reflect outcome of failure analysis
More on Test Step

Generate initial policy

Test

if policy is good

Debug and repair policy

Test
Error Due to Sampling

- Probability of false negative: $\leq \alpha$
  - Rejecting a good policy
- Probability of false positive: $\leq \beta$
  - Accepting a bad policy

$(1-\beta)$-soundness
Acceptance Sampling

- Hypothesis: $\Pr_{\geq \theta}(\rho)$
Performance of Test

Actual probability of \( \rho \) holding

Probability of accepting \( \Pr_{\geq \theta}(\rho) \) as true

1 - \( \alpha \)

\( \beta \)

\( \theta \)

Actual probability of \( \rho \) holding
Ideal Performance

Actual probability of $\rho$ holding

Probability of accepting $\Pr_{\geq \theta} (\rho)$ as true

$1 - \alpha$

$\beta$

False negatives

False positives

Actual probability of $\rho$ holding
Realistic Performance

$$\Pr_{\geq \theta}(\rho) \text{ as true}$$

- False positives
- False negatives

Actual probability of $\rho$ holding

Indifference region

$$\theta - \delta \quad \theta \quad \theta + \delta$$
Sequential Acceptance Sampling [Wald 45]

- Hypothesis: $\Pr_{\geq \theta}(\rho)$

True, false, or another sample?
Graphical Representation of Sequential Test

Number of positive samples

Number of samples
We can find an acceptance line and a rejection line given $\theta$, $\delta$, $\alpha$, and $\beta$.
Graphical Representation of Sequential Test

- Reject hypothesis

![Graphical Representation](image-url)
Graphical Representation of Sequential Test

- Accept hypothesis
Anytime Policy Verification

- Find best acceptance and rejection lines after each sample in terms of \( \alpha \) and \( \beta \)
Verification Example

- Initial policy for example problem

![Graph showing CPU time (seconds) vs. Error probability with λ = 0.01]
More on Debug Step

1. Generate initial policy
2. Test if policy is good
3. Debug and repair policy
Role of Negative Sample Paths

- Negative sample paths provide evidence on how policy can fail
  - “Counter examples”
Generic Repair Procedure

1. Select some state along some negative sample path
2. Change the action planned for the selected state

Need heuristics to make informed state/action choices
Scoring States

- Assign $-1$ to last state along negative sample path and propagate backwards
- Add over all negative sample paths
Example

- Package gets lost at Minneapolis airport while waiting for the taxi
- Repair: store package until taxi arrives
Verification of Repaired Policy

Error probability vs. CPU time (seconds)

- Red line: reject
- Green line: accept

Error probability: \( \delta = 0.01 \)
Comparing Policies

- Use acceptance sampling:
  - Pair samples from the verification of two policies
  - Count pairs where policies differ
  - Prefer first policy if probability is at least 0.5 of pairs where first policy is better
Summary

- Framework for dealing with complex stochastic domains
- Efficient sampling-based anytime verification of policies
- Initial work on debug and repair heuristics