Learning-Based Mean-Payoff Optimization

Guillermo A. Pérez

with J. Křetínský, F. Michel, L. Michel (TU Munich); R. Berthon, G. Geeraerts, S. Guha, J.-F. Raskin (U. libre Bruxelles)
### Reactive systems

Reactive (computer) systems maintain a **continuous** interaction with their environment.

1. The evolution of the state of the system partially depends on the uncontrollable environment.
2. Correct executions of the system **do not terminate**.

![Diagram of Agent/Controller and Environment interaction](image)

- **Agent/Controller**
- **Environment**
- **action**
- **next state**, **reward/cost**
Reactive systems

Reactive (computer) systems maintain a continuous interaction with their environment.

1. The evolution of the state of the system partially depends on the uncontrollable environment.
2. Correct executions of the system do not terminate.

Main problem: How does one choose actions?
Reactive synthesis for hard guarantees

Reactive synthesis

Given a model $M$ of the system and a formal specification $\varphi$, synthesize a strategy $\sigma$ such that, against any behaviour of the environment, the resulting execution satisfies $\varphi$.

Specs: see red only finitely often (eventual safety), blue at least once (reachability), green infinitely often (liveness)

- there exists a strategy for $\Box \neg \text{red} \land (\Box \text{green} \lor \Diamond \text{blue})$
Reinforcement learning

Online reinforcement learning

Given no prior information about the system, play actions and obtain information about the states, transition probabilities, and rewards so as “to maximize” some notion of cumulative reward.

- This can be seen as fixing a strategy for all systems, and having it adapt to the actual system (an MDP) on the fly.

![Diagram of an agent/controller environment](image-url)

- Action
- Next state, reward/cost
- Environment
Beyond-worst case in MDPs (1/2)

<table>
<thead>
<tr>
<th>Must be known</th>
<th>Reactive synthesis</th>
<th>Reinforcement learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>model, spec.</td>
<td>worst-case</td>
<td>depends?</td>
</tr>
</tbody>
</table>

This talk

How efficient can reinforcement-learning techniques be if one asks that almost all executions must be correct, i.e. satisfy a given formal specification, and only finite memory is used?

- efficiency with respect to the expected mean payoff
- a parity objective as the specification
Objectives

▶ The sequence of priorities $p_0 p_1 \cdots \in \mathbb{N}^\omega$ is good for the parity objective iff $\liminf_{i \in \mathbb{N}} p_i$ is even.

▶ The mean payoff function maps weights $x_0 x_1 \cdots \in \mathbb{Q}^\omega$ to

$$\text{MP}(x_0 x_1 \ldots) = \liminf_{n \geq 1} \frac{1}{n} \sum_{i=0}^{n-1} x_i.$$
How unknown is the MDP?

What do we feed into our RL/synthesis algorithm?

As input we have only the support automaton of an unknown MDP.

- Probabilities and rewards are unknown but learnable on the fly,
- while the transitions, states, actions, and priorities, are known.
How unknown, REALLY, is the MDP?

**Assumptions**

- A transition-probability lower bound $\pi_{\text{min}}$ is known, so that $P(q, a, q') \neq 0 \implies P(q, a, q') \geq \pi_{\text{min}}$ for all $(q, a, q')$.

- All transition rewards $r$ are rationals $0 \leq r \leq 1$.

- Rewards are observed immediately after a transition has been performed.

- Only outcomes of probabilistic transitions are observed, not the probabilities themselves.
Task (solved here with infinite memory)

Synthesize a strategy that satisfies the parity objective almost surely, and “maximizes” the expected mean payoff.

