Proactive Latency-Aware Self-Adaptation under Uncertainty

Gabriel Moreno

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

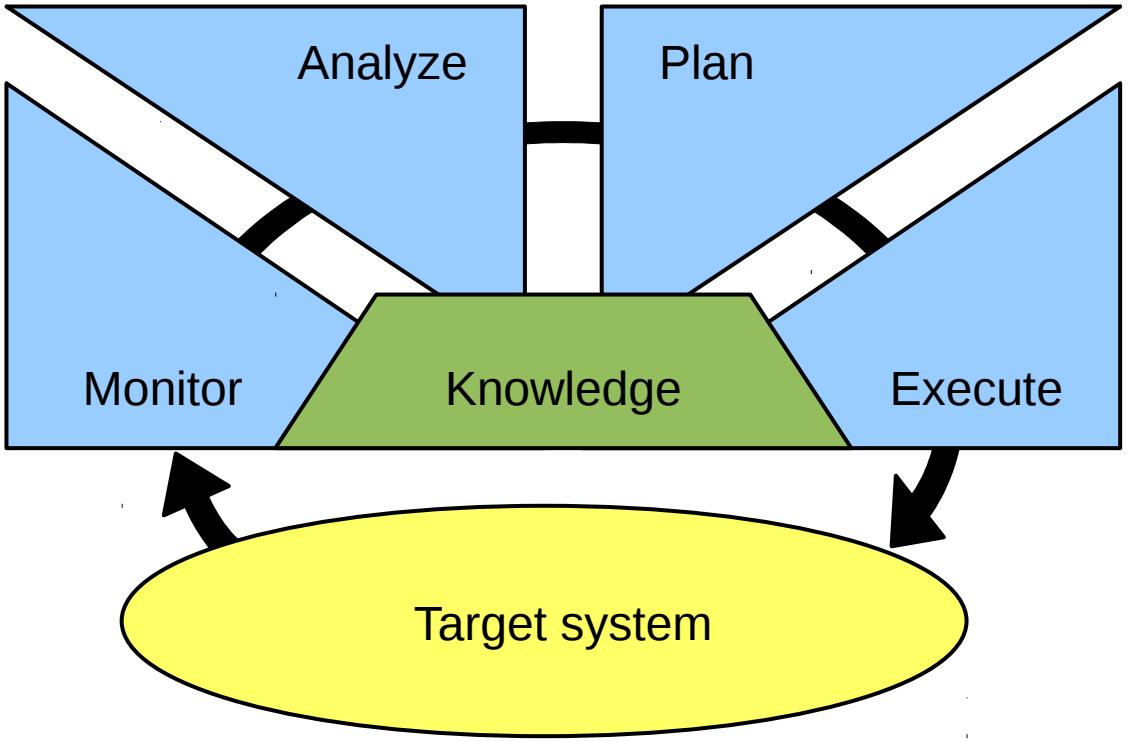
DM17-0959

Black Friday

Self-Adaptive Software evaluates its own behavior and changes behavior when the evaluation indicates that it is not accomplishing what the software is intended to do, or when better functionality or performance is possible.

[DARPA BAA-98-12]

MAPE-K loop



J.O. Kephart, and D.M. Chess. "The vision of autonomic computing." *Computer* 36, no. 1, 2003

Adaptation tactics

Action primitives that change the system

Used to achieve adaptation goals

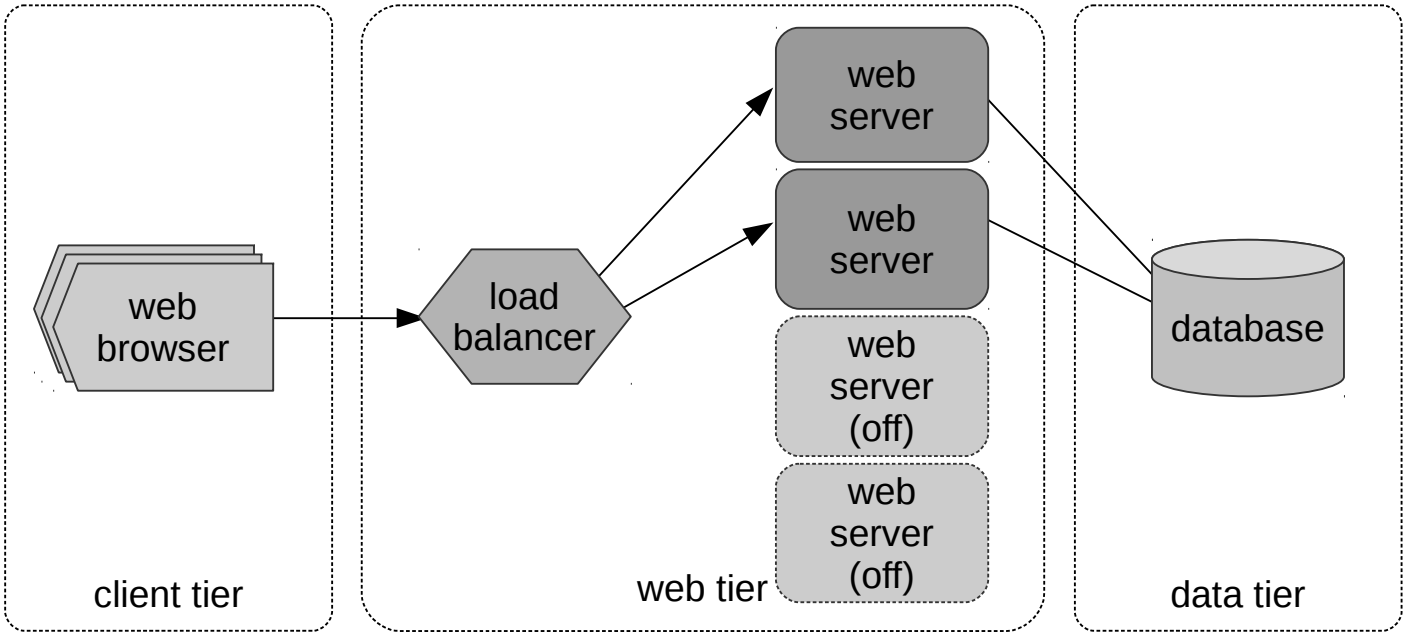
- e.g., add server to deal with increased load

Tactics have latency

Time since tactic is started until it produces its effect

- switching web page to text-only: < 1 sec
- turning on a GPS receiver: 30 sec
- spin up VM in the cloud: minutes

RUBiS



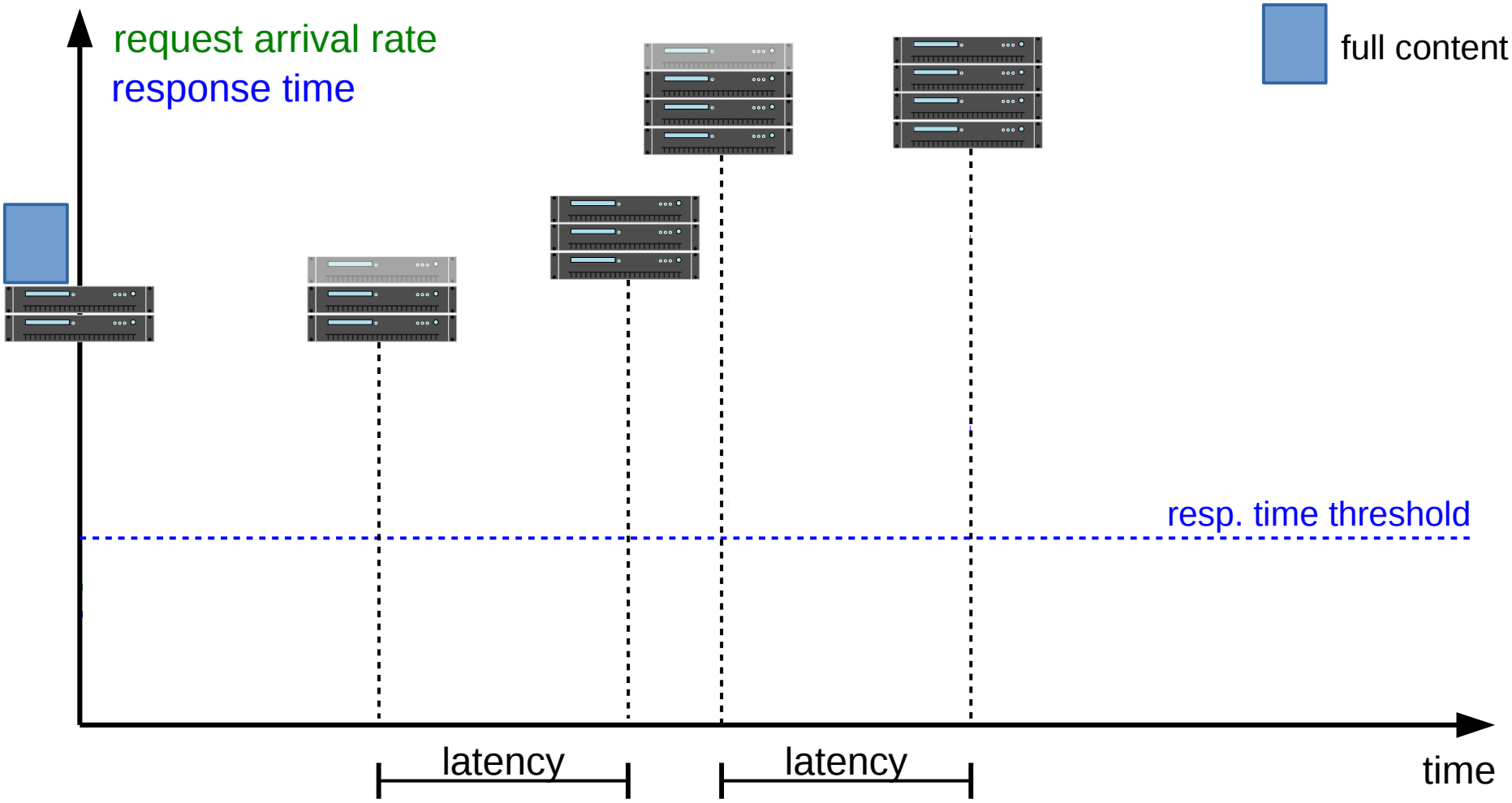
Adaptation tactics:

- add/remove servers
- increase/decrease proportion of optional content (a.k.a. *dimmer*)

Adaptation goal:

- keep response time below threshold
- serve as much optional content as possible
- minimize cost (servers)

Reactive time-agnostic adaptation in RUBiS



Limitations of reactive time-agnostic adaptation

lags behind environment changes

cannot favor a fast tactic over a slower one

cannot complement a slow tactic with a fast one

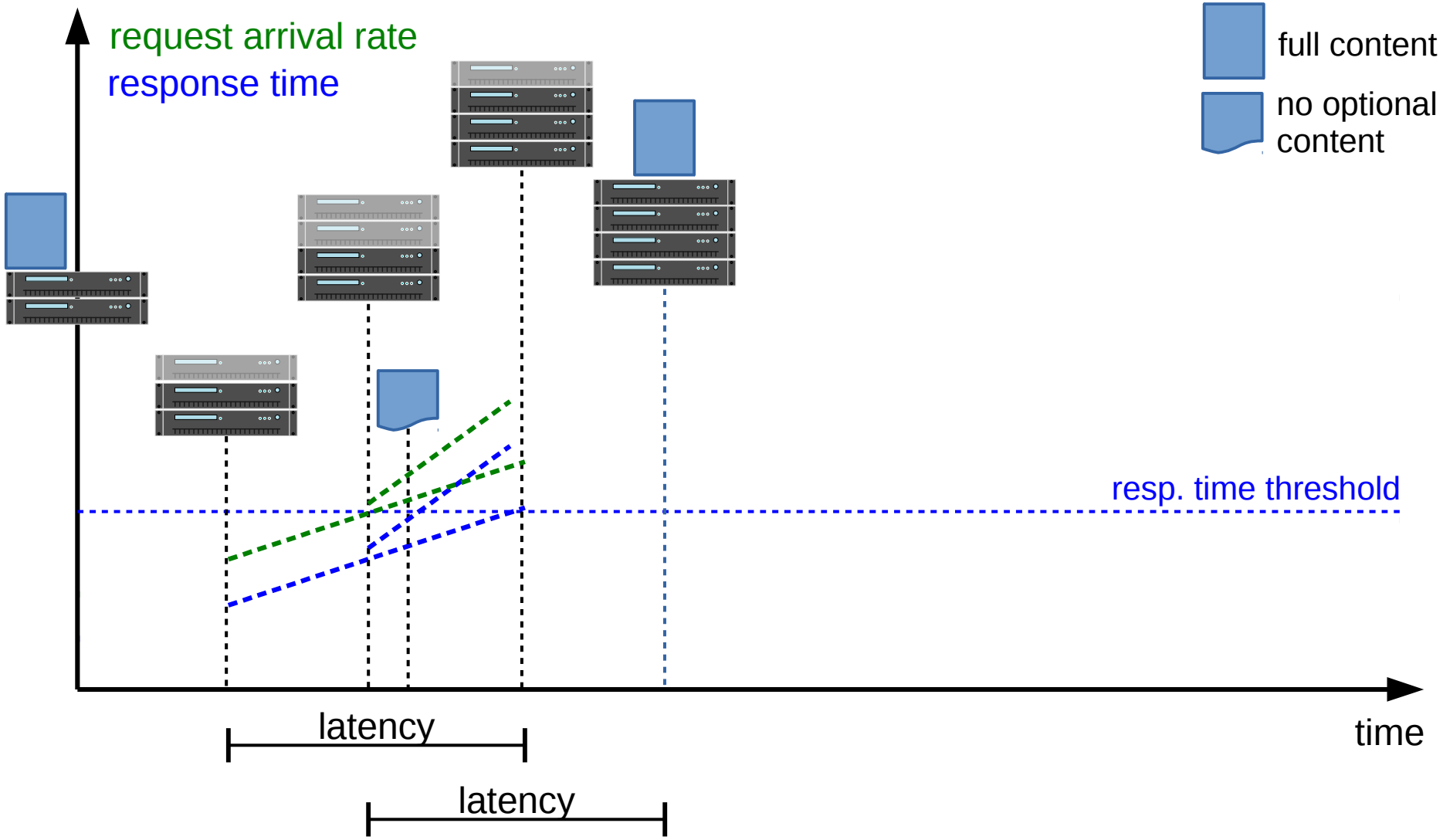
may perform a suboptimal sequence of adaptations

Proactive latency-aware (PLA) adaptation

We can improve the effectiveness of self-adaptation over reactive time-agnostic adaptation by

- explicitly considering the latency of adaptation tactics,
- adapting proactively, and
- potentially allowing concurrent execution of adaptation tactics.

Proactive latency-aware adaptation in RUBiS



Other examples

Cloud computing: VM provisioning times vary across providers. Could trade off latency and cost

Wireless sensor networks: adaptations requiring firmware updates take ~75 seconds

Cyber-physical systems: adaptations may require physical movements; powering sensors up also takes time

Systems with human actuators: decide between fast system actions and slow human-performed actions

Security: considering the consequences of the time attacker observation tactics take

Outline

Adaptation decision problem

Solution approaches

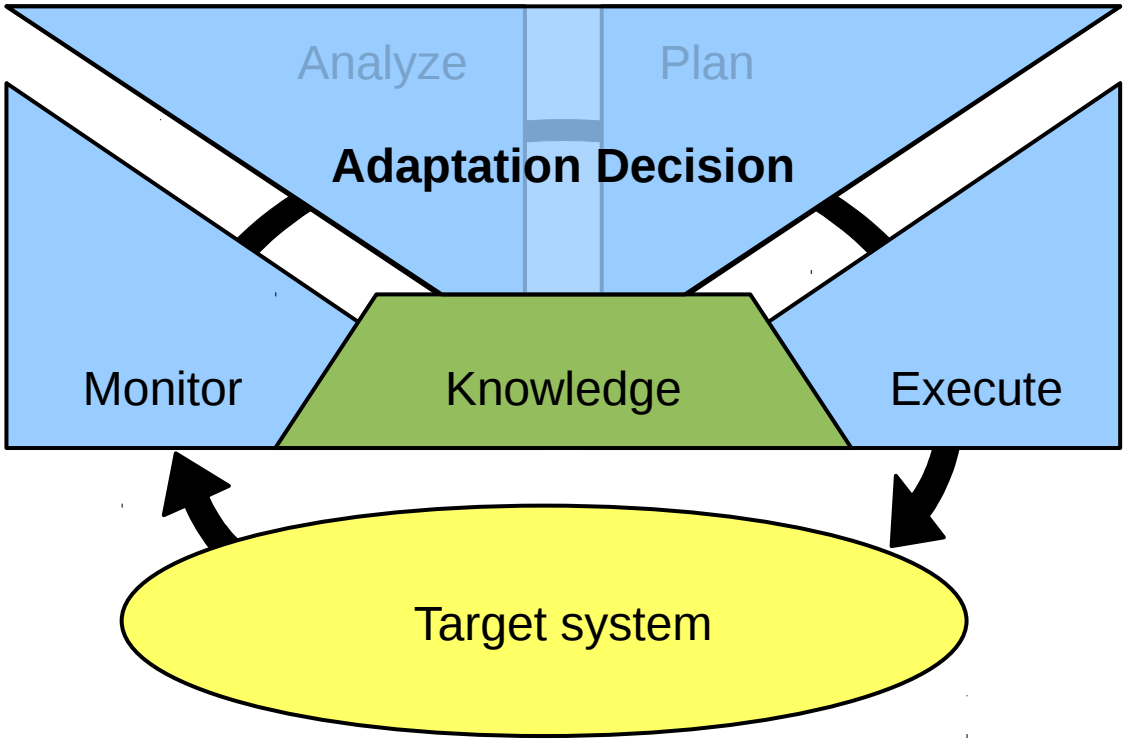
- Probabilistic Model Checking (PLA-PMC)
- Stochastic Dynamic Programming (PLA-SDP)

Other forms of utility

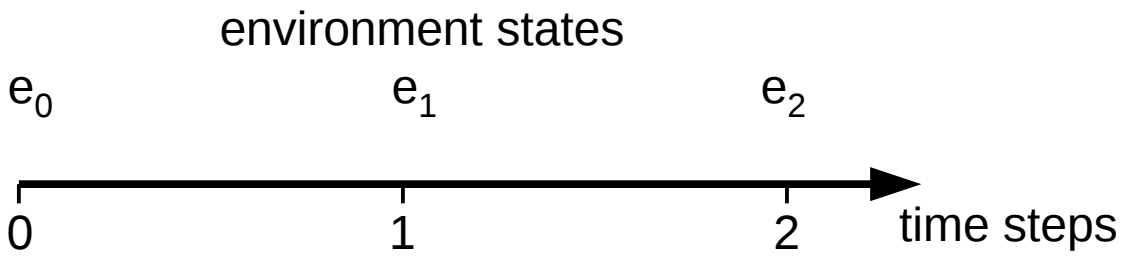
Approximate solution with Cross-Entropy (PLA-CE)

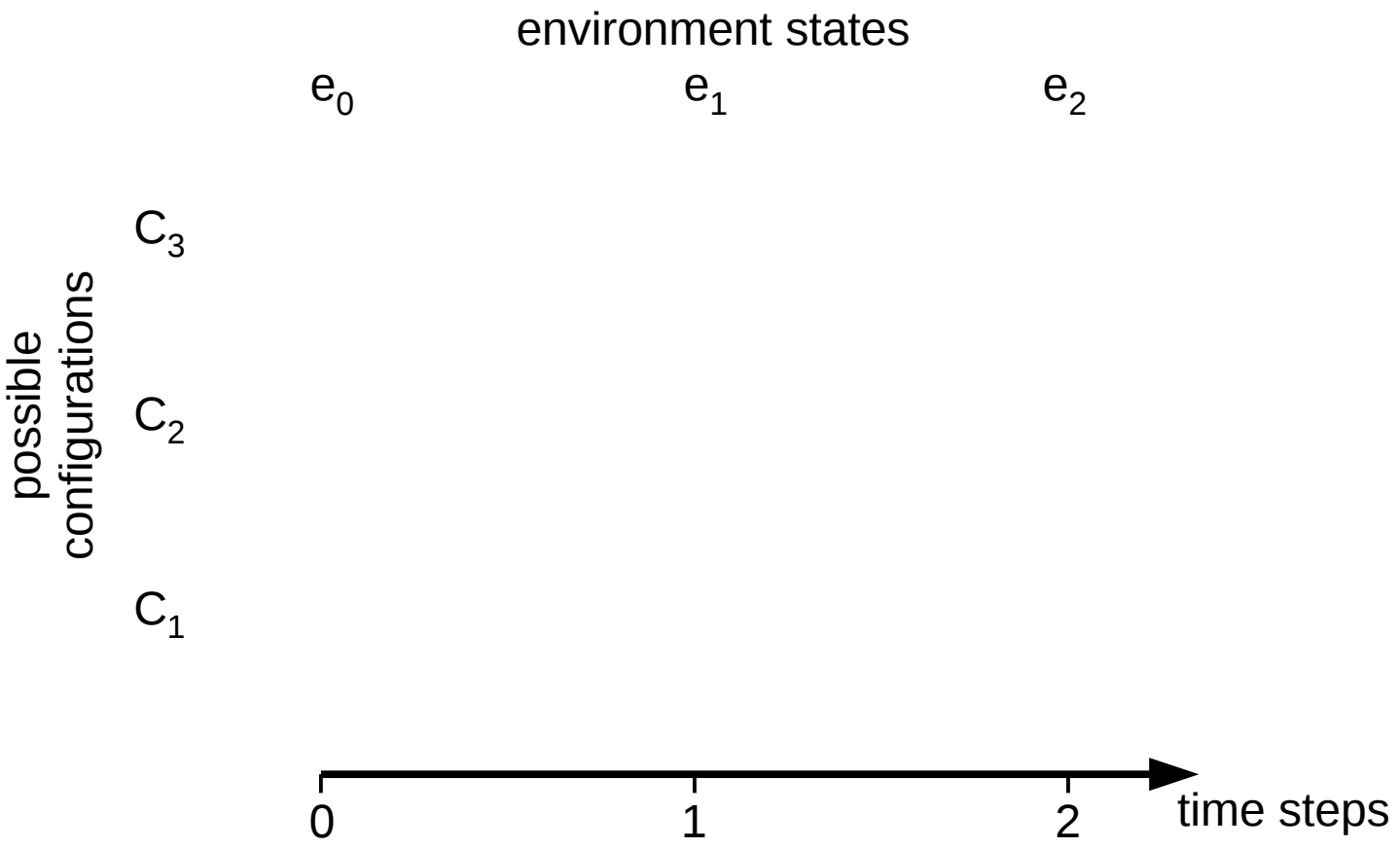
Uncertainty reduction

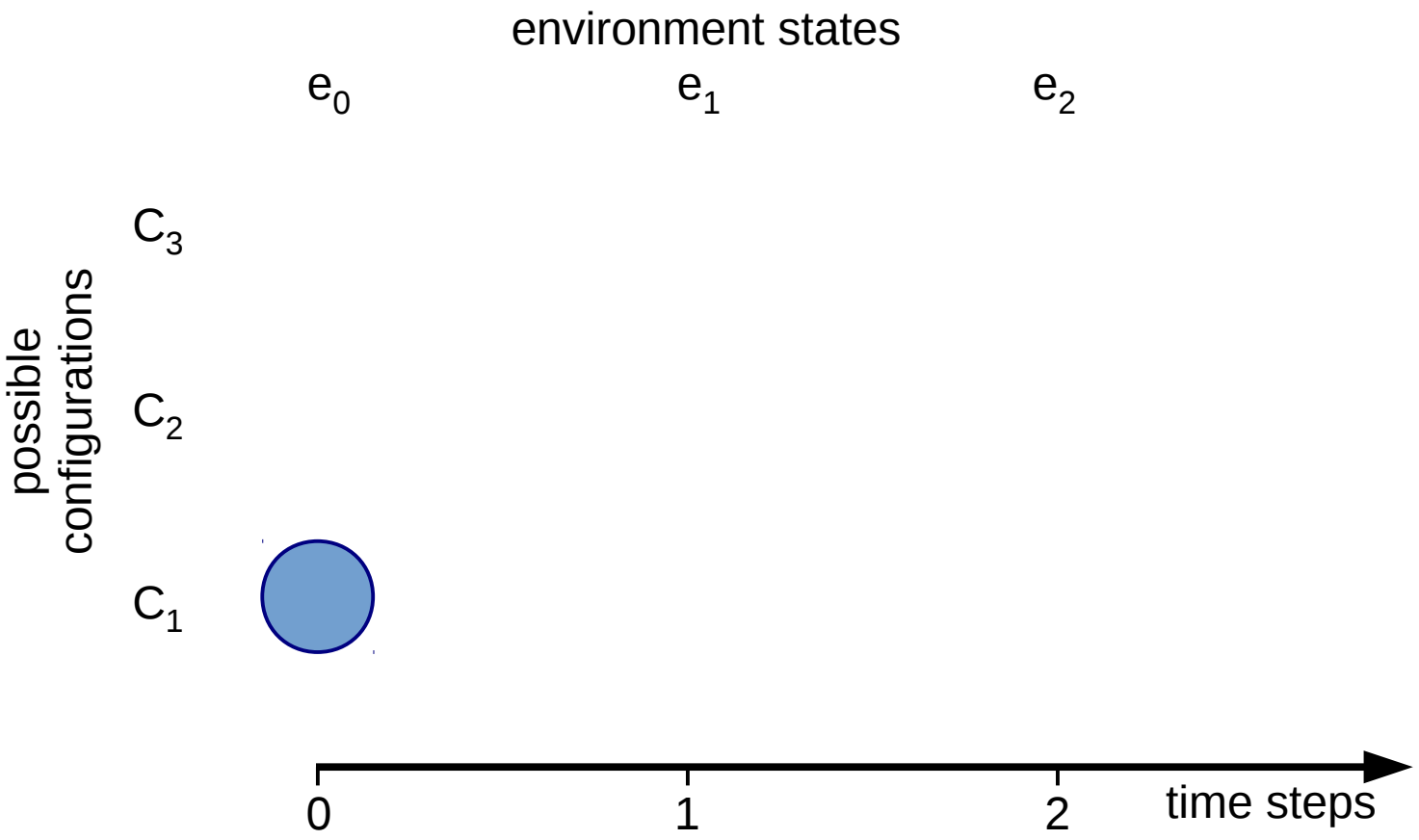
Adaptation decision



Answers the question of what adaptation tactic(s) should be started now to achieve the adaptation goal

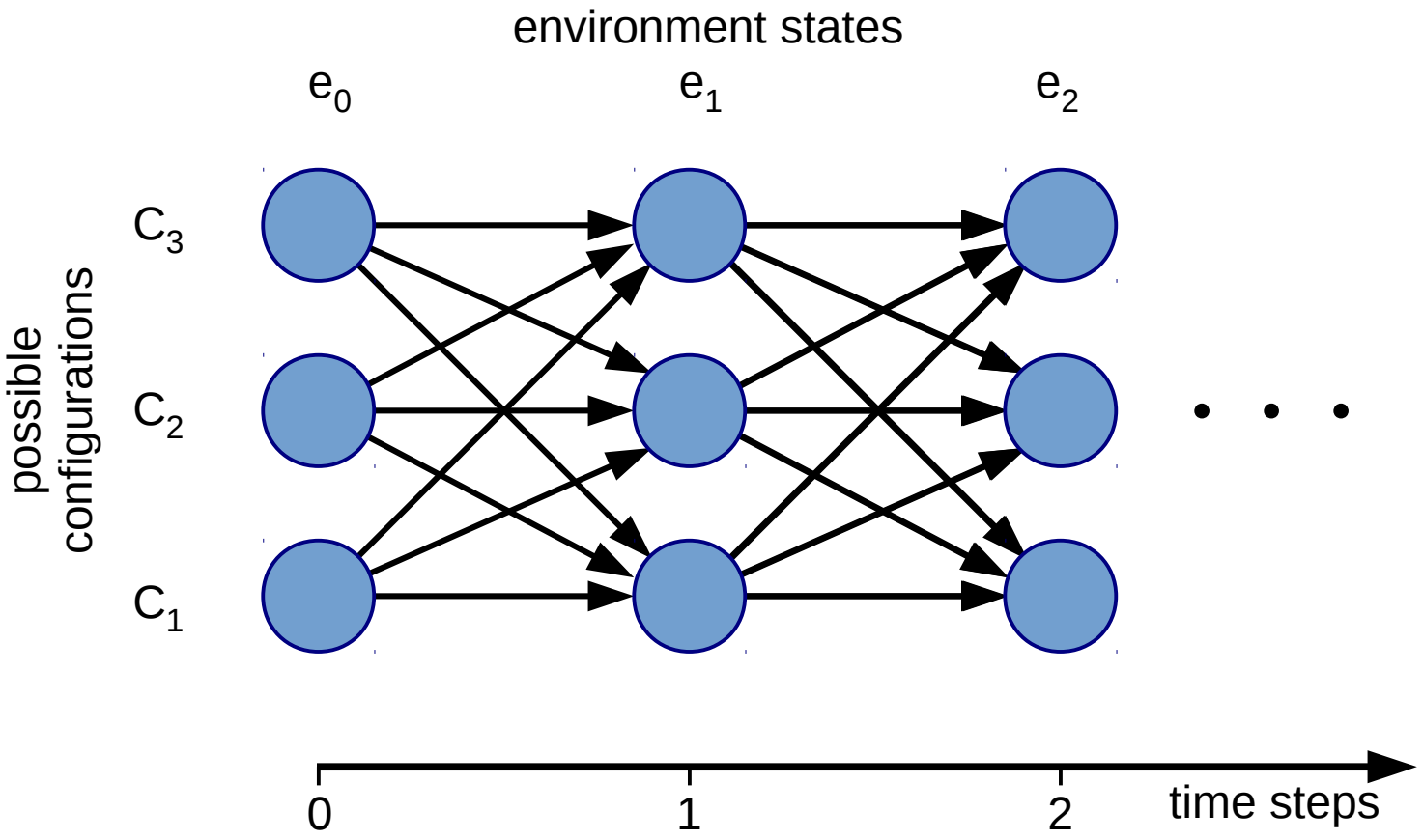






configuration utility at given time

$$u_{i,j} = U(C_i, e_j)$$

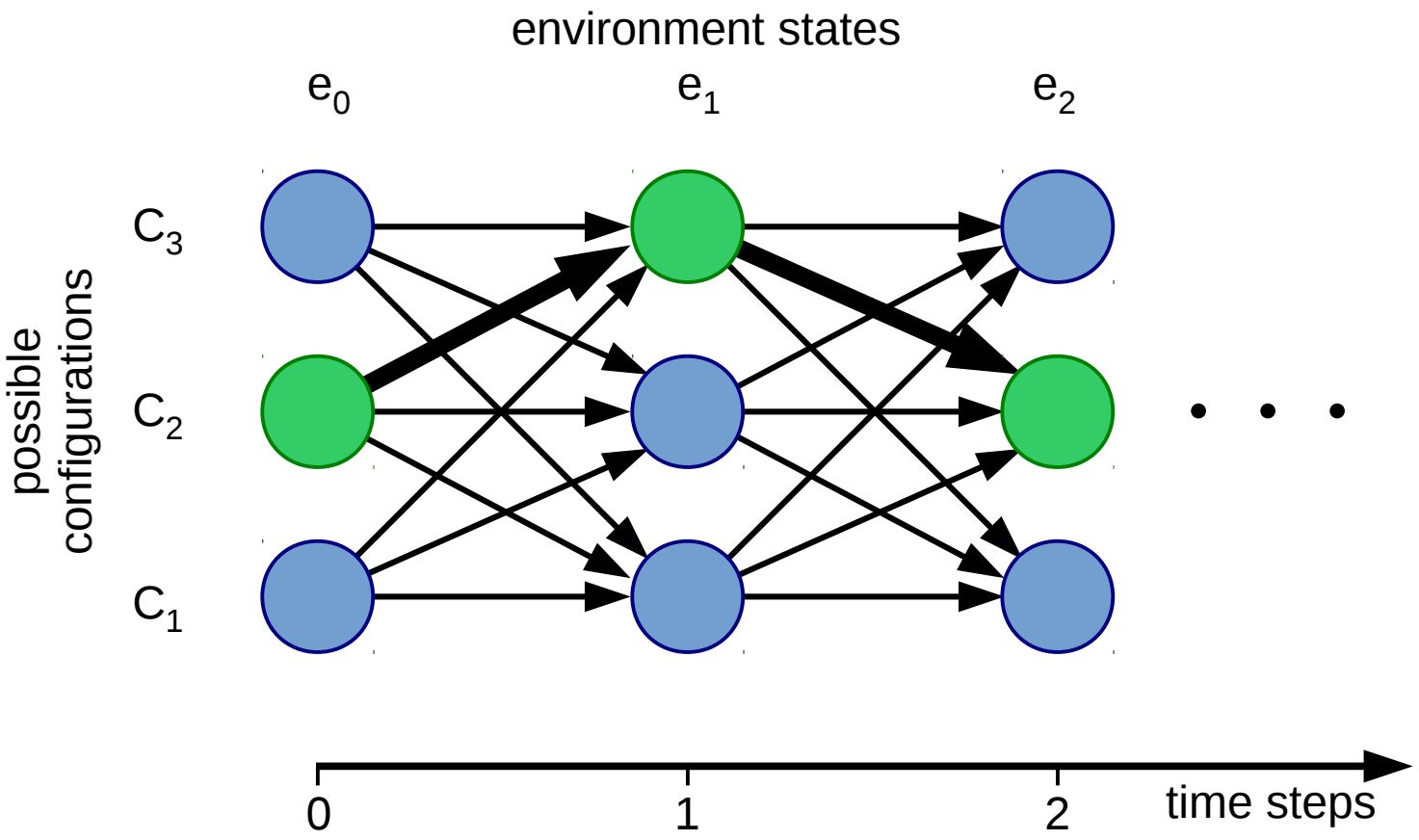


configuration utility at given time

$$u_{i,j} = U(C_i, e_j)$$

objective:

$$max \sum_t U_t$$



configuration utility at given time

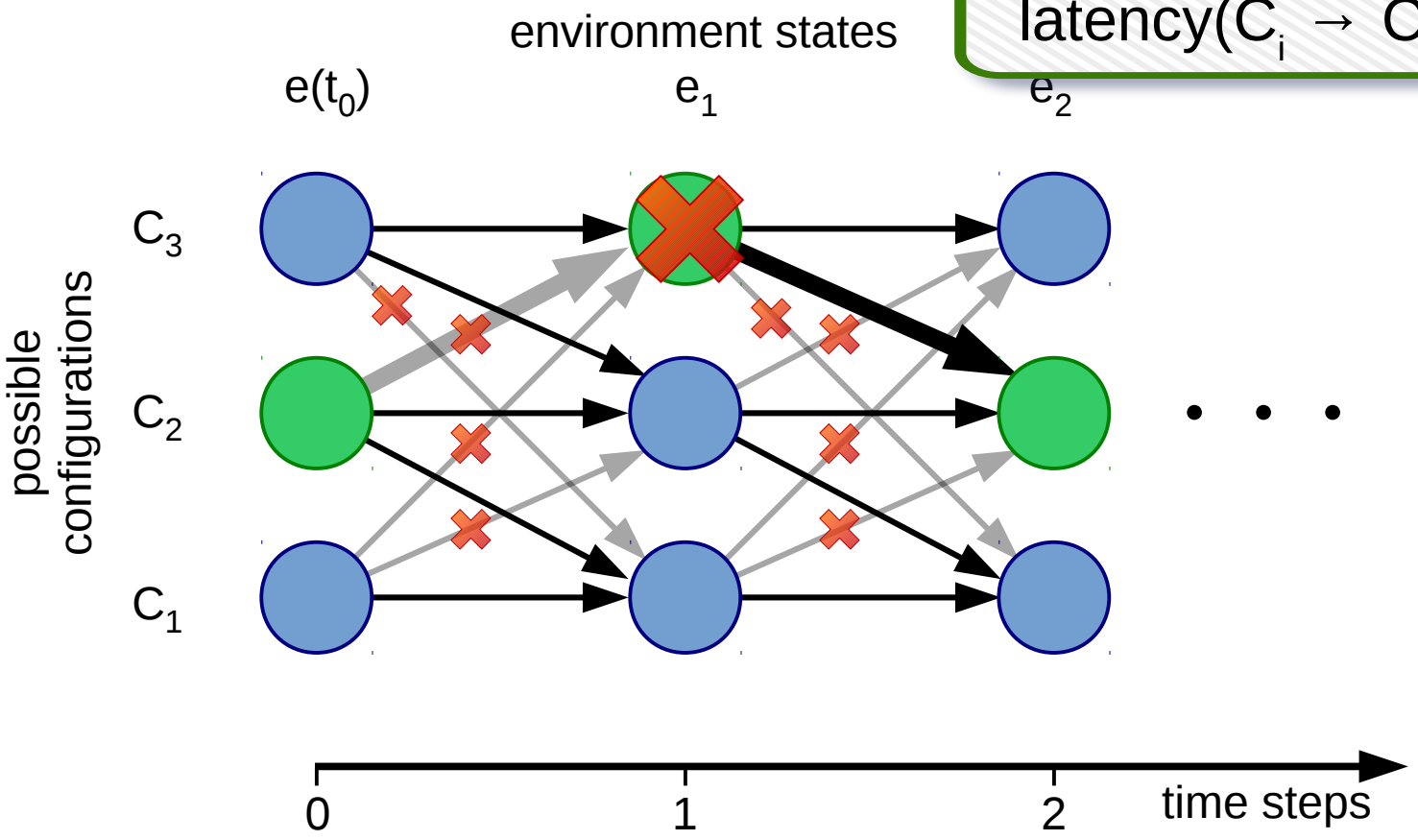
$$u_{i,j} = U(C_i, e_j)$$

objective:

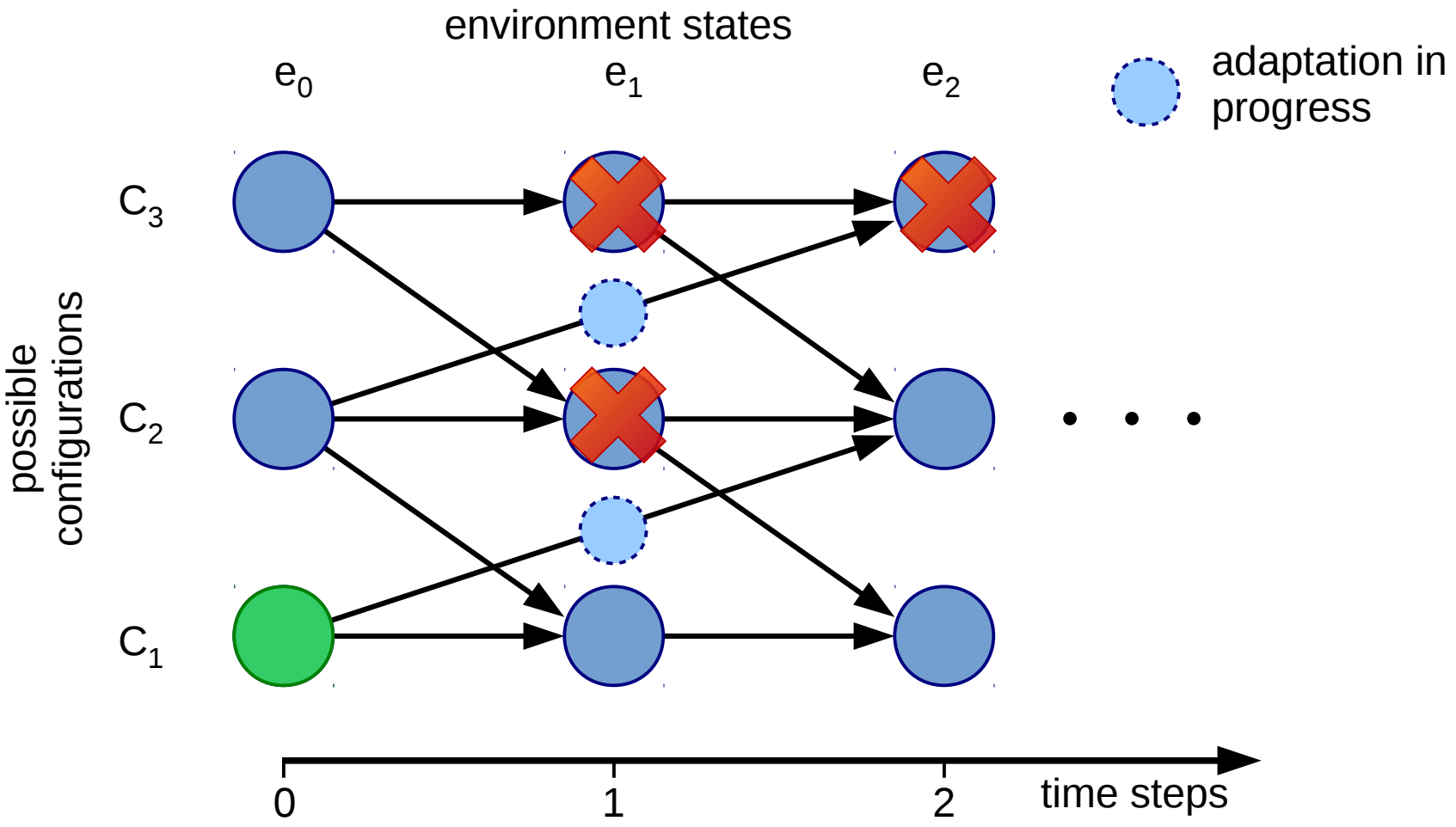
$$max \sum_t U_t$$

Adaptation (in)feasibility due to latency

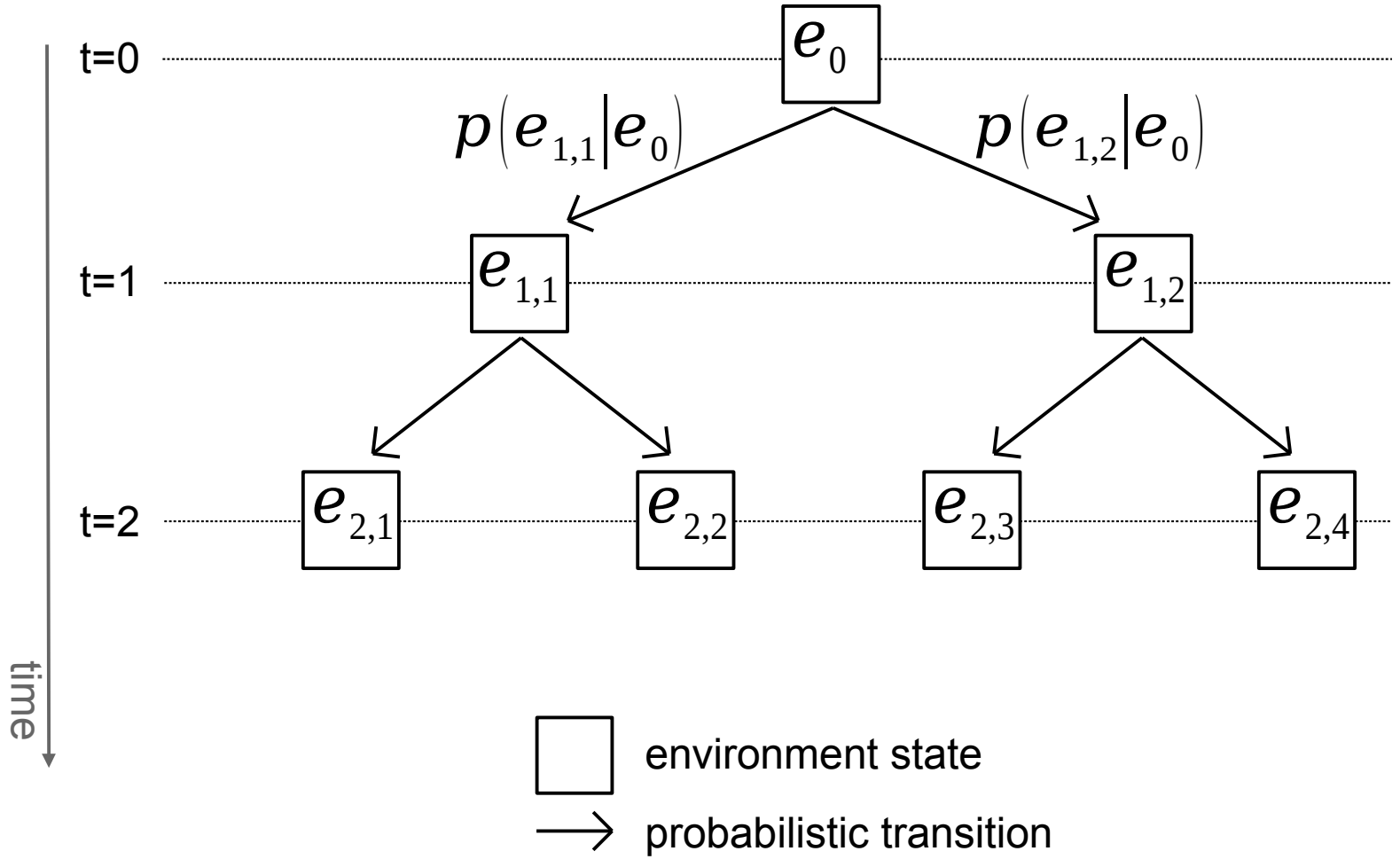
latency($C_i \rightarrow C_{i+1}$) = 2
latency($C_i \rightarrow C_{i-1}$) = 1