1. Walk randomly $|Q|$ steps, compute (new) estimates $\hat{x}, \hat{y}$ of $x, y$.
2. Play an optimal MP strategy for $\hat{x}, \hat{y}$ during $k$ steps.
3. Restart (from 1.) with larger $k$. 
Optimizing the mean payoff in an unknown MDP

Mean-payoff optimization with almost-sure parity constraints
  in a single good end component
  in a single end component

Other results

Application to safe scheduling
Mean-payoff optimization

**Input**
- the support automaton $\mathcal{A}$ (no priorities for now)
- a transition-probability lower bound $\pi_{\min}$

**Assumptions**
- the unknown MDP $\mathcal{M}$ is an end component
  1. $\mathcal{A}$ is strongly connected
  2. if $\lambda$ is the strategy that plays actions uniformly at random, then
     $\Pr_{\mathcal{M}}^{q_0, \lambda} [\Diamond q_1] = 1$ for all pairs of states $q_0, q_1$
MP optimization: main approach

Proposed solution

1. Play $\lambda$ for $L$ steps.
   - $L$ should be large enough so that we can compute an estimated $P'$ s.t. $\|P - P'\|_\infty \leq \varepsilon$ and observe all transition rewards, both with probability at least $1 - \delta$.

2. Follow a unichain memoryless deterministic expectation-optimal strategy $\tau$ for the MDP with $P'$ for $O$ steps.

3. Forget and repeat from 1.

Theorem

For all $\varepsilon \in (0, 1)$ one can construct a finite-memory strategy $\sigma$ s.t.
$$\Pr_{M, \sigma}^{q_0} \left[ q : MP(q) \geq \sup_{\tau} \mathbb{E}_{M, \tau}^{q_0} [MP] - \varepsilon \right] = 1 \text{ for all } q_0.$$
Lemma

In single-end-component MDPs and for unichain memoryless deterministic strategies $\sigma$, almost all executions achieve the expected mean payoff, i.e. for all $q_0$

$$\Pr_{M^\sigma}^{q_0} [\rho : \text{MP}(\rho) \geq E_{M^\sigma}^{q_0} [\text{MP}]] = 1.$$ 

Lemma (Solan 03; Chatterjee 12)

All memoryless deterministic expectation-optimal strategies $\sigma$ for an MDP $N$ with the same support as $M$ and such that

$$\|P_N - P_M\|_\infty \leq \frac{\pi_{\min}}{2} \left(1 + \frac{\varepsilon}{2}\right) \frac{1}{2|Q|} - 1$$

guarantee that for all $q_0$

$$\left|E_{M^\sigma}^{q_0} [\text{MP}] - \sup_{\tau} E_{M^\tau}^{q_0} [\text{MP}]\right| \leq \varepsilon.$$
Since the MDP is an end component...

**Lemma (Hoeffding’s inequality)**

*All hitting probabilities when playing according to λ can be under-approx’d as biased coins X with success probability σ := (π_{\text{min}}/|A|)^{|Q|}, and we have:*

\[
\Pr \left[ \left| \frac{1}{k} \sum_{j=1}^{k} X_j - \sigma \right| \geq \varepsilon \right] \leq \frac{2}{\exp(2k\varepsilon^2)}.
\]

▶ we get a bound on the number of steps we need to follow λ to get a (1 − δ)-confidence interval P ± ε
We can choose $O$ so that almost all learn-and-exploit episodes achieve an $\varepsilon'$-optimal MP.

**Lemma (Tracol 09)**

In single-end-component MDPs and for unichain memoryless deterministic strategies $\sigma$, for all $\varepsilon \in (0, 1)$, one can compute $K_0 \in \mathbb{N}$ and $c_1, c_2 > 0$ such that for all $k \geq K_0$

$$\Pr_{\mathcal{M}_\sigma}^{q_0} \left[ \varrho : \text{FinAvg}(\varrho(\ldots k)) \geq \mathbb{E}_{\mathcal{M}_\sigma}^{q_0}[\text{MP}] - \varepsilon \right] \geq \frac{1 - c_1}{\exp(k \cdot c_2 \cdot \varepsilon^2)}.$$
MP optimization: correctness (3/3)

We can choose $O$ so that almost all learn-and-exploit episodes achieve an $\varepsilon'$-optimal MP.