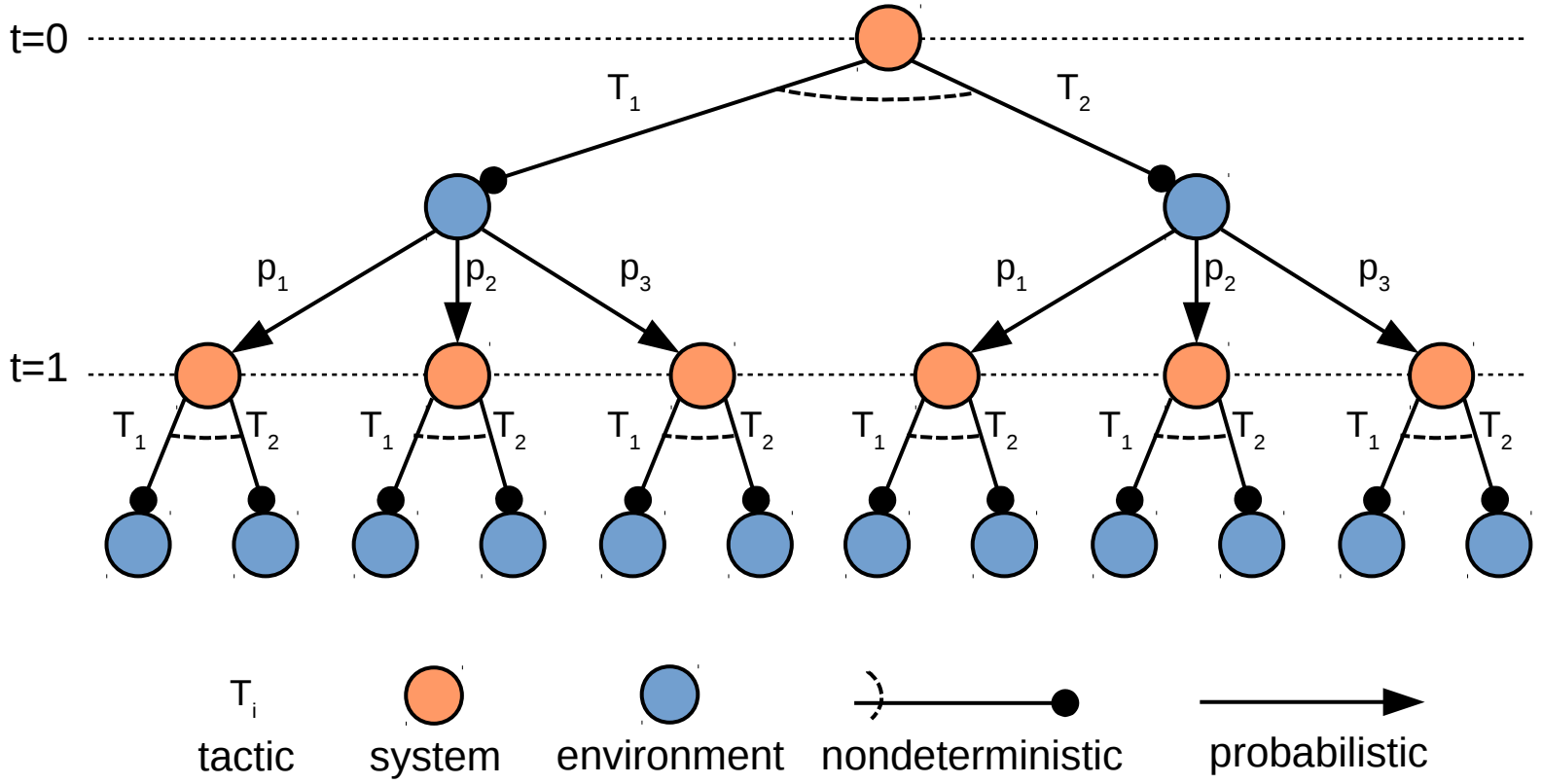
Decisions constrain future decisions



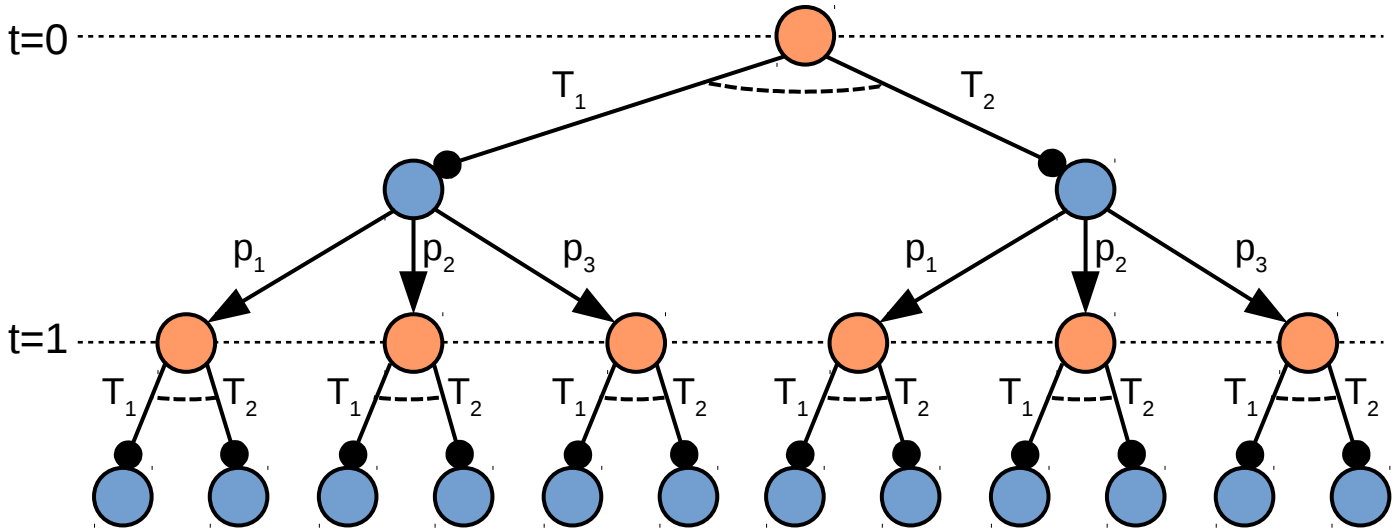
Stochastic environment model

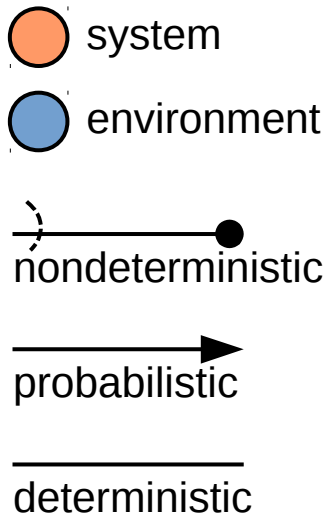


Decision with Markov Decision Processes



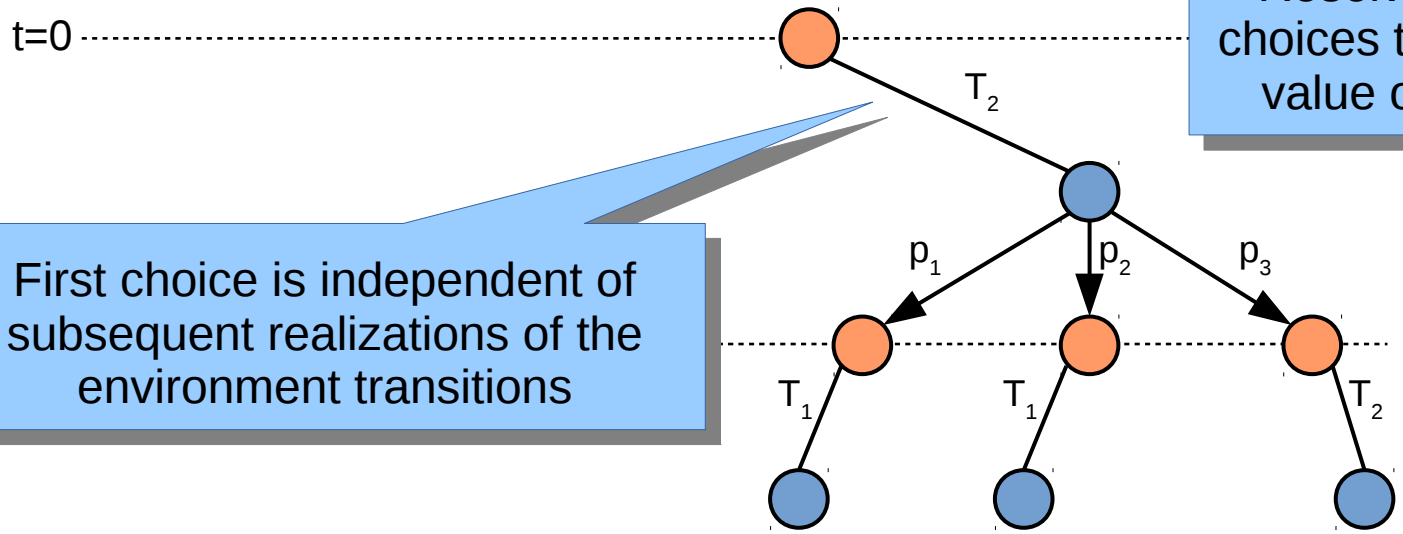
Adaptation decisions are modeled as nondeterministic choices





optimal policy synthesis

Resolves nondeterministic choices to maximize expected value of the utility function



First choice is independent of subsequent realizations of the environment transitions

Challenges

MDPs do not have built-in support for both

- concurrent actions; and
- actions with different latencies

Constructing the explicit representation of the MDP for a self-adaptive system is unwieldy

Need to compute the optimal policy sufficiently fast for run-time decisions

Outline

Adaptation decision problem

Solution approaches

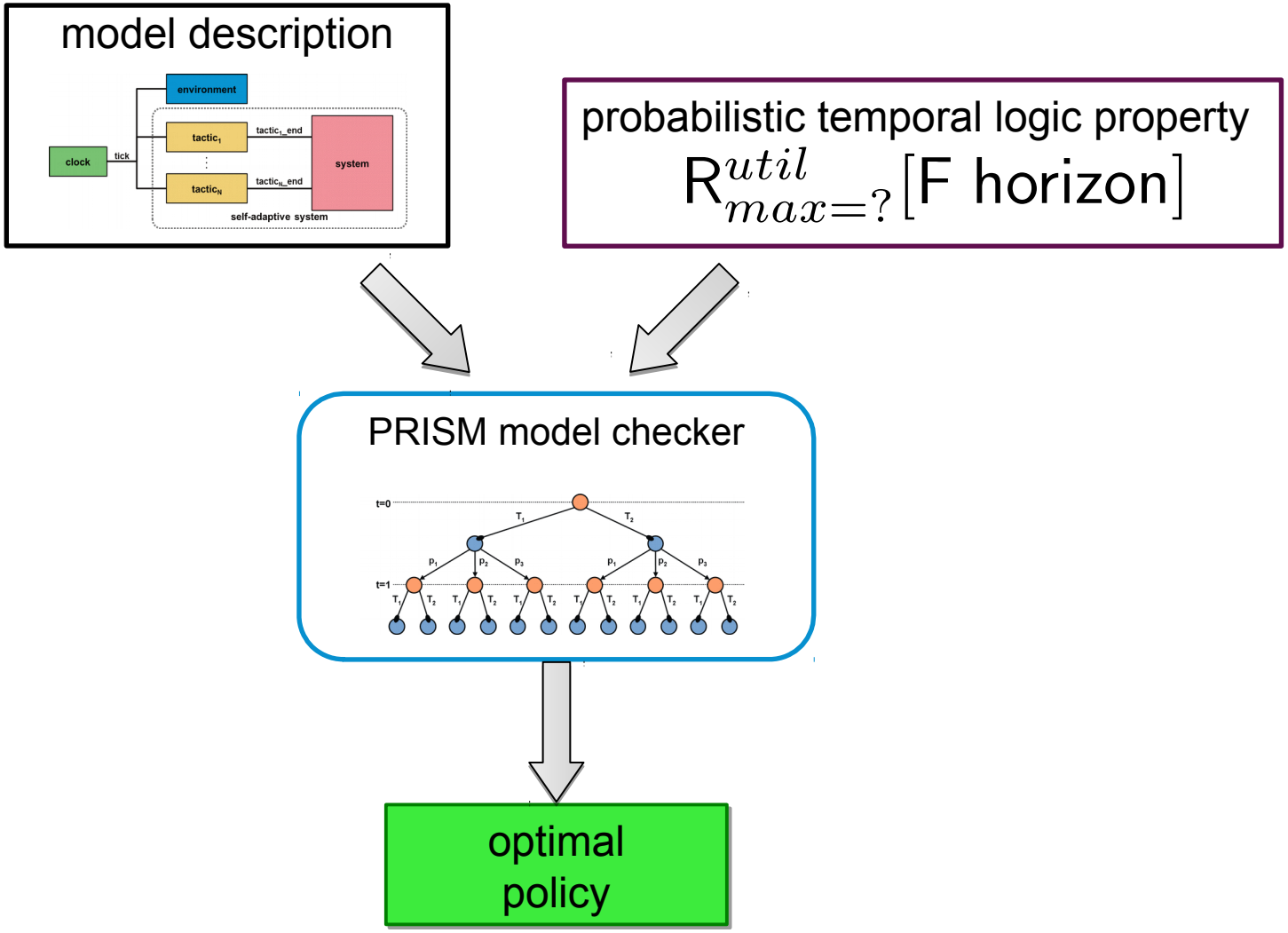
- Probabilistic Model Checking (PLA-PMC)
- Stochastic Dynamic Programming (PLA-SDP)

Other forms of utility

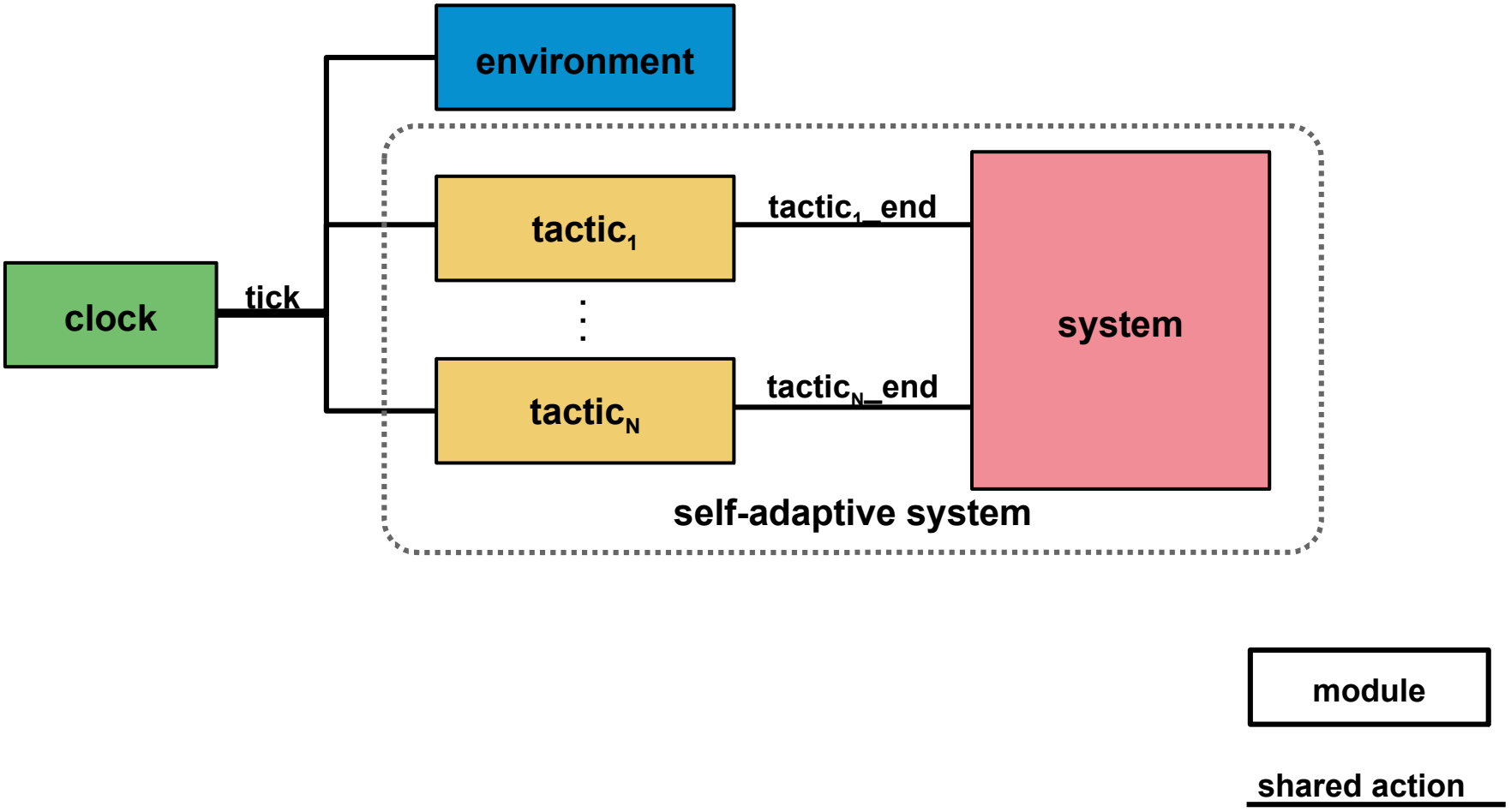
Approximate solution with Cross-Entropy (PLA-CE)

Uncertainty reduction

Probabilistic model checking

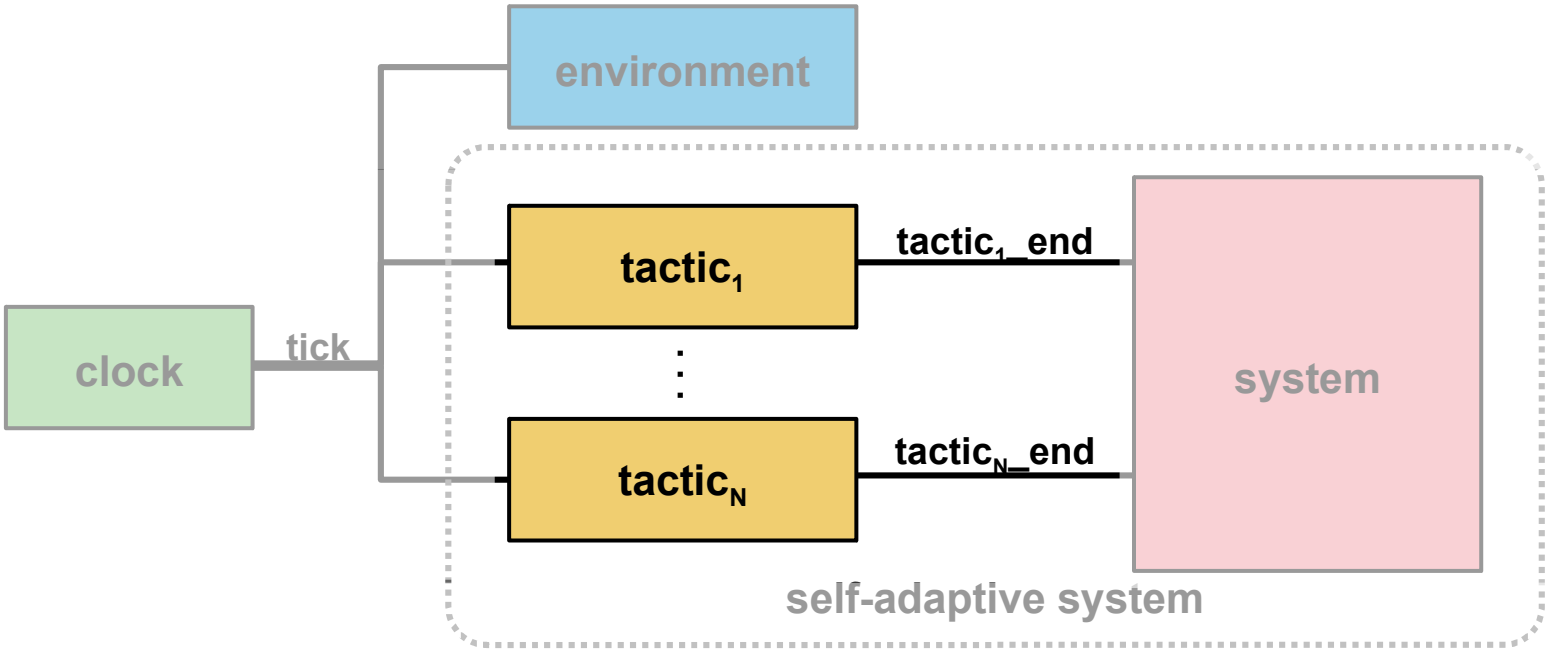


PLA-PMC model*



* Moreno, Cámara, Garlan, and Schmerl. “Proactive self-adaptation under uncertainty: a probabilistic model checking approach.” ESEC/FSE 2015

PLA-PMC model



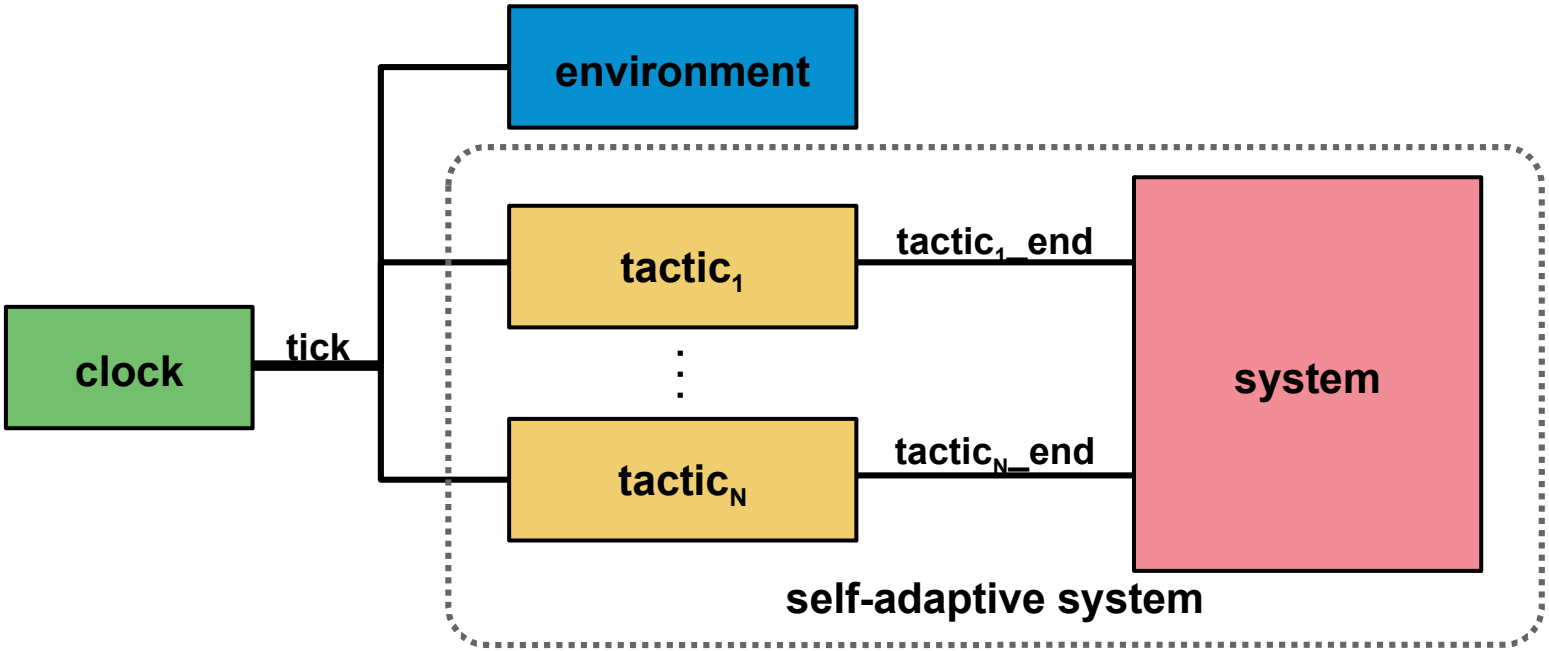
Tactic modules

- applicability conditions
- concurrency compatibility conditions
- non-deterministically starts (or not) the tactic
- keeps track of its own progress (latency)

module

shared action

PLA-PMC model



Key Idea

Leave the decision to start a tactic underspecified through nondeterminism, and have the model checker resolve the nondeterministic choices

module

shared action

PLA-SDP*

Builds most of the MDP off-line—**avoids run-time overhead of MDP construction**

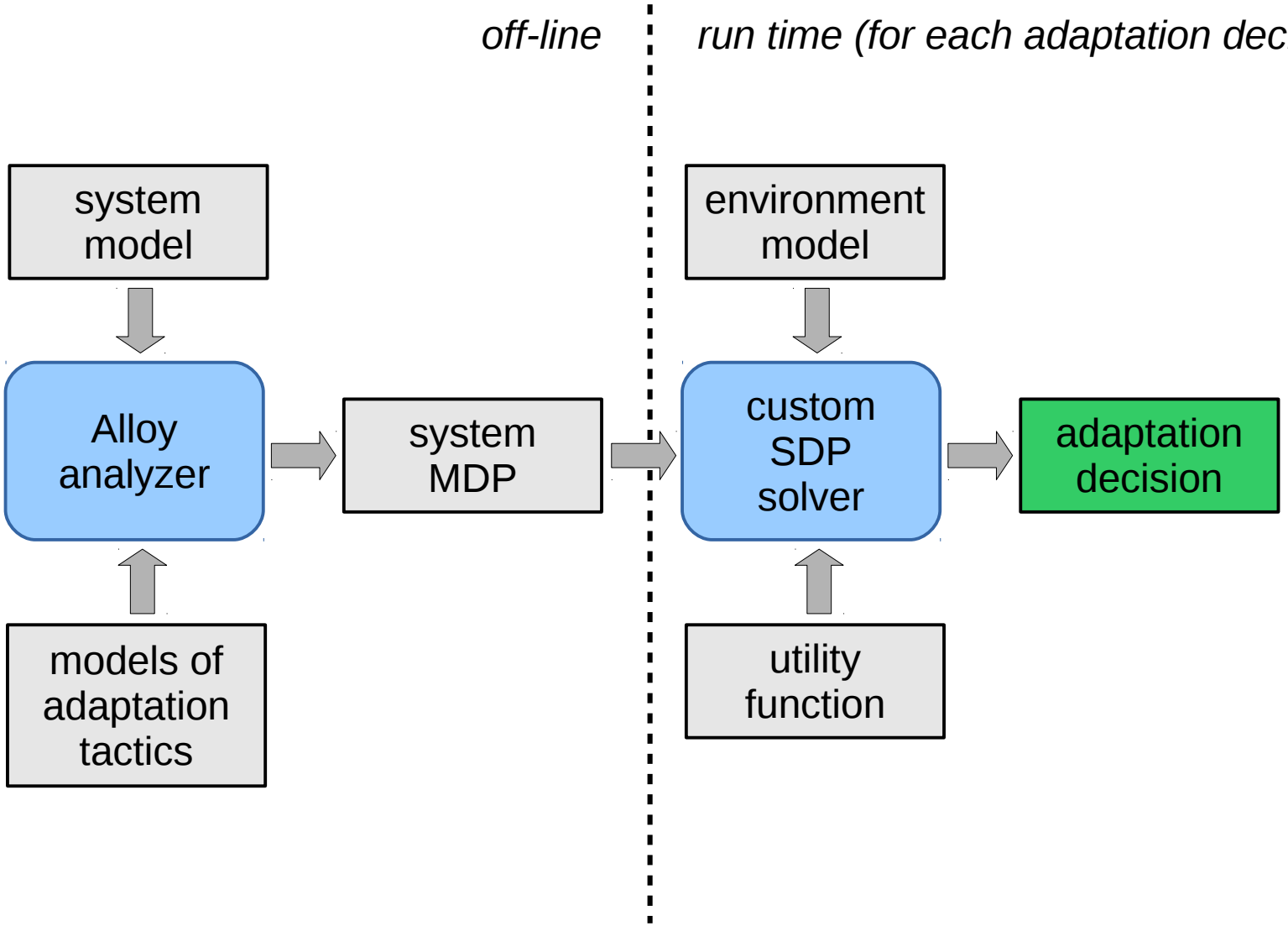
Has custom **stochastic dynamic programming** algorithm to solve the MDP updating environment part at run time

Much faster decisions

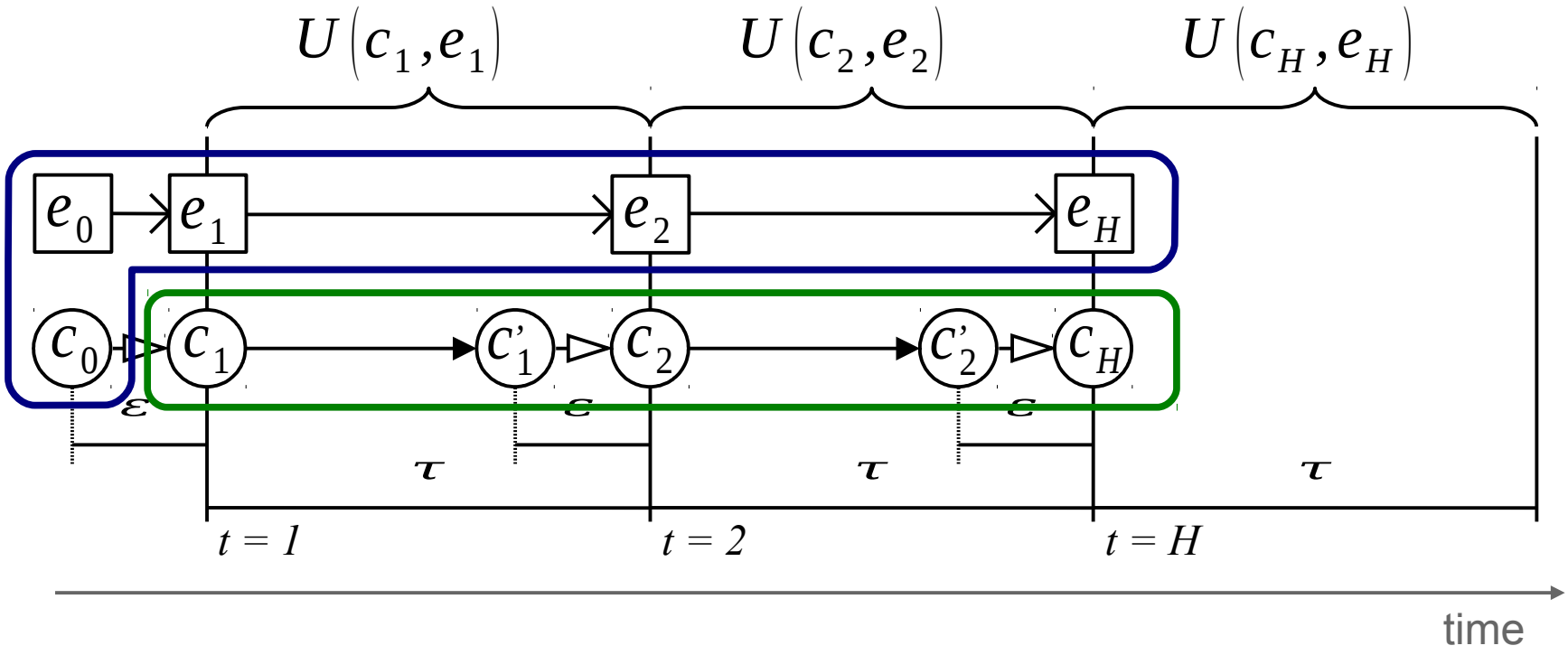
Does not compromise optimality—computes same solution as PLA-PMC

* Moreno, Cámara, Garlan, and Schmerl. “Efficient decision-making under uncertainty for proactive self-adaptation.” ICAC 2016

PLA-SDP



Pattern of adaptation transitions



system state

immediate transition

delayed transition

environment state

environment transition

$U(c, e)$ utility function

Reachability predicates

Define the feasible transitions in the system MDP

$R^I(c, c')$ — c' can be reached from c with an immediate transition

$R^D(c, c')$ — c' can be reached from c with a delayed transition

$R^T(c, c') \equiv \exists c'' : R^D(c, c'') \wedge R^I(c'', c')$

c' can be reached from c with a delayed transition followed by an immediate transition

Defining them requires considering all the possible combinations of adaptation tactics, their phasing, and the different system states.

Stochastic dynamic programming solution

$$v^H(c,e) = \hat{U}(c,e), \quad \forall c \in C, e \in E_H$$

$$v^t(c,e) = \hat{U}(c,e) + \max_{c' \in C^T(c)} \sum_{e' \in E_{t+1}} p(e'|e) v^{t+1}(c',e'),$$

$$\forall c \in C, e \in E_t, t = H-1, \ldots, 1$$

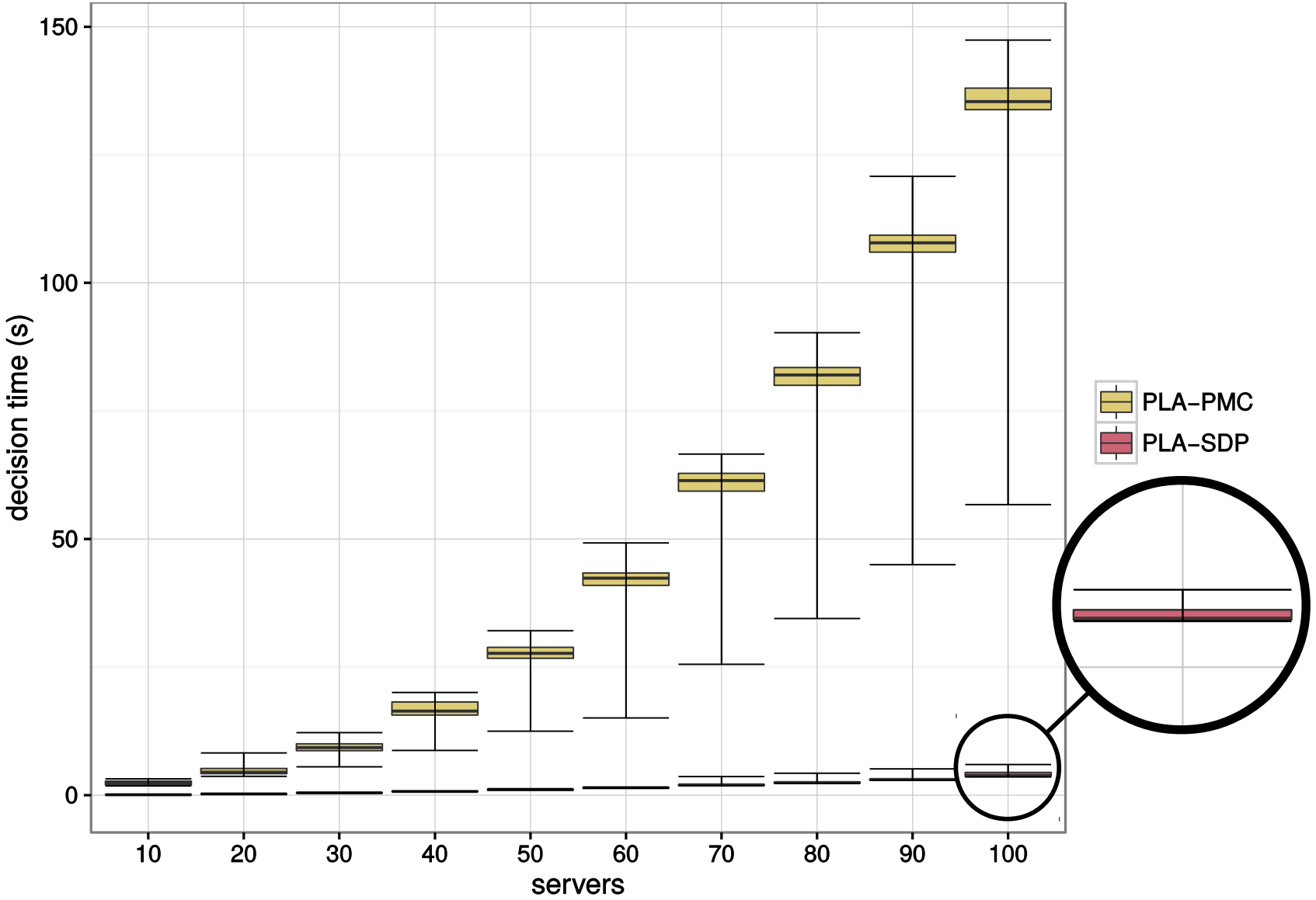
$$C^* = \arg\max_{c}$$

environment

Iterates only over the reachable states

$$C^T(c) = \{c' \in C : R^T(c,c')\}$$

Adaptation decision speedup

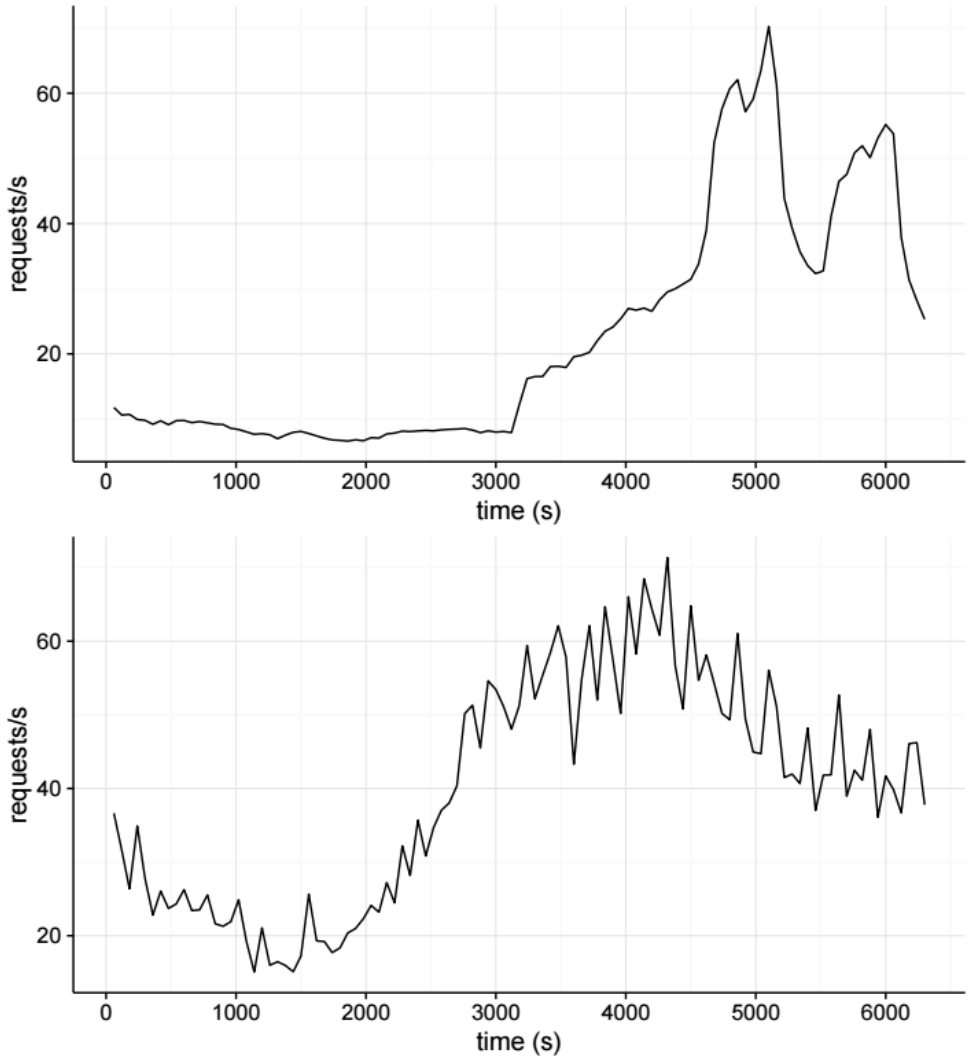


Effectiveness Improvement with PLA

Approach: compare effectiveness of PLA with a feed-forward (FF) self-adaptation approach that lacks proactivity and latency-awareness

Request traces for RUBiS

Effectiveness improvement

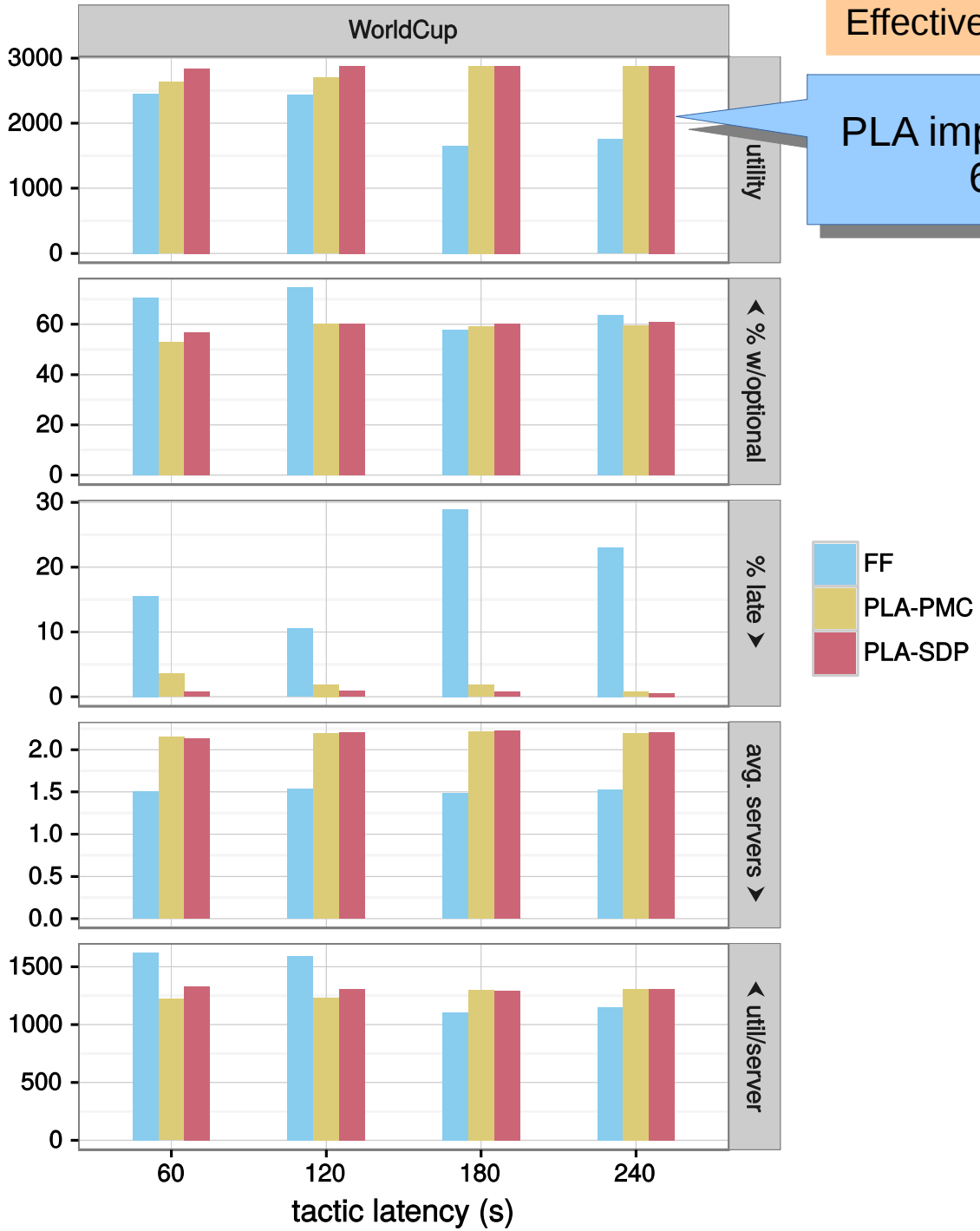


World Cup ‘98

ClarkNet

The Internet Traffic Archive http://ita.ee.lbl.gov/

Effectiveness improvement



PLA improvement: 65%

Outline

Adaptation decision problem

Solution approaches

- Probabilistic Model Checking (PLA-PMC)
- Stochastic Dynamic Programming (PLA-SDP)