**Lemma (Tracol 09)**

*In single-end-component MDPs and for unichain memoryless deterministic strategies $\sigma$, for all $\varepsilon \in (0, 1)$, one can compute $K_0 \in \mathbb{N}$ and $c_1, c_2 > 0$ such that for all $k \geq K_0$

$$\Pr_{q_0}^{q_0 \sigma} \left[ \varrho : \text{FinAvg}(\varrho(\ldots k)) \geq \mathbb{E}_{q_0}^{q_0 \sigma} [\text{MP}] - \varepsilon \right] \geq \frac{1 - c_1}{\exp(k \cdot c_2 \cdot \varepsilon^2)}.$$*

- **limited information:** instead use an exact-mixing stopping-time algorithm (Lovász & Winkler '95; Propp & Wilson '98)
MP optimization: can’t always do better

The limits of finite-memory learning

- Either ‘a’ does not appear in any BSCC in the product Markov chain and thus the strategy “chooses a side” with finite information and probably incorrectly so.
- Or ‘a’ does appear in a BSCC and the strategy achieves a sub-optimal mean payoff with nonzero probability.
Optimizing the mean payoff in an unknown MDP

Mean-payoff optimization with almost-sure parity constraints
  in a single good end component
  in a single end component

Other results

Application to safe scheduling
Outline

Optimizing the mean payoff in an unknown MDP

Mean-payoff optimization with almost-sure parity constraints
  in a single good end component
  in a single end component

Other results

Application to safe scheduling
MP and parity in a good end component

<table>
<thead>
<tr>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ the support automaton $A$ (with priorities now!)</td>
</tr>
<tr>
<td>▶ a transition-probability lower bound $\pi_{\text{min}}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assumptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ the unknown MDP $M$ is a <strong>good end component</strong></td>
</tr>
<tr>
<td>1. the minimal priority of a state in $A$ is even</td>
</tr>
<tr>
<td>2. <em>almost all executions</em> consistent with $\lambda$ are good for the parity objective</td>
</tr>
<tr>
<td>3. $\Pr_{M,\lambda}^{q_0} [\text{Parity}] = 1$, for all $q_0$</td>
</tr>
</tbody>
</table>
Proposed solution

Follow our solution for MP only with $L \geq |Q|$.

Theorem

For all $\varepsilon \in (0, 1)$ one can construct a finite-memory strategy $\sigma$ s.t. for all $q_0$ the following hold:

- $\Pr_{q_0}^{M_\sigma} [\text{Parity}] = 1$, and
- $\Pr_{q_0}^{M_\sigma} [q : \text{MP}(q) \geq \sup_\tau E_{q_0}^{\tau_q} [\text{MP}] - \varepsilon] = 1$. 

20/38
Same as before, additionally we observe the following

- the strategy learns by random exploration;
- it learns for at least $L$ steps infinitely often;
- since we are in a GEC, the parity condition is satisfied almost surely!
Outline

Optimizing the mean payoff in an unknown MDP

Mean-payoff optimization with almost-sure parity constraints
  in a single good end component
  in a single end component

Other results

Application to safe scheduling
MP and parity in an end component

**Input**
- the support automaton $A$
- a transition-probability lower bound $\pi_{\text{min}}$

**Assumptions**
- the unknown MDP $\mathcal{M}$ is an end component
- there exists a strategy $\sigma$ that almost-surely satisfies the parity objective
- $\implies$ the MDP contains a good end component
Proposed solution

1. Play $\lambda$ for $L$ steps.
   - get an estimated transition function $P'$ such that $\|P - P'\|_\infty \leq \varepsilon$ and record all rewards, both with probability at least $1 - \delta$.

2. Select a **good end component** with maximal expectation in the MDP with estimated functions $P'$ and $r'$.

3. Follow $\lambda$ until $E$ is reached.

4. Apply the solution for good end components.
MP and parity in an EC: the guarantees

Theorem

For all $\varepsilon, \delta \in (0, 1)$ one can construct a finite-memory strategy $\sigma$ s.t. for all $q_0$ the following hold:

$\Pr_{\mathcal{M}\sigma}^{q_0} [\text{Parity}] = 1$, and

$\Pr_{\mathcal{M}\sigma}^{q_0} [\varrho : \text{MP}(\varrho) \geq \text{asVal}(\mathcal{M}) - \varepsilon] \geq 1 - \delta$

where $\text{asVal}(\mathcal{M})$ is the best $\mathbb{E}[\text{MP}]$ under the almost-sure parity constraint.
We are recycling the result for good end components, we still need

- establish a relation between $\text{asVal}(\mathcal{M})$ and good sub-end components with maximal expected mean payoff
We are recycling the result for good end components, we still need to establish a relation between $\text{asVal}(\mathcal{M})$ and good sub-end components with maximal expected mean payoff.