Other forms of utility

Approximate solution with Cross-Entropy (PLA-CE)

Uncertainty reduction

DART

Team of unmanned air vehicles (UAVs)

Mission: fly predefined route, finding as many targets as possible, avoiding threats

Uncertain environment: location of targets and threats is unknown, sensed with uncertainty as the mission is executed

Adaptation tactics:

- altitude up/down (slow tactic)
- change formation tight/loose (fast tactic)

Adaptation goal:

- survive the mission with probability >= 90%
- maximize number of targets detected

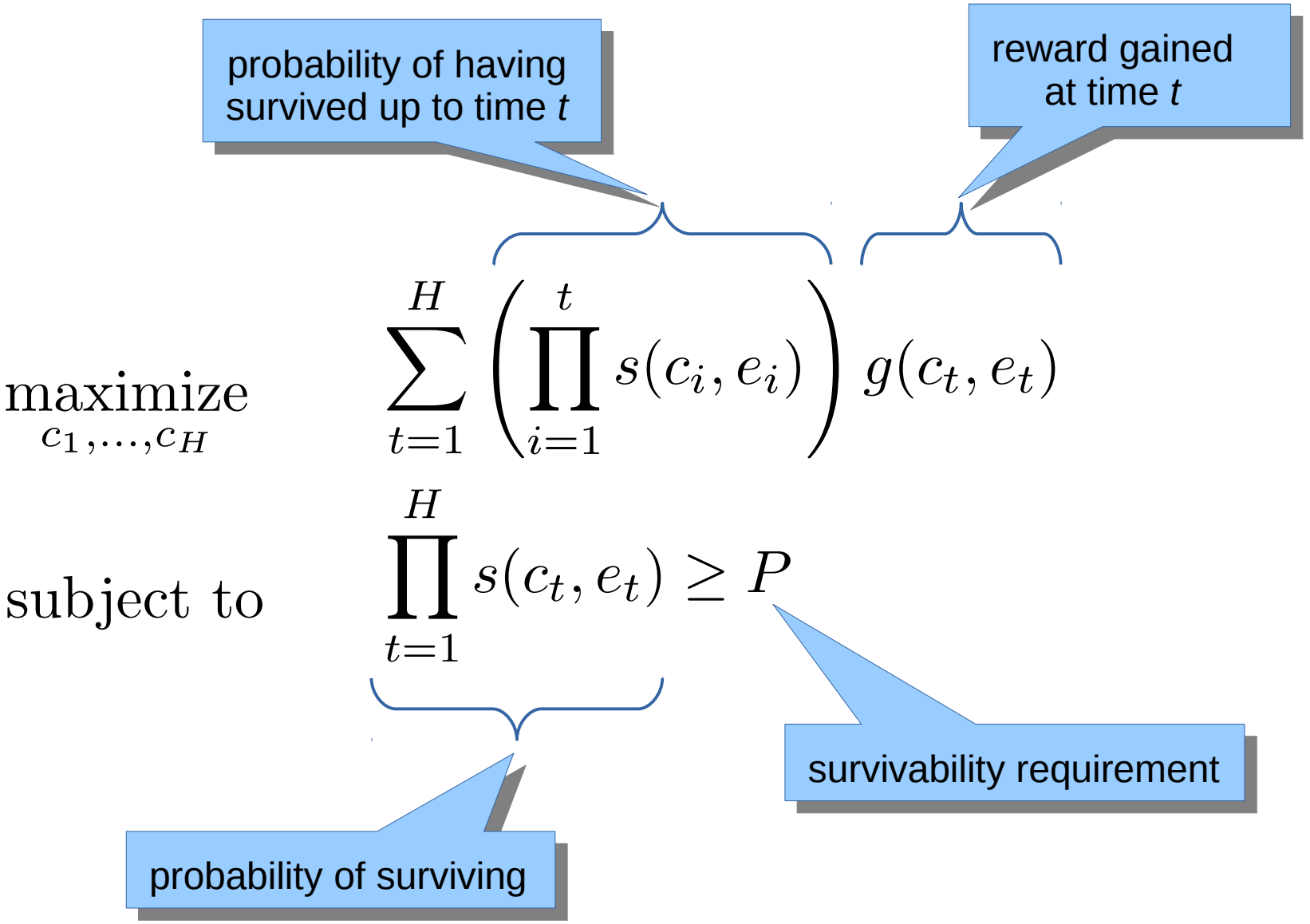
Different form of utility

The team of drones can get reward by detecting targets as long as it is not destroyed

Adaptation affects both reward and survivability

Reward maximization is not sufficient

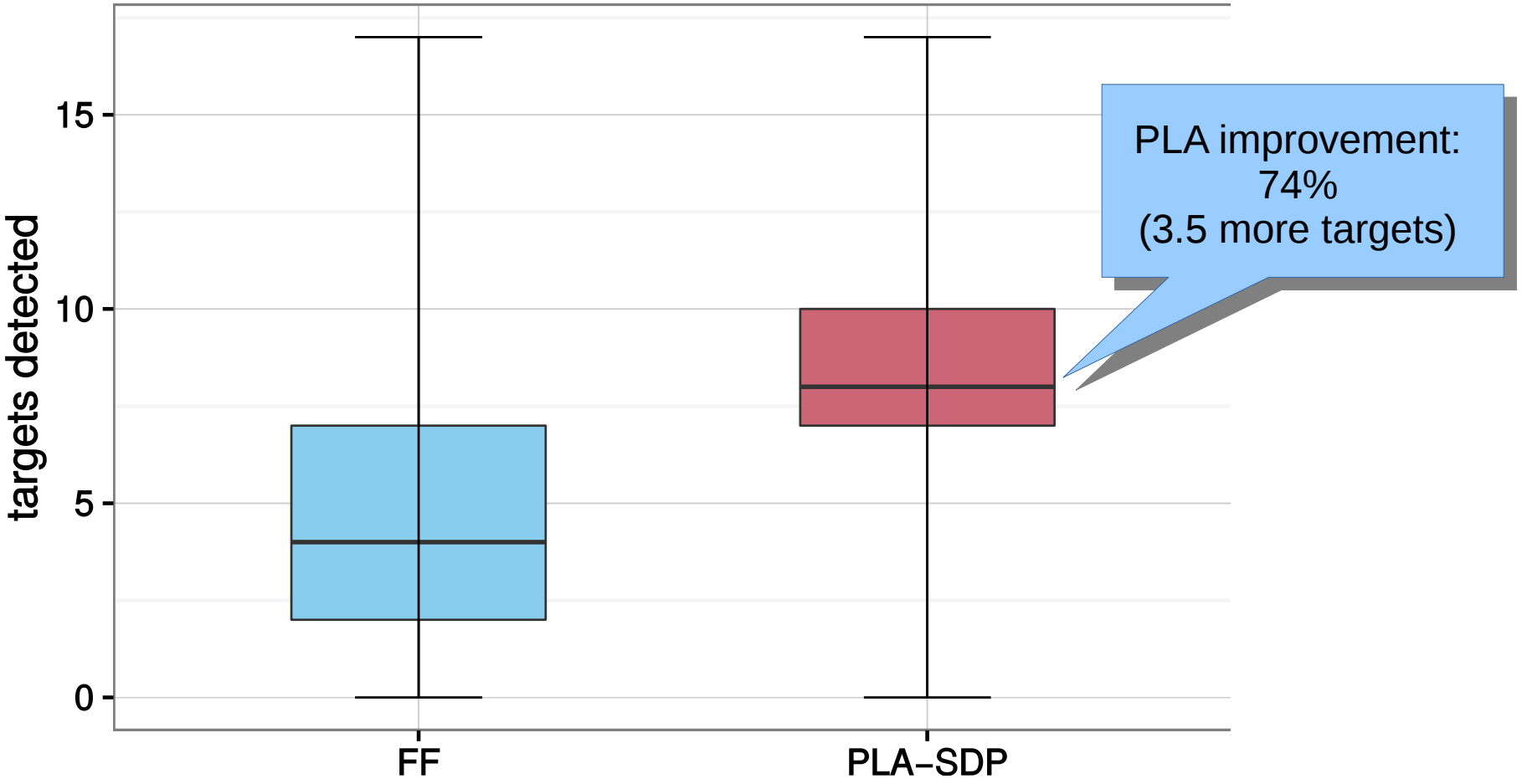
- reward and survivability are not comparable



* Moreno, Cámara, Garlan, and Schmerl. “Flexible and efficient decision-making for proactive latency-aware self-adaptation.” ACM TAAS (to appear)

DART

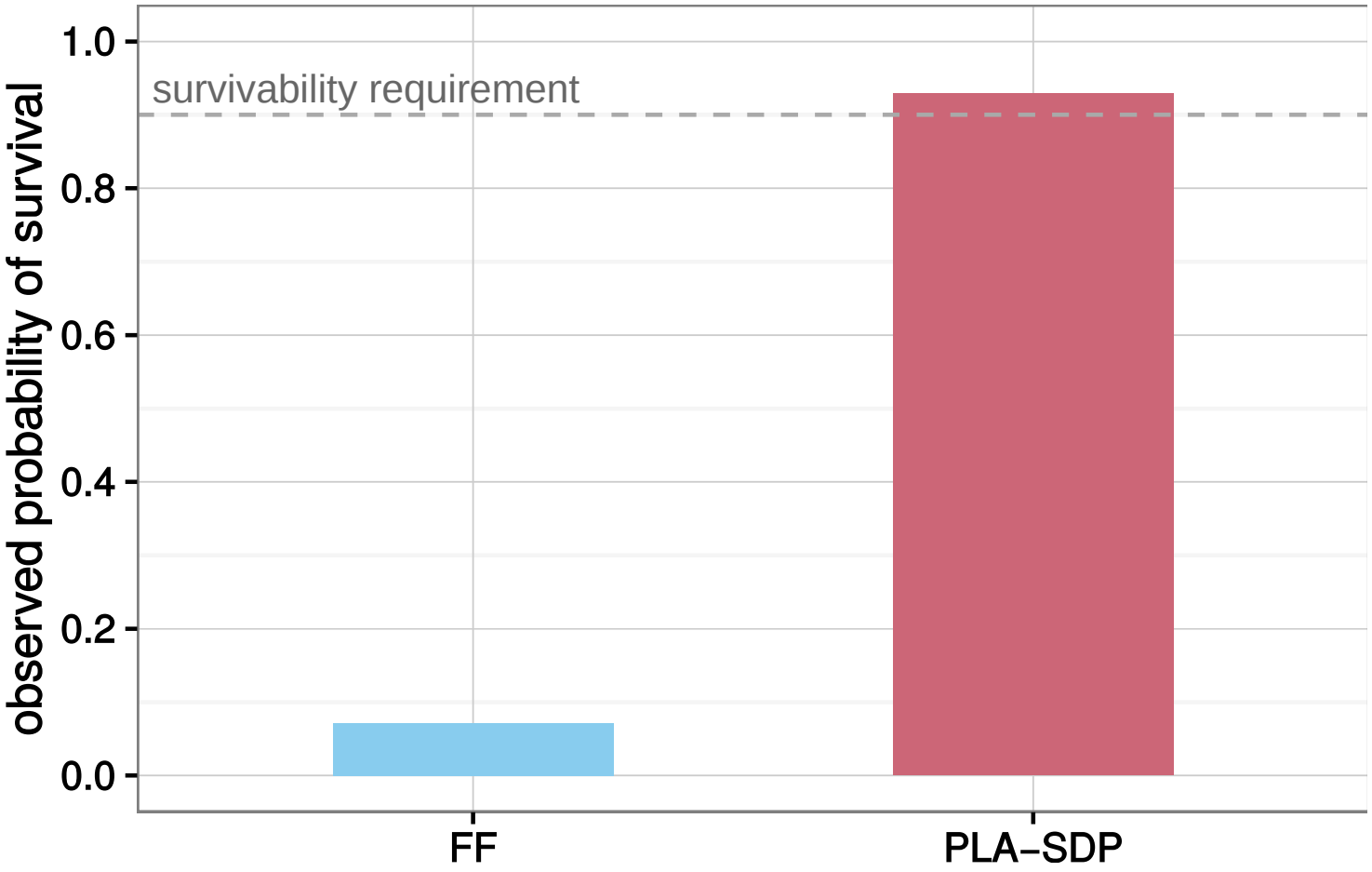
Effectiveness improvement



5000 simulated missions, 20 targets, 7 threats

DART

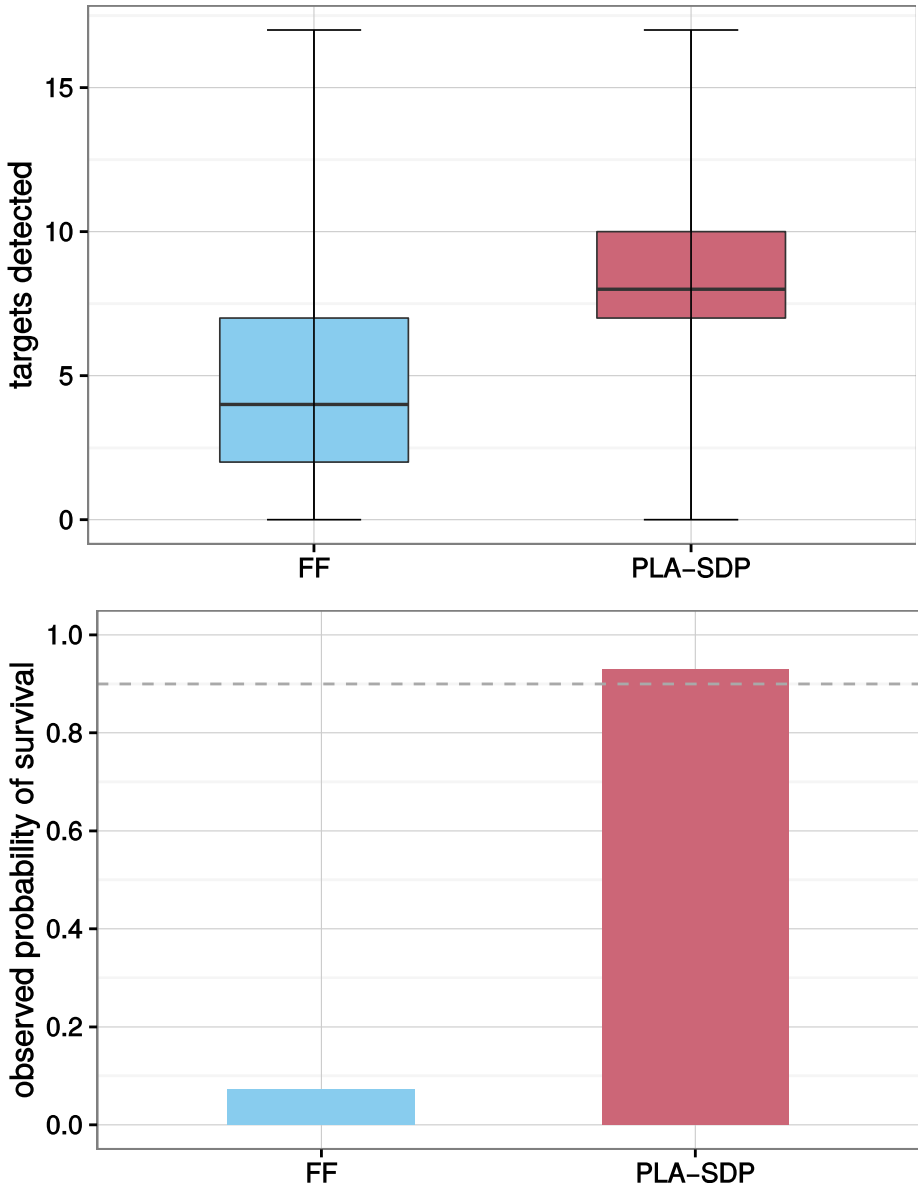
Effectiveness improvement



5000 simulated missions, 20 targets, 7 threats

DART

Effectiveness improvement



5000 simulated missions, 20 targets, 7 threats

Outline

Adaptation decision problem

Solution approaches

- Probabilistic Model Checking (PLA-PMC)
- Stochastic Dynamic Programming (PLA-SDP)

Other forms of utility

Approximate solution with Cross-Entropy (PLA-CE)

Uncertainty reduction

Cross-entropy method for optimization

The CE method is an adaptive sampling procedure for estimating the probability of rare events [Rubinstein 1997]*

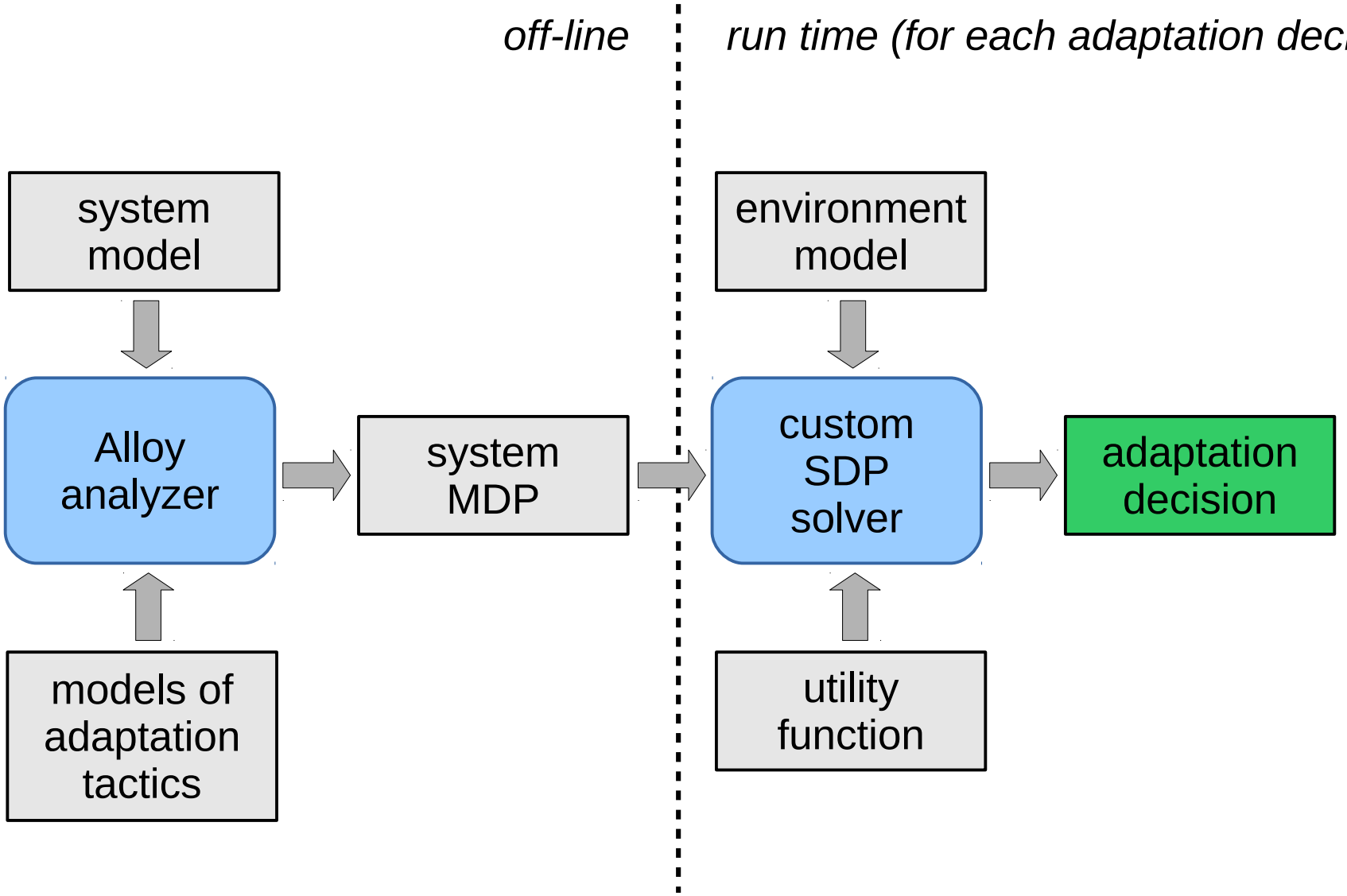
It can also be used to solve optimization problems

- Finding the optimal solution in a random search is a rare event
- In each iteration CE changes the sampling distribution using cross-entropy minimization
- Makes finding the optimal solution (the rare event) more likely

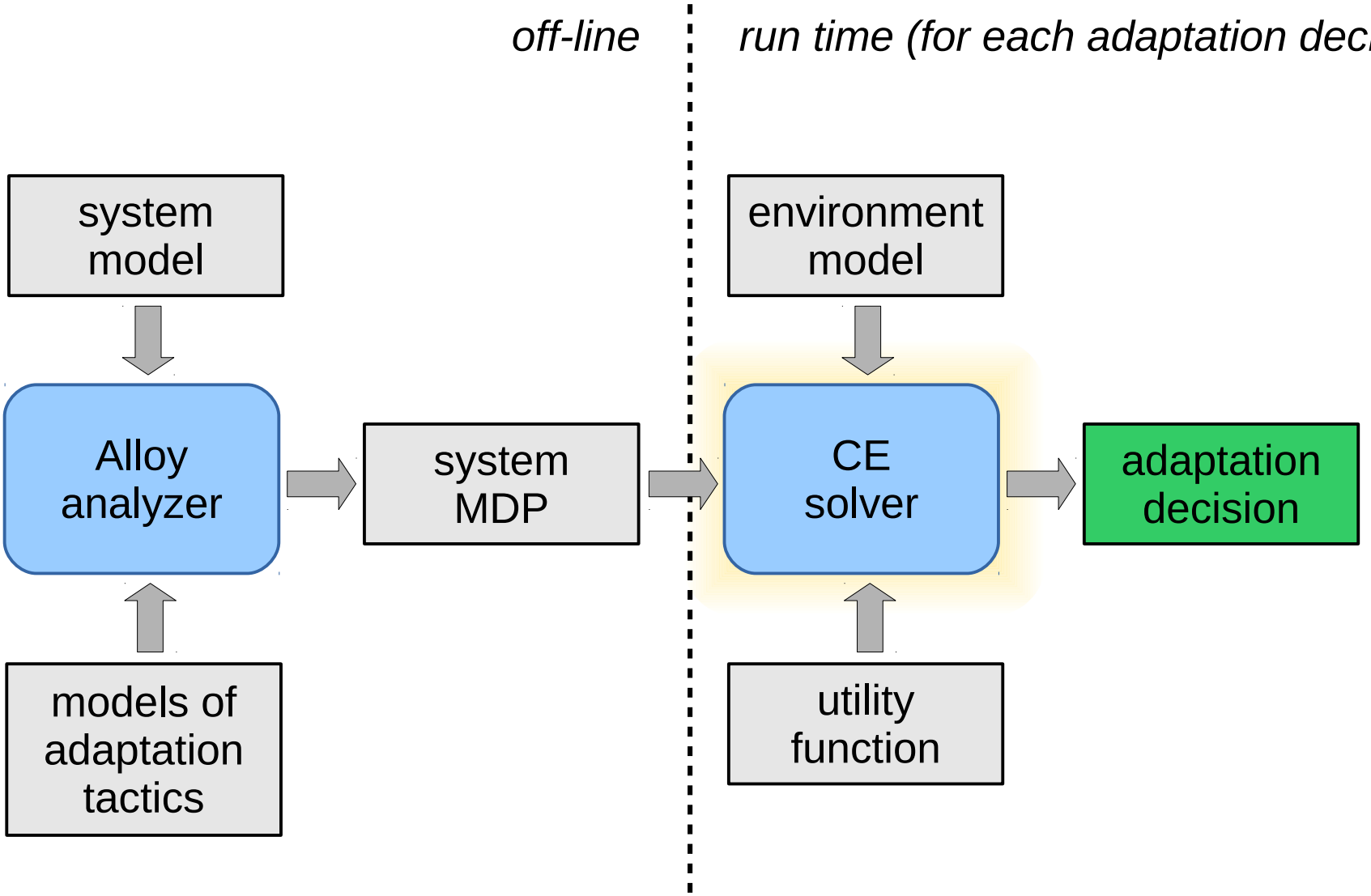
PLA-CE uses the cross-entropy method to find an approximate solution to the adaptation decision problem.