Lemma (Almagor, Kupferman, Velner 16)

In single-end-component MDPs $\mathcal{M}$ the following holds:

$$\text{asVal}(\mathcal{M}) = \max \left\{ \sup_{\tau} \mathbb{E}^{q_0}_{\mathcal{N}\tau} [\text{MP}] : \mathcal{N} \text{ is a GEC contained in } \mathcal{M} \right\}$$

for any $q_0$. 

MP and parity in an EC: can’t do better

The limits of infinite-memory learning

- The global learning step must be finite along almost all executions because of the a.s. parity constraint.
- So we can never be totally sure which side has the best expected mean payoff:
  - we will have a non-zero prob. of having a sub-optimal exp. mean payoff.
Outline

Optimizing the mean payoff in an unknown MDP

Mean-payoff optimization with almost-sure parity constraints
  in a single good end component
  in a single end component

Other results

Application to safe scheduling
Better with infinite memory

- With(out) almost-sure parity constraints and in a GEC, we get the optimal expected mean payoff almost surely.
- This is a better guarantee than, e.g. Q-learning, with the same assumptions.

Sure parity constraints with infinite memory

- In a GEC: optimal with arbitrarily high probability
- In an EC: near-optimal with arbitrarily high probability
Optimizing the mean payoff in an unknown MDP

Mean-payoff optimization with almost-sure parity constraints
  in a single good end component
  in a single end component

Other results

Application to safe scheduling
Scheduling hard and soft tasks

**Hard tasks**

are tuples \((I, C, D, A)\) such that

- \(I \in \mathbb{N}\) is the **arrival time** of the first job,
- \(D \in \mathbb{N}\) is the (relative) **deadline** of all jobs generated by the task
- \(C : \{1, 2, \ldots, D\} \rightarrow [0, 1]\) is a discrete probability distribution over possible job-computation times, and
- \(A : \{D, D + 1, \ldots\} \rightarrow [0, 1]\) is a distribution over finitely many possible inter-arrival times.

**Soft tasks**

These are hard tasks with an associated cost \(c \in \mathbb{Q}, c \geq 0\).
Scheduling hard and soft tasks

Hard tasks

are tuples \((I, C, D, A)\) such that

- \(I \in \mathbb{N}\) is the arrival time of the first job,
- \(D \in \mathbb{N}\) is the (relative) deadline of all jobs generated by the task
- \(C : \{1, 2, \ldots, D\} \rightarrow [0, 1]\) is a discrete probability distribution over possible job-computation times, and
- \(A : \{D, D + 1, \ldots\} \rightarrow [0, 1]\) is a distribution over finitely many possible inter-arrival times.

Soft tasks

These are hard tasks with an associated cost \(c \in \mathbb{Q}, c \geq 0\).
Actions

The scheduler chooses a task and gives one CPU time unit of execution time to its active job.
### Actions
The scheduler chooses a task and gives one **CPU time unit** of execution time to its active job.

### States
It consists of a product of the states per task:

1. the remaining time $R \leq D$ to deadline,
2. a distribution $\hat{D} : \{1, 2, \ldots, R\} \rightarrow [0, 1]$ over the possible remaining computation times,
3. a distribution $\hat{A} : \{R, R + 1, \ldots\} \rightarrow [0, 1]$ over the possible (relative) inter-arrival times.
Transitions

We update the state via the natural conditional probability updates of $\hat{D}, \hat{A}$ using Bayes’ rule and $R := R - 1$. Problem statement [GGR18]

Find a strategy $\sigma$ that avoids states with hard-deadline misses and which minimizes the mean-payoff of soft-deadline costs.
Semantic MDP and problem statement