* Rubinstein, “Optimization of computer simulation models with rare events,” European Journal of Operational Research, 1997

PLA-SDP



PLA with Cross-Entropy Method: PLA-CE*



* Moreno, Chaki, Strichman, and Vaisman. “Decision-making with cross-entropy for self-adaptation.” SEAMS 2017

Cross-entropy method algorithm

1: **while** (!converged and !timeout) **do**
2: **for** $j = 1..N$ **do**
3: **for** $i = 1..n$ **do**
4: choose value for v_i randomly according to F_i;
5: **end for**
6: Score solution;
7: Update *BestSolution* if ap
8: **end for**
9: Choose $\lceil \rho * N \rceil$ best solutions to form distribution F'_i;
10: $F_i = \alpha F'_i + (1 - \alpha) F_i$;
11: **end while**
12: **return** *BestSolution*;

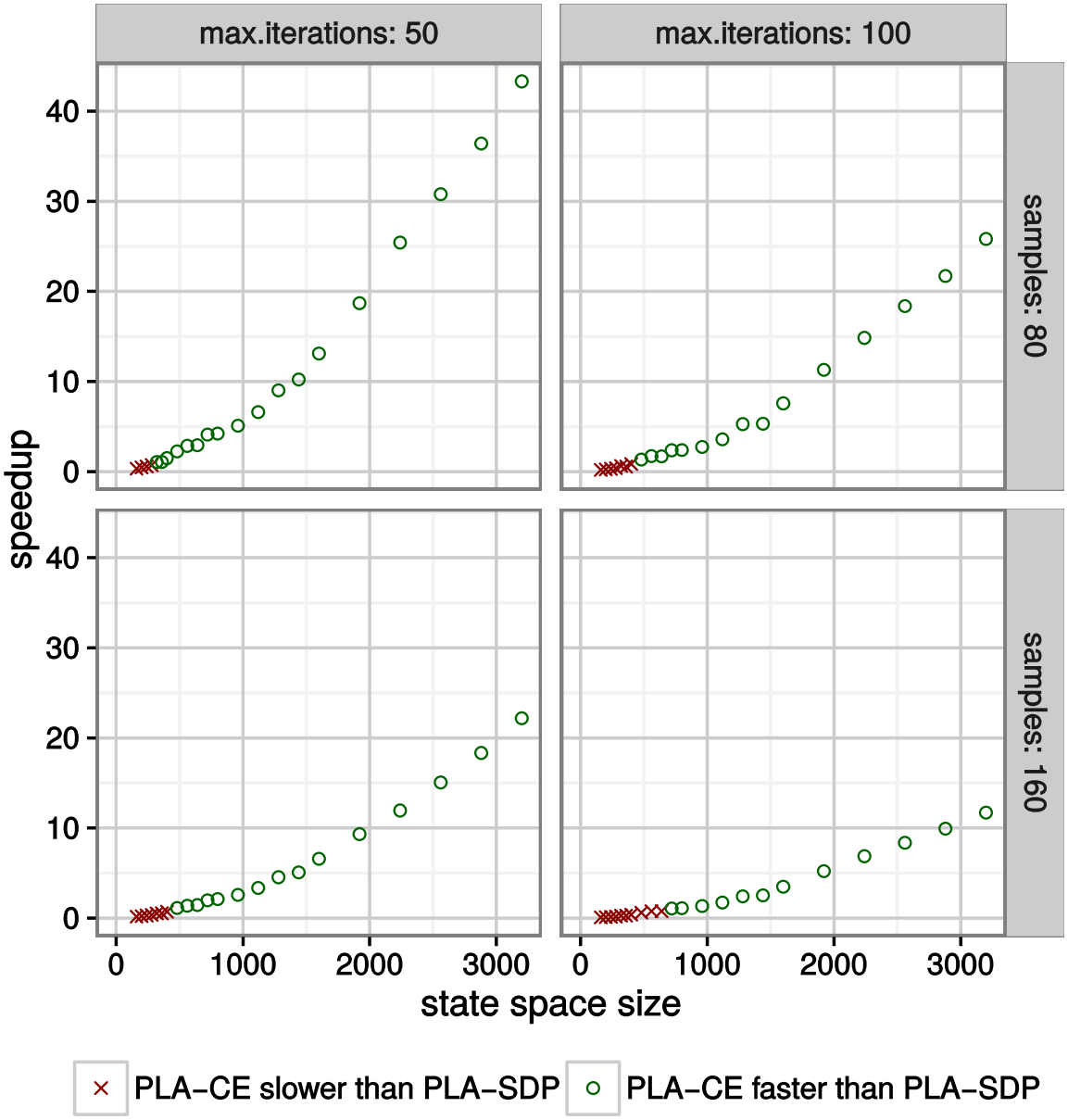
anytime algorithm

N random solutions

A solution is an *adaptation path*, each variable v_i is an adaptation action in the path. The system MDP is used to construct feasible adaptation paths

System MDP, environment model and utility function used to score solution

PLA-CE Speedup Compared to PLA-SDP



*each point based on avg. time of 2250 adaptation decisions

Outline

Adaptation decision problem

Solution approaches

- Probabilistic Model Checking (PLA-PMC)
- Stochastic Dynamic Programming (PLA-SDP)

Other forms of utility

Approximate solution with Cross-Entropy (PLA-CE)

Uncertainty reduction

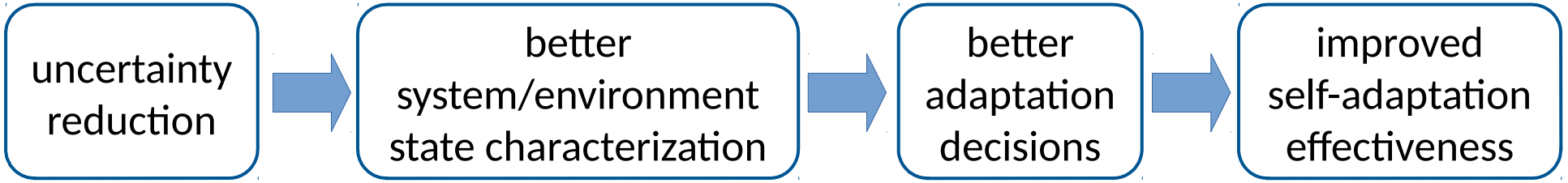
Uncertainty reduction

When deciding how to adapt, a self-adaptive system usually faces uncertainty.

For example:

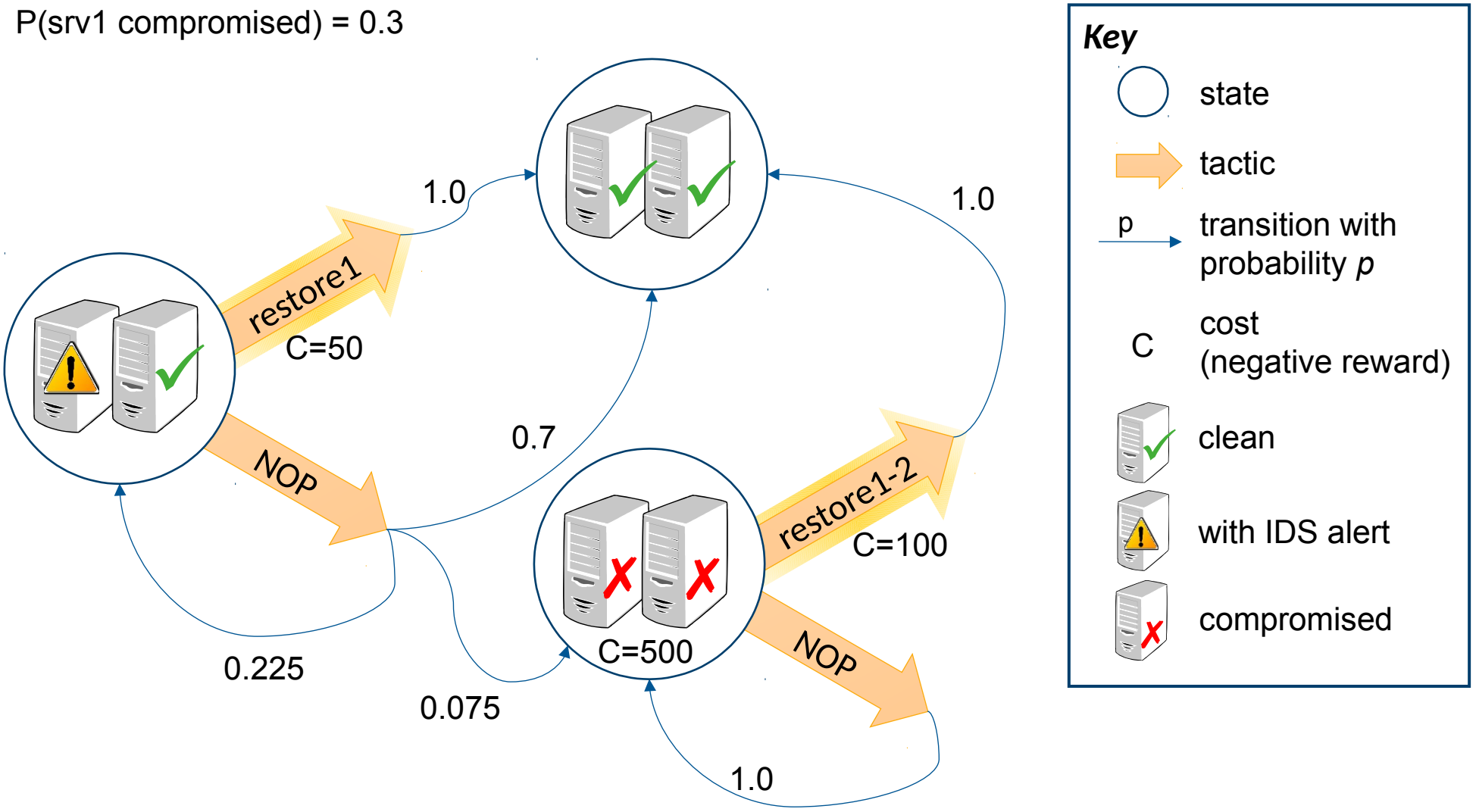
- uncertainty about whether a service not recently used is still available
- uncertainty about the state of the environment, perhaps only known with a wide confidence interval

Existing self-adaptation approaches can reason about uncertainty, but do not consider ways to reduce it. This is a missed opportunity to improve.

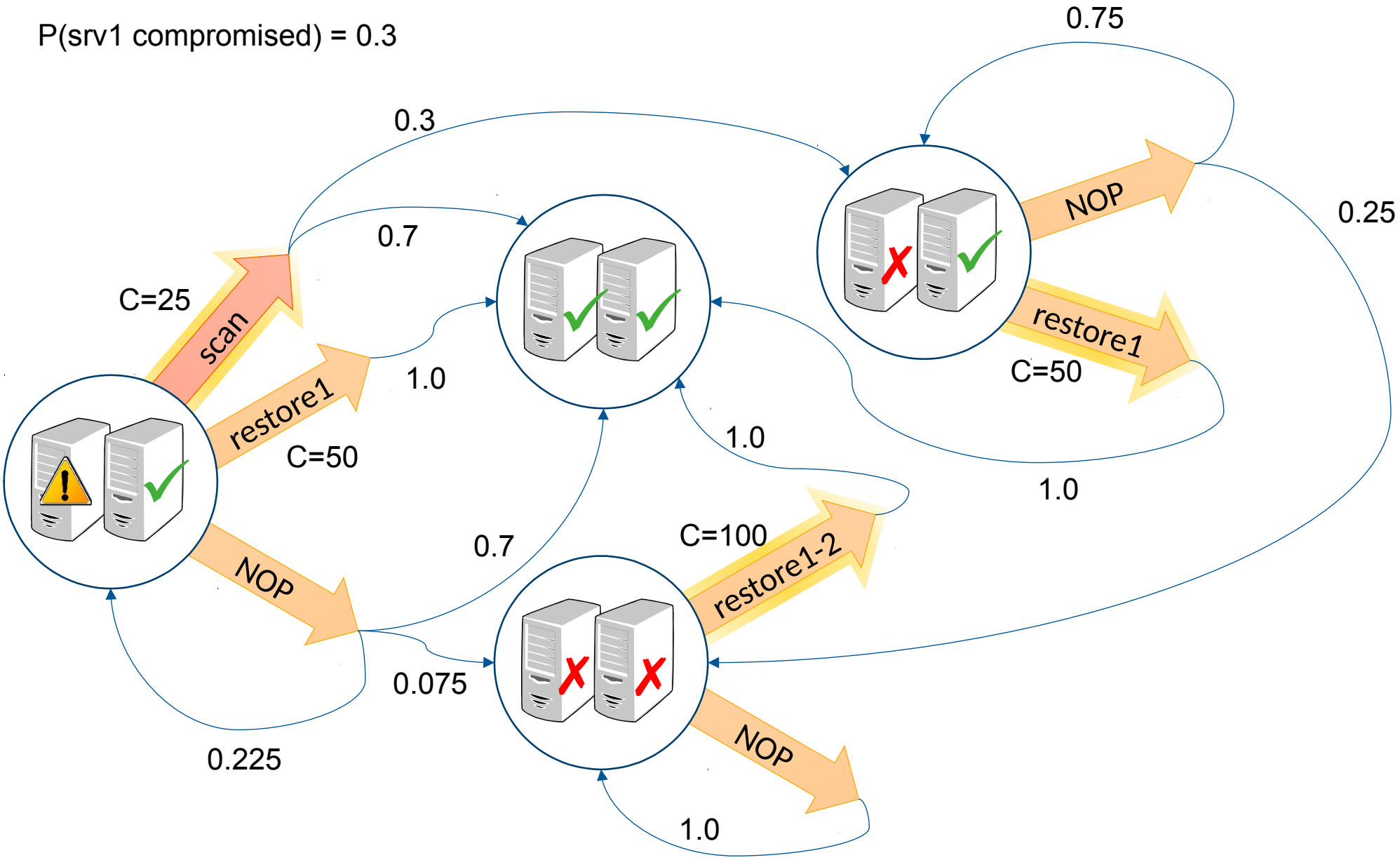


However, there are tradeoffs (resource consumption, user annoyance, etc.), which must be considered when deciding if and when to reduce uncertainty.

Example of Adaptation Decision Under Uncertainty



Introducing Uncertainty Reduction



Summary

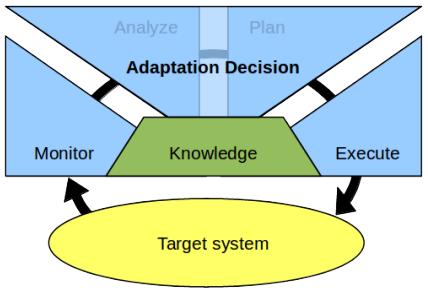
Proactive latency-aware adaptation improves the effectiveness of self-adaptation

- tactic latency
- look-ahead decision horizon
- uncertainty of the environment
- concurrency tactic execution

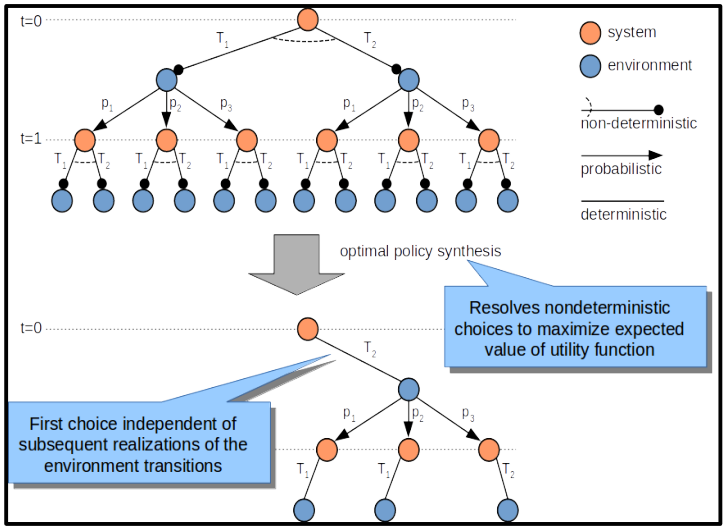
Anytime approximations perform well in practice and can speed up adaptation decisions.

Considering uncertainty in self-adaptation has benefits. Next step is to manage uncertainty in self-adaptive systems.

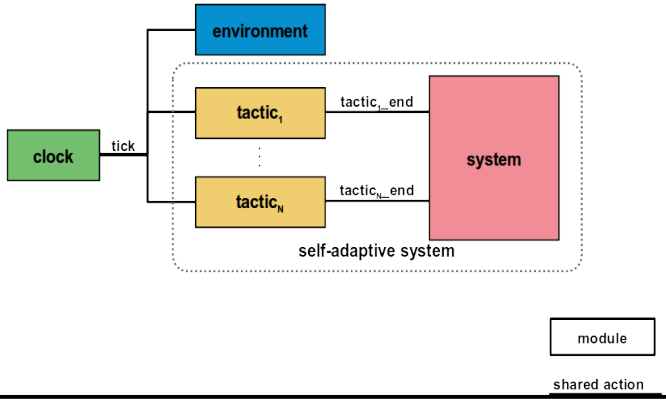
Adaptation decision



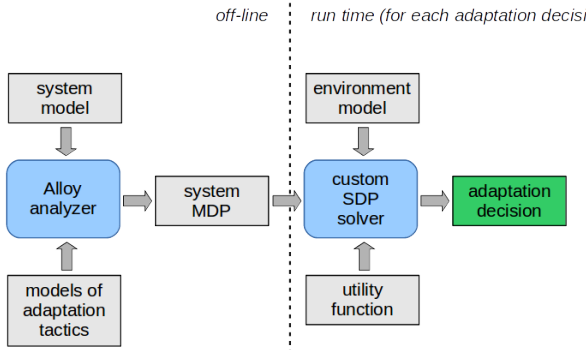
Answers the question of what adaptation tactic(s) should be started now to achieve the adaptation goal



PLA-PMC model

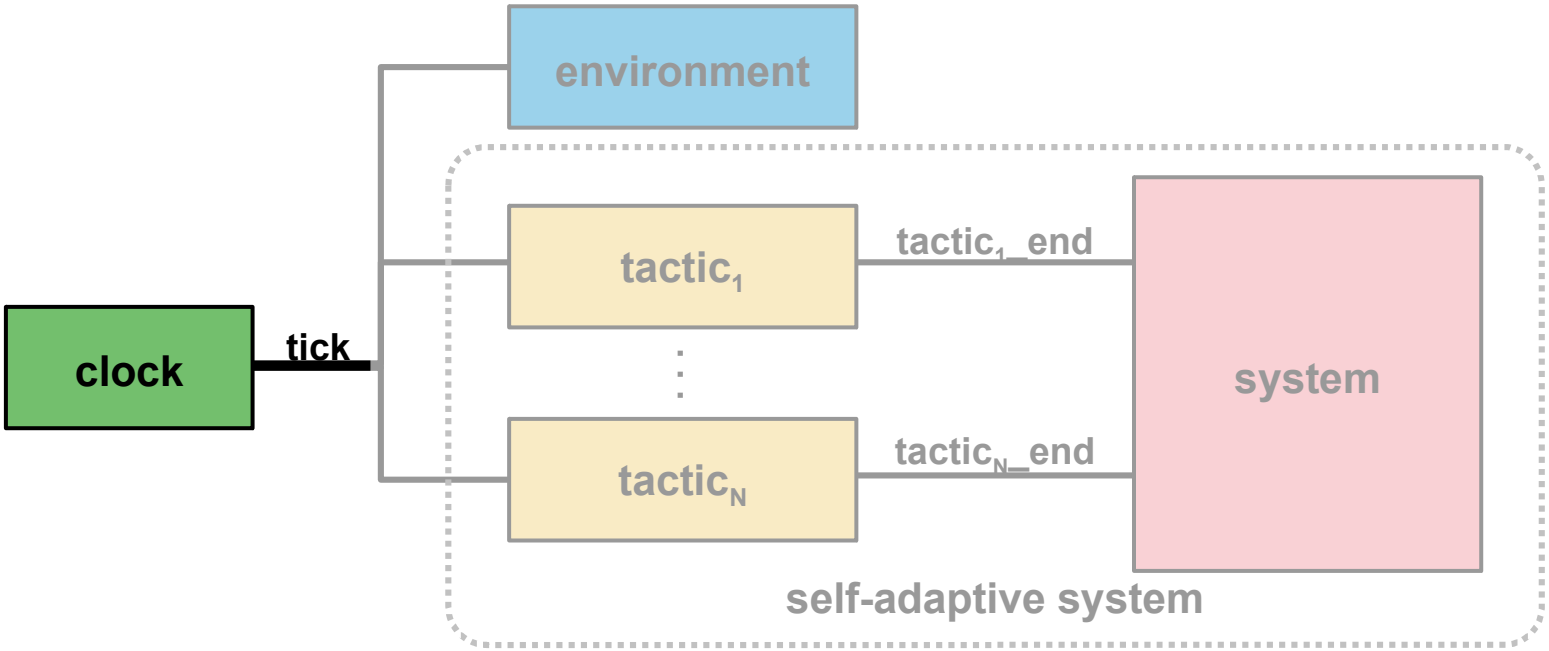


PLA-SDP



Additional Slides

PLA-PMC model



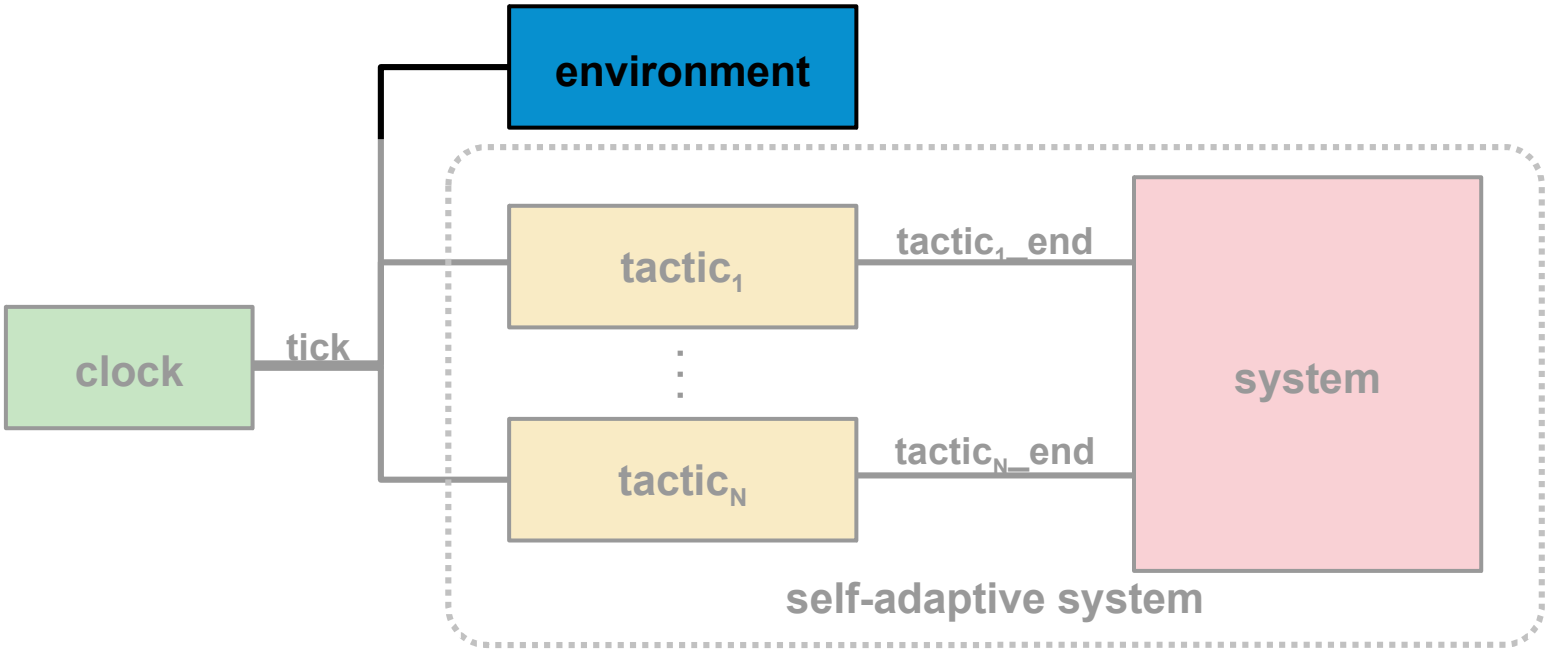
Clock module

- controls the progress of time
- schedules actions: system, environment, utility calculation

module

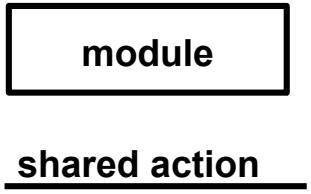
shared action

PLA-PMC model

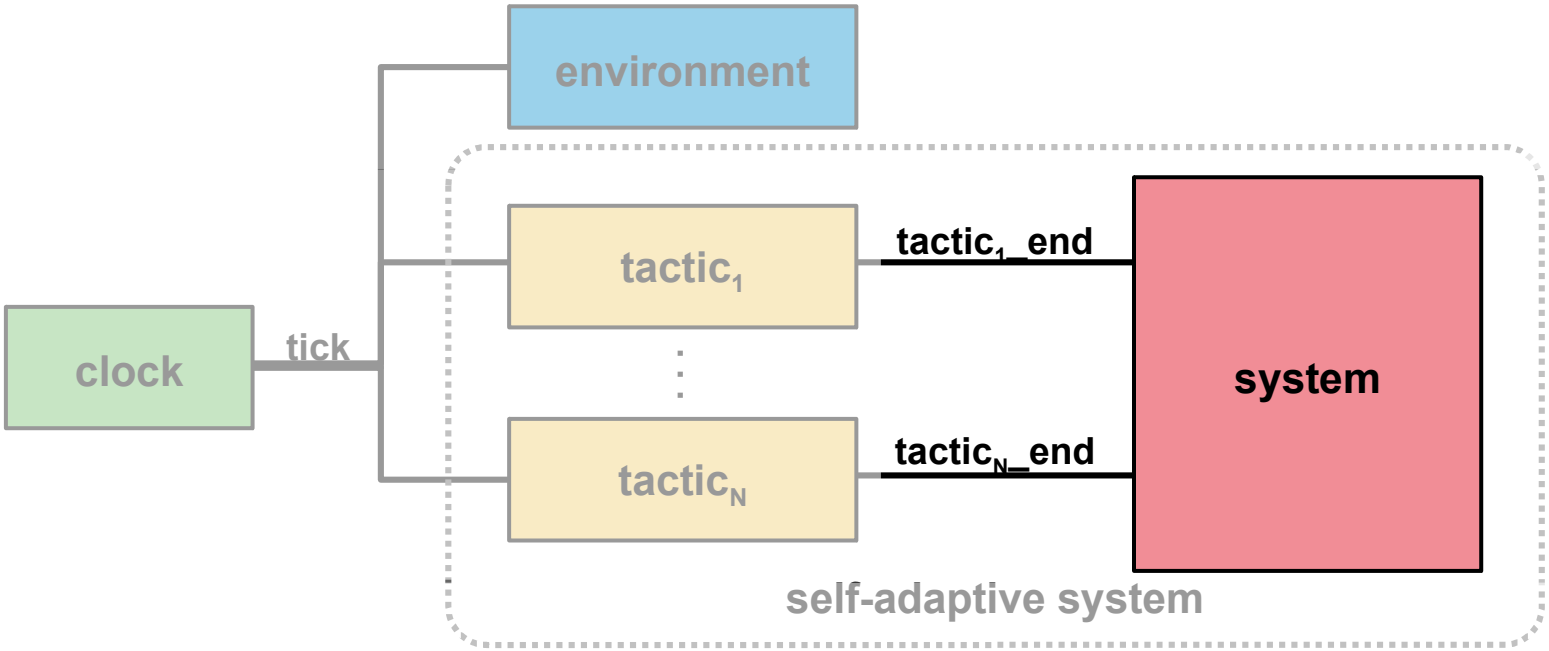


Environment module

- stochastic model of the environment prediction for the decision horizon
- generated at runtime before each adaptation decision

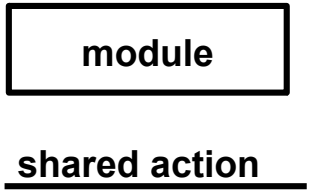


PLA-PMC model



System module

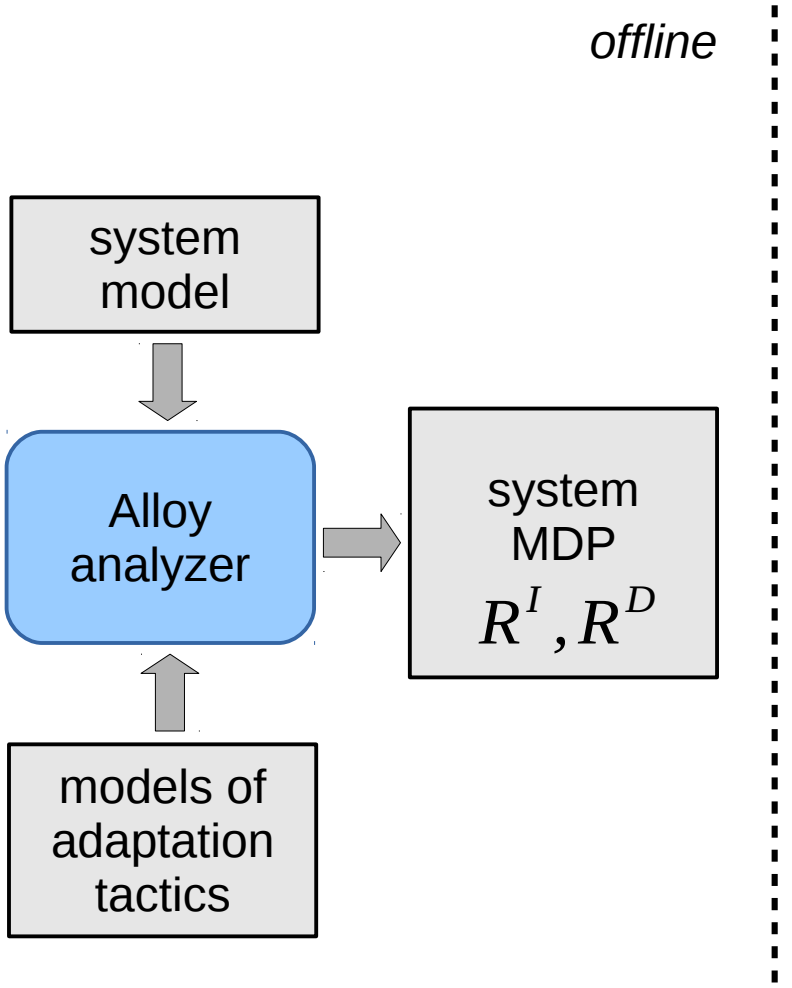
- properties needed to compute utility function or determine tactic applicability
- commands synchronized with tactics to capture their effect on the system



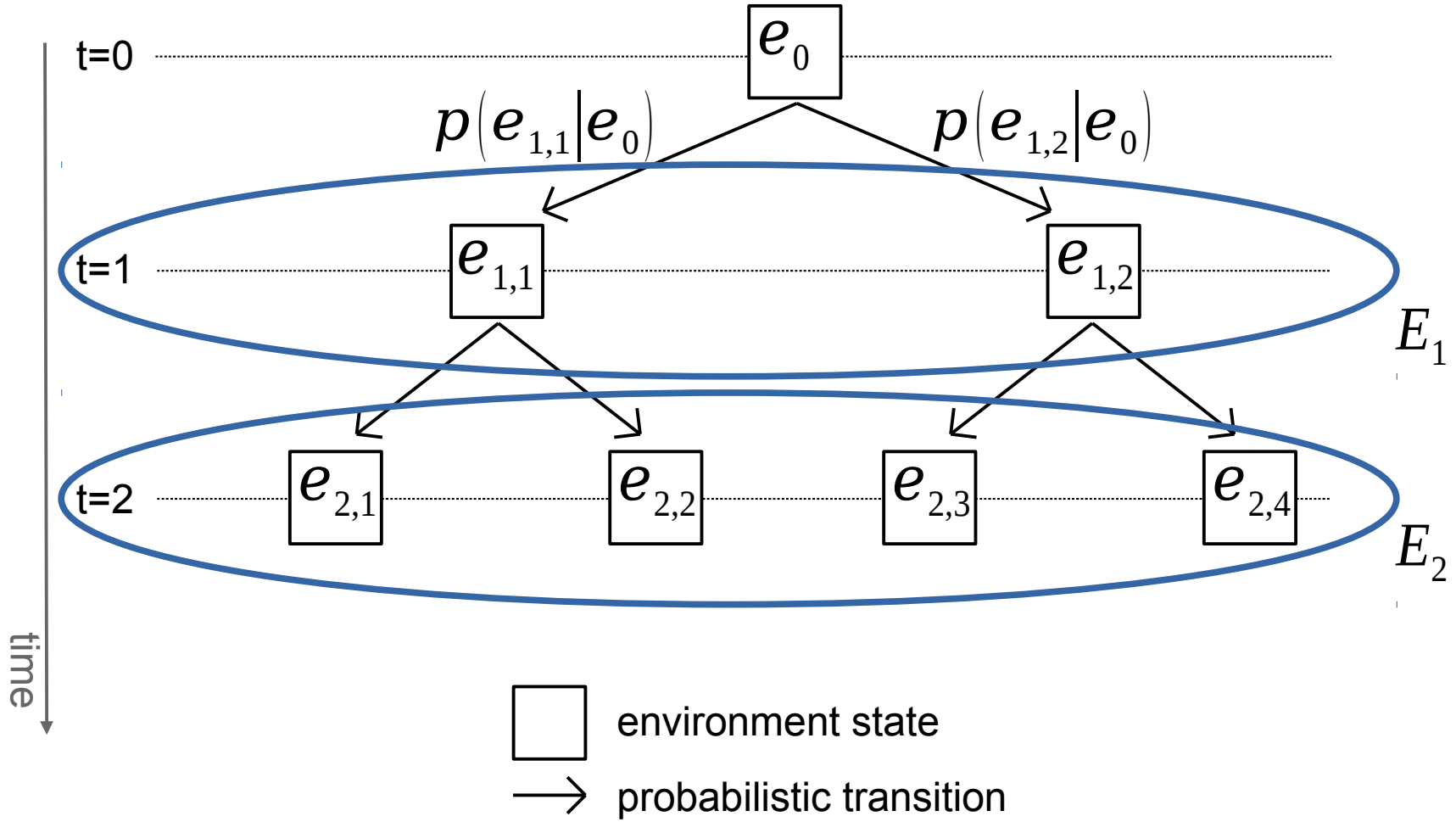
Computing reachability predicates

We use formal specification and analysis with Alloy

- declaratively specify the effect of tactics on system state
- exhaustively explore all the combinations of tactics and system states
- generate reachability predicate definition by extension



Stochastic environment model



Solution approaches comparison

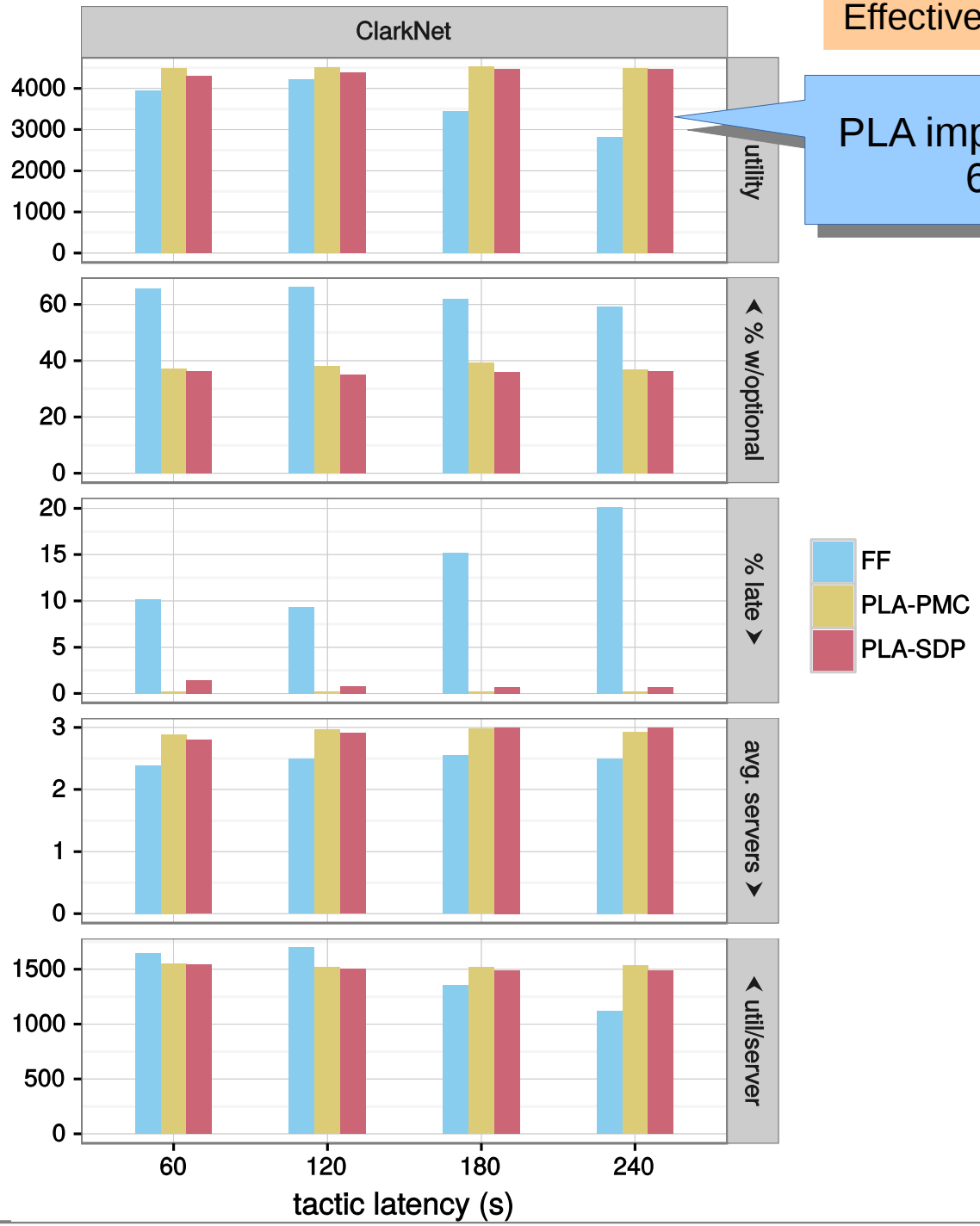
Feature	PLA-SDP	PLA-PMC
handles environment uncertainty	yes	yes
flexibility to change adaptation goal	limited	yes
optimal	yes	yes
fast	yes	no
can invoke external code	yes	no
easy to modify	no	yes

Forms of utility supported

When reward is gained	maximize $_{c_1,\ldots,c_H}$
RG1: *regardless of* the constraint satisfaction	$\sum_{t=1}^{H} g(c_t, e_t)$
RG2: *only when* the constraint is satisfied	$\sum_{t=1}^{H} s(c_t, e_t) g(c_t, e_t)$
RG3: *as long as* the constraint has been and is satisfied	$\sum_{t=1}^{H} \left(\prod_{i=1}^{t} s(c_i, e_i) \right) g(c_t, e_t)$

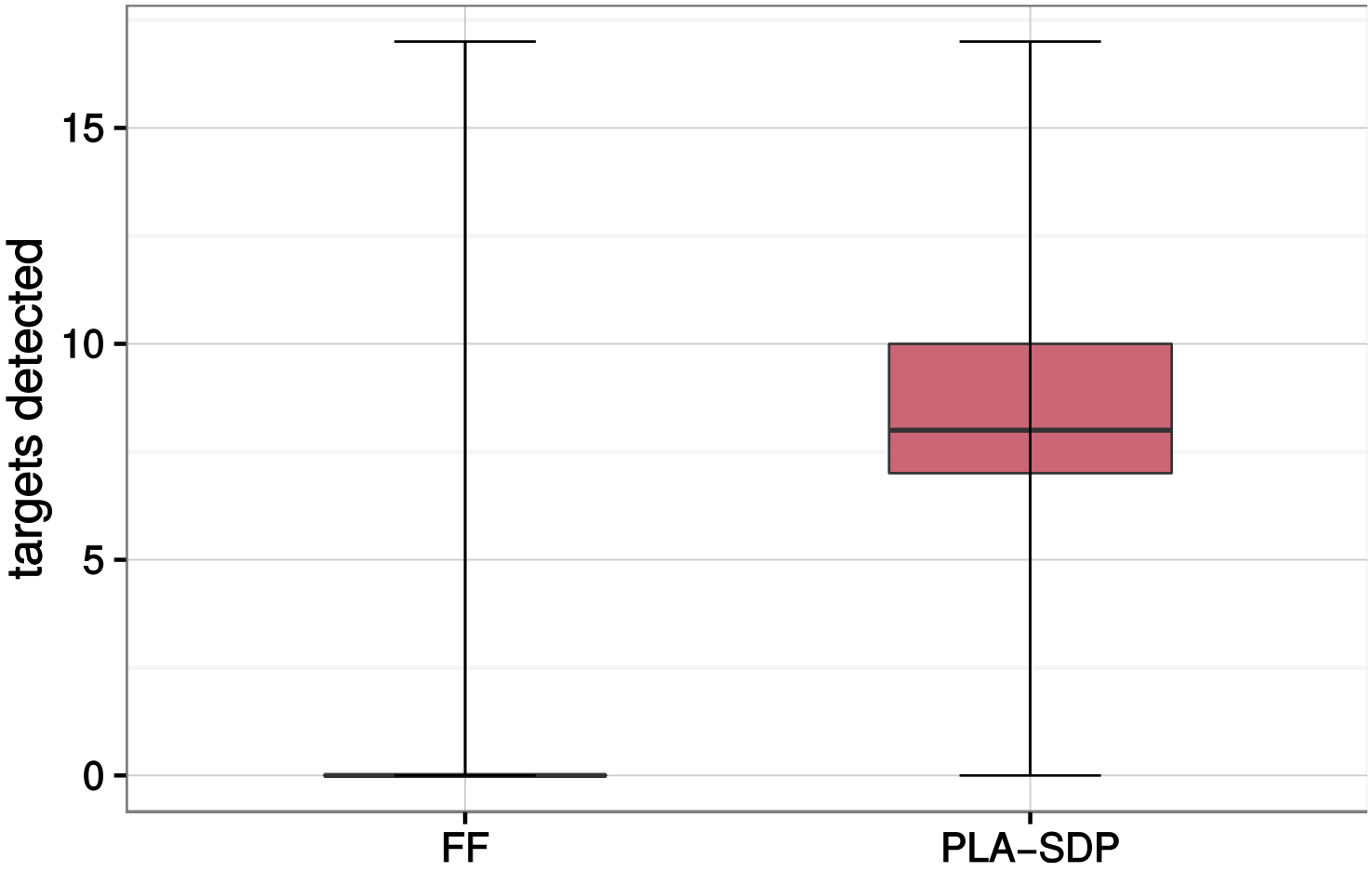
Bound on the probability of	subject to
CS1: satisfying the constraint *in each period*	$\forall t \in \{1, \ldots, H\}\ s(c_t, e_t) \geq P$
CS2: satisfying the constraint *over all periods*	$\prod_{t=1}^{H} s(c_t, e_t) \geq P$

Effectiveness improvement



PLA improvement: 60%

If only targets of survived missions count...