Transitions
We update the state via the natural conditional probability updates of $\hat{D}, \hat{A}$ using Bayes’ rule and $R := R - 1$.

\[
(1, 2, 3) \overset{h}{\rightarrow} (0, 1, 2) \quad (0, 1, 2) \overset{1}{\rightarrow} (0, 1, 2)
\]

\[
(1, 1, 2) \quad (0, 1, 2) \quad (1, 1, 2)
\]

\[
\{1 \mapsto .4, 2 \mapsto .6\}, 2, 3 \quad \{1 \mapsto .4, 2 \mapsto .6\}, 1, 2 \quad \{1 \mapsto .4, 2 \mapsto .6\}, 1, 2
\]

\[
\{0 \mapsto .4, 1 \mapsto .6\}, 1, 2
\]

\[
(1, 1, 2) \quad (1, 1, 2)
\]
Semantic MDP and problem statement

Transitions

We update the state via the natural conditional probability updates of $\hat{D}, \hat{A}$ using Bayes’ rule and $R := R - 1$.

Problem statement [GGR18]

Find a strategy $\sigma$ that avoids states with hard-deadline misses and which minimizes the mean-payoff of soft-deadline costs.
A more realistic model

What if the execution and inter-arrival probabilities were not known?
Offline efficient PAC learnability

A more realistic model
What if the execution and inter-arrival probabilities were not known?

Efficiently PAC learnable (thanks Hoeffding!)
In the absence of hard tasks, all probabilities can be learned within an \((\varepsilon, \delta)\)-confidence interval if we have access to

\[
 n \geq \frac{\log(2/\delta)}{2\varepsilon^2}
\]

scheduling experiments per task, so \(\text{poly}(\delta^{-1}, \varepsilon^{-1}, \#\text{tasks})\).
Offline efficient PAC learnability

A more realistic model

What if the execution and inter-arrival probabilities were not known?

Efficiently PAC learnable (thanks Hoeffding!)

In the absence of hard tasks, all probabilities can be learned within an \((\varepsilon, \delta)\)-confidence interval if we have access to

\[
    n \geq \frac{\log(2/\delta)}{2\varepsilon^2}
\]

scheduling experiments per task, so \(\text{poly}(\delta^{-1}, \varepsilon^{-1}, \#\text{tasks})\).

- offline learning, and each scheduling experiment may take long
- online scheduling experiments are not safe in the presence of hard tasks
Deciding learnability

### Safe sampling (a.k.a. almost-sure Büchi)

1. The semantic MDP and the safe region are both end components.

2. The safe region contains a state from which full scheduling experiments are possible (for all tasks) iff we can almost surely carry out $\infty$-ly many experiments!
Safe sampling (a.k.a. almost-sure Büchi)

1. The semantic MDP and the safe region are both end components.

2. The safe region contains a state from which full scheduling experiments are possible (for all tasks) iff we can almost surely carry out infinitely many experiments!

- reaching the state may take exponential time in expectation…
Deciding learnability

Safe sampling (a.k.a. almost-sure Büchi)

1. The semantic MDP and the safe region are both end components.
2. The safe region contains a state from which full scheduling experiments are possible (for all tasks) iff we can almost surely carry out \(\infty\)-ly many experiments!

- reaching the state make take exponential time in expectation...

Efficient safe sampling (a.k.a. sure Büchi)

If the experiment state is surely reachable no matter what the environment does then we can carry out \(\infty\)-ly many experiments within linear steps of each other.
Experiments

- Deep RL’d strategy obtained via openAI’s baselines
- Safety shields obtained via AbsSynthe tool
Conclusions and future work

Conclusions
- We give a verification-friendly MDP-learning algorithm for mean payoff.
- We explored the limits of learning algorithms/strategies in unknown MDPs under (almost-)sure parity constraints.

Future work
- What happens if we remove some assumptions/inputs?
  - over/under-approximated support in our UAI'20 paper!
- Can we re-do everything with “model-free” learning strategies?
A PhD in related topics?

Algorithms and tools to verify timing and performance properties of AI-enabled controllers using:

- Monte-Carlo Tree Search
- Model-Predictive Control