5000 simulated missions

PLA-SDP scalability

Examples:

- it is fast to run DART self-adaptation on the drone on-board computer
- it can handle self-adaptation for a full regional cluster for the World Cup ‘98 traffic

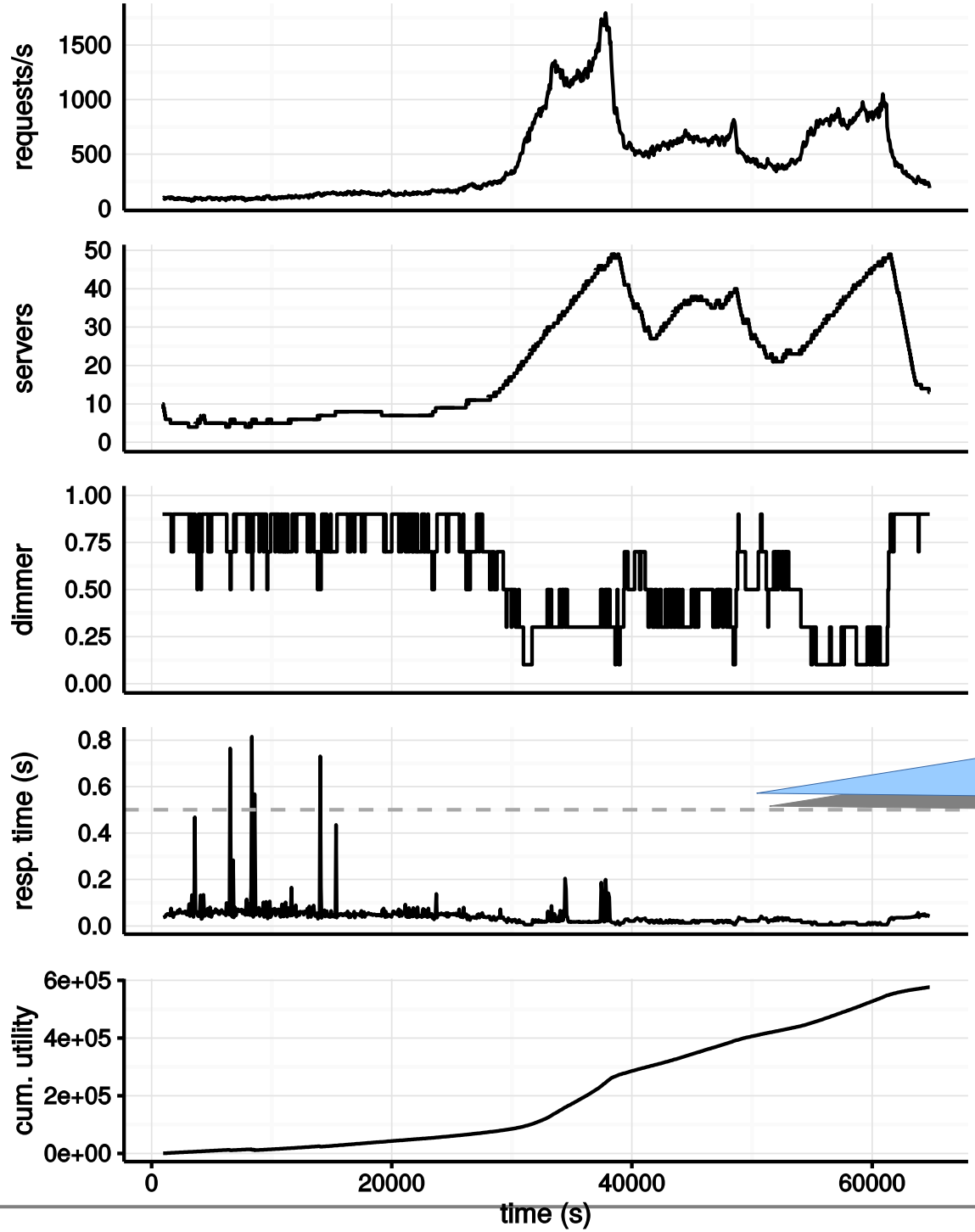
PLA in embedded computer

Decision time requirement for missions like DART's in published literature:

- Most stringent: 1 adaptation decision per second

Ran PLA-SDP on Raspberry Pi 3, on-board computer of DART drones

- avg. decision time: 113 ms
- with additional pair of tactics: 494 ms

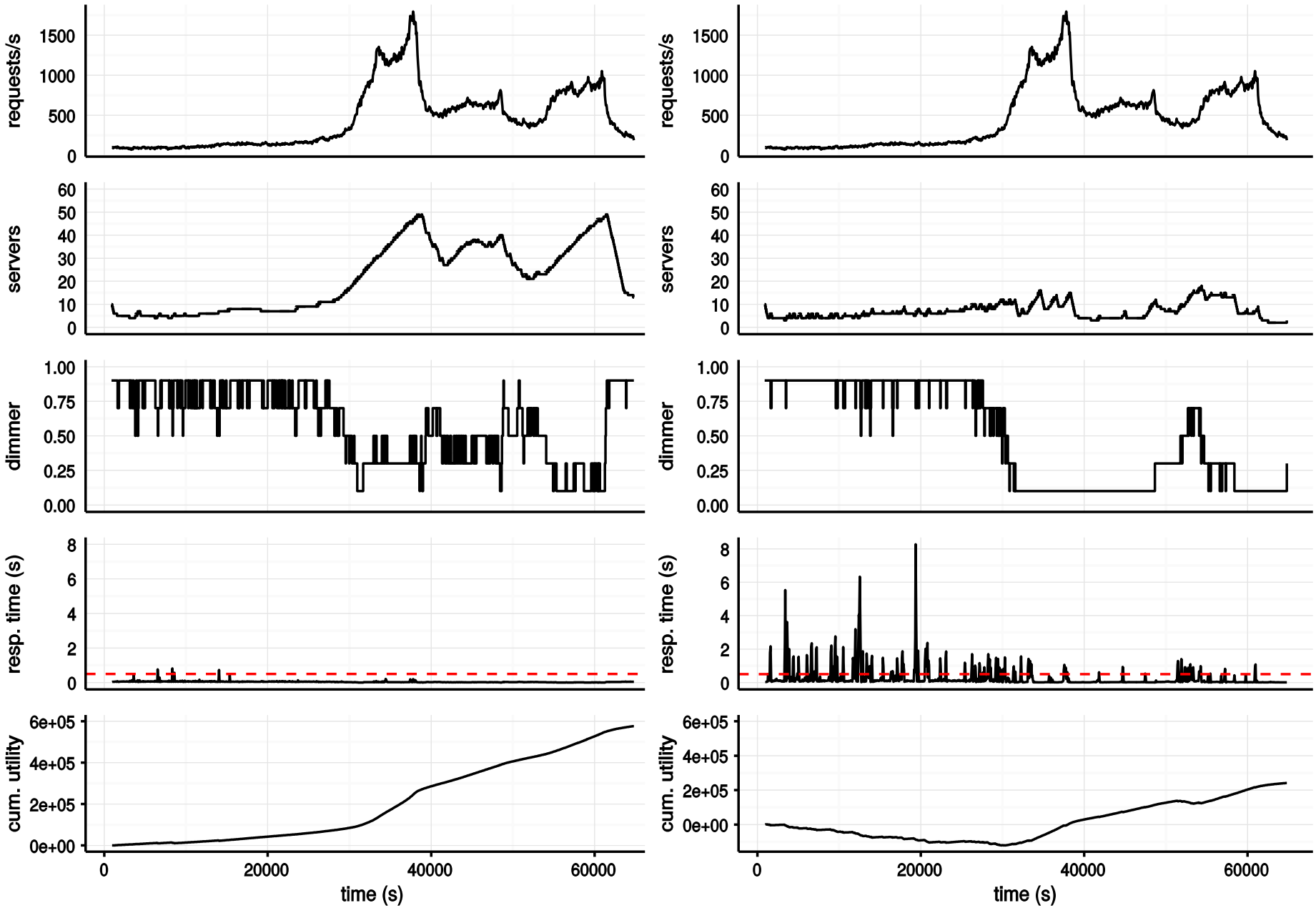


Trace: 18 hours of traffic to a whole regional World Cup ‘98 regional cluster (not scaled)

Cluster size: 60 servers

Evaluation period: 60 secs.

Only 4 response time violations
Avg. decision time 8 secs.



PLA-SDP

FF

Related work: proactive adaptation

service-based systems [Calinescu 2011, Hielscher 2008, Metzger 2013, Wang 2012]

- to deal with changing QoS
- mostly limited to testing services proactively

RESIST, proactive adaptation for reliability [Cooray 2010]

anticipatory dynamic configuration [Poladian 2007]

- look-ahead to account for reconfiguration cost
- ignores latency

Related work: adaptation latency

limiting the time to synthesize a controller [Musliner 2001]

specifically dealing with server setup time [Gandhi 2012]

minimizing adaptation time once the target configuration has been selected [Zhang 2004]

Related work: quantitative verification

runtime quantitative verification (RQV): use of model checking at runtime in self-adaptive systems [Calinescu 2012]

techniques to speed up RQV [Gerasimou 2014]

Related work: MDPs and other timed models

Concurrent Markov Decision Processes [Mausam 2004]

- supports concurrent actions of unit duration

continuous-time MDP [Guo 2009], and timed MDP [Jurdzinsky 2007],

- associate timing properties with actions, which can be taken one at a time.

UPPAAL [Bengtsson 2004]

- only supports verification of liveness, safety and reachability properties

UPPAAL Stratego [David 2015]

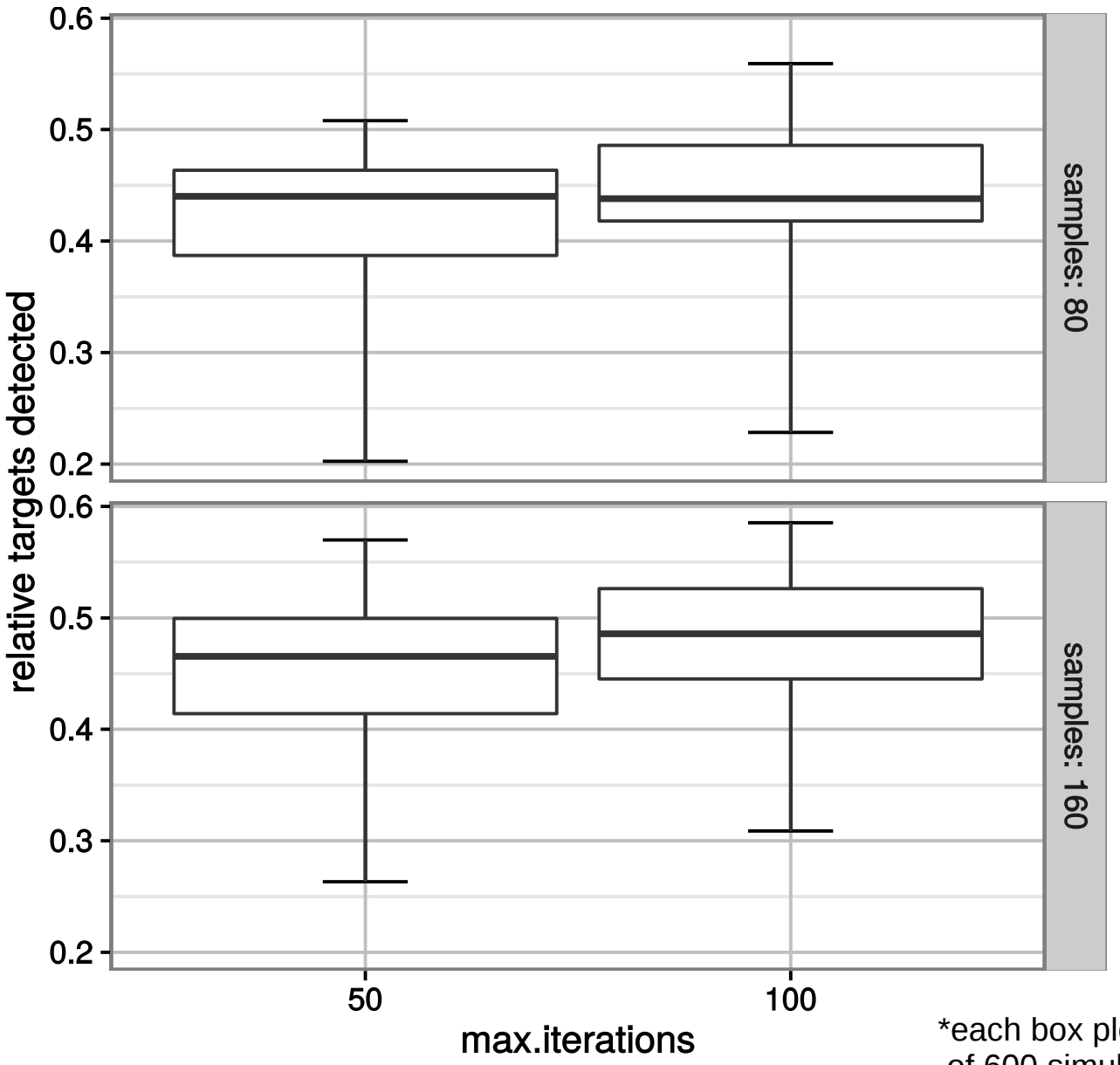
- Generates nondeterministic strategy that ignore rewards, and then uses learning to resolve the nondeterminism for reward maximization.

Related work: Model Predictive Control

CobRA [Angelopoulos 2016]

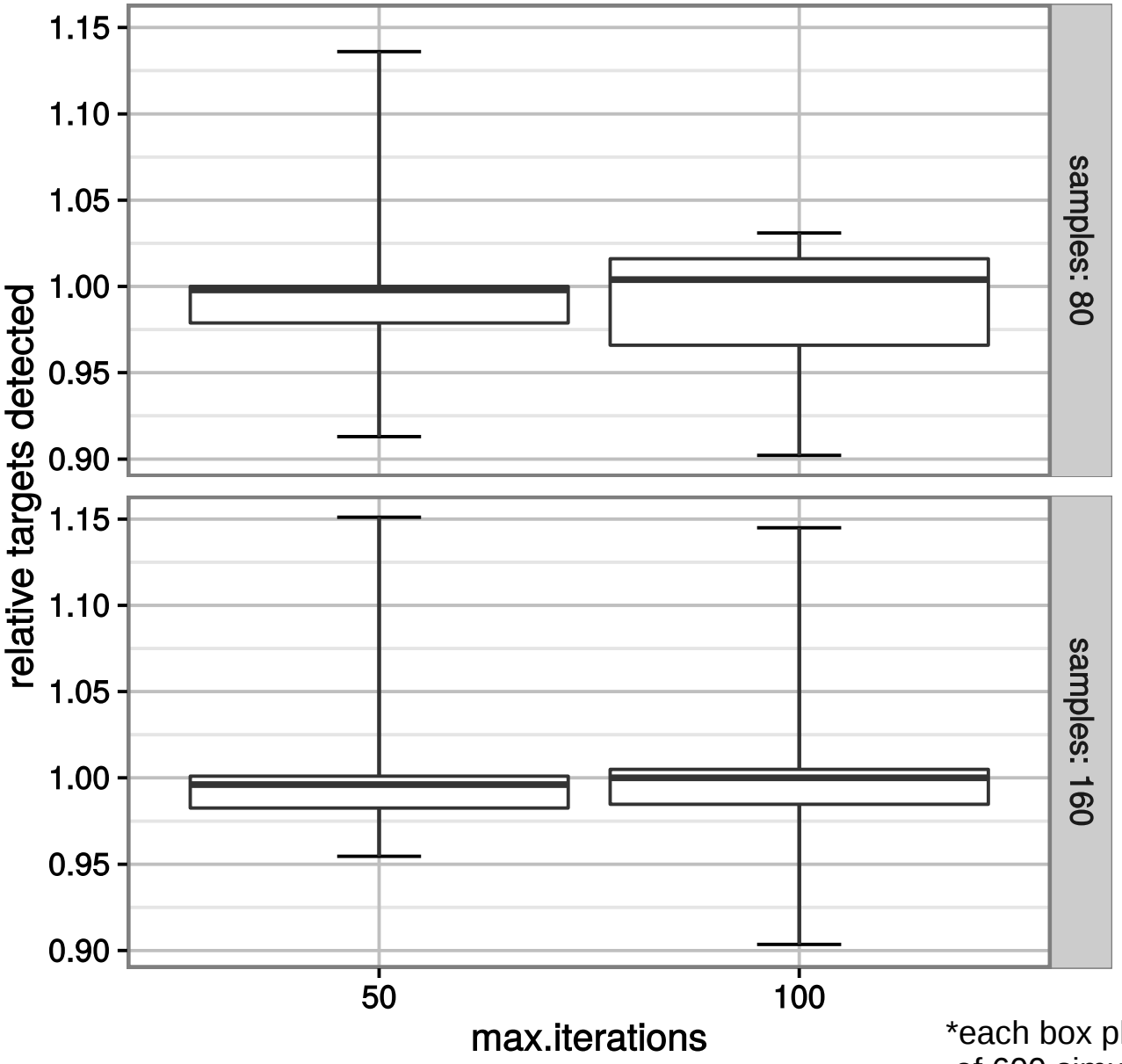
- Ignores that control actions may preclude others in the near future
- Does not consider control action latency
- Does not handle state-dependent applicability of tactics

PLA-CE Effectiveness Relative to Optimal



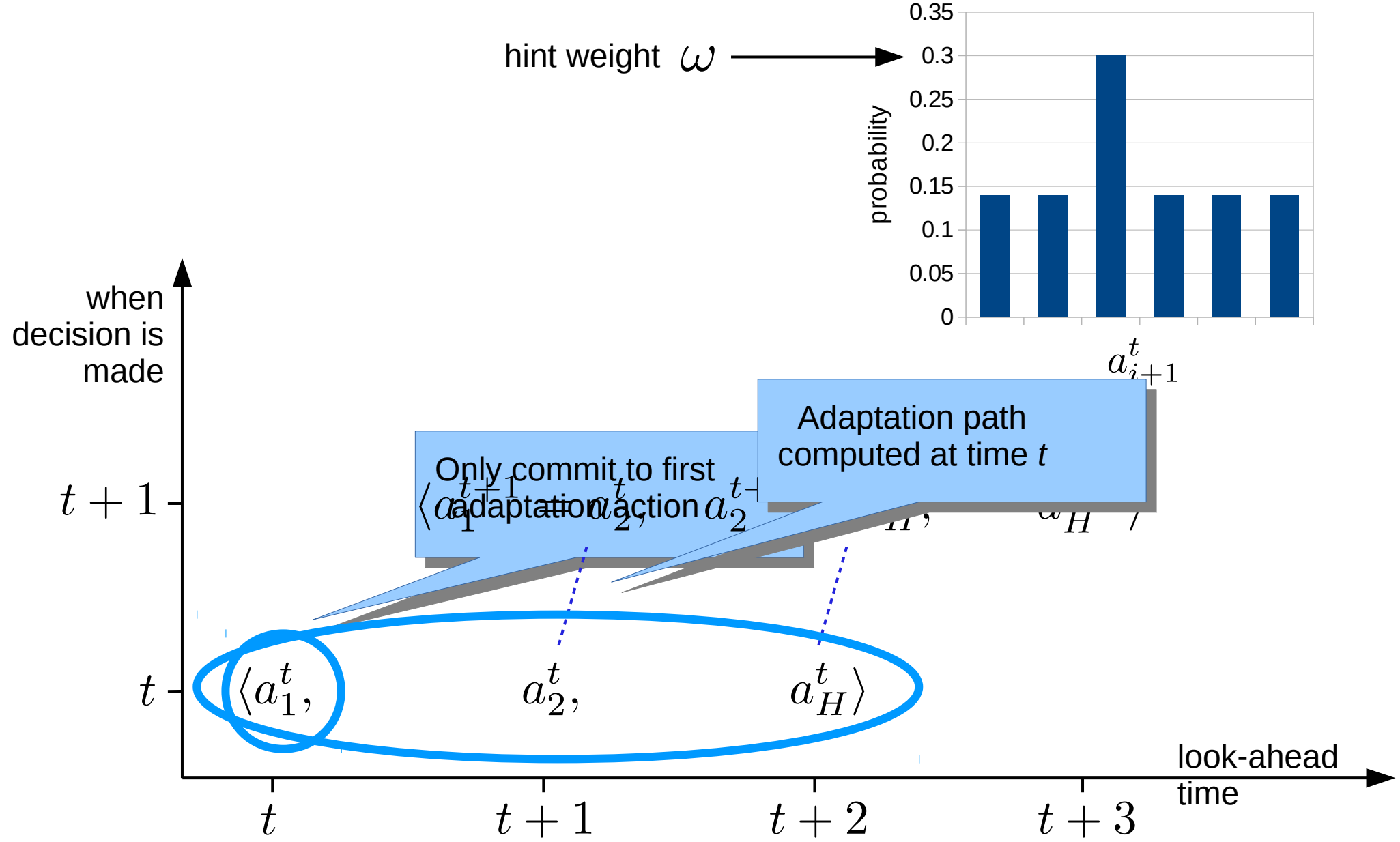
*each box plot shows result of 600 simulated missions

PLA-CE Effectiveness Relative to PLA-SDP



*each box plot shows result of 600 simulated missions

Using Hints



PLA-CE Decision Time