Optimizing Local Satisfaction of Long-Run Average Objectives in Markov Decision Processes

David Klaška, Antonín Kučera, Vojtěch Kůr, Vít Musil, Vojtěch Řehák

Masaryk University, Brno, Czechia

david.klaska@mail.muni.cz, tony@fi.muni.cz, vojtech.kur@mail.muni.cz, musil@fi.muni.cz, rehak@fi.muni.cz

Abstract

Long-run average optimization problems for Markov decision processes (MDPs) require constructing policies with optimal steady-state behavior, i.e., optimal limit frequency of visits to the states. However, such policies may suffer from *local instability*, i.e., the frequency of states visited in a bounded time horizon along a run differs significantly from the limit frequency. In this work, we propose an efficient algorithmic solution to this problem.

Introduction

A *long-run average objective* for a Markov decision process (MDP) D is a property depending on the proportion of time (frequency) spent in the individual states of D. Typical examples of such properties include

- the total frequency of visits to "bad" states is ≤ 0.05;
- the state frequency vector is equal to a given vector ν.

The existing works on long-run average optimization (see Related Work) concentrate on constructing a strategy σ such that the Markov chain D^σ obtained by applying σ to D is irreducible and the *invariant* (also called *steady-state* (Norris 1998)) distribution $\mathbb{I}_\sigma$ achieves the objective. Unfortunately, the existing algorithms cannot influence the *local stability* of the invariant distribution along a run.

More concretely, for a given time horizon n, consider the *local frequency* Freq_n of states sampled from n consecutive states along a run, starting at a randomly chosen *pivot position* (we refer to Section for precise definitions). The local stability of the invariant distribution is the probability that Freq_n stays "close" to $\mathbb{I}_\sigma$. If the local stability is low, then the probability of achieving the considered objective *locally* (i.e., within the prescribed time horizon) is also low, and this may lead to severe problems in many application scenarios.

Example 1. Consider a system of Fig. 1(a) that can be either in the running (R) or maintenance (M) state. A long-run sustainability of the system requires that the system is running for 90% of time and the remaining 10% is spent on maintenance. Hence, we aim at constructing a strategy σ such that $\mathbb{I}_\sigma = \nu$, where $\nu(R) = 0.9$ and $\nu(M) = 0.1$. Ideally, the maintenance should be performed *regularly*, i.e., the state M should be visited once in 10 consecutive states. That is, Freq_{10} should be equal to ν with high probability.

For every $y \in [0, 1)$, the memoryless strategy σ_y of Fig. 1(b) satisfies $\mathbb{I}_{\sigma_y} = \nu$. However, the probability of $\mathit{Prob}^{\sigma_y}[\mathit{Freq}_{10}{=}\nu]$ approaches *zero* as $y \to 1$. The best result is achieved for $y = 0$, where this probability is ≈ 0.43. Hence, even the best memoryless strategy may considerably degrade the reliability of the system.

The simple deterministic strategy π of Fig. 1(c) satisfies $\mathbb{I}_\pi = \nu$ and $\mathit{Prob}^\pi[\mathit{Freq}_{10}{=}\nu] = 1$. Note that π needs 9 memory states to "count" the repeated visits to R before visiting M. A "tradeoff" between memory size and the local satisfaction of the sustainability objective is achieved by the strategy η of Fig. 1(d) where $\mathbb{I}_\eta = \nu$ and $\mathit{Prob}^\eta[\mathit{Freq}_{10}{=}\nu] \approx 0.74$. □

Other examples of long-run average objectives where the local satisfaction/stability requirements rise naturally are

- *critical supply delivery* (see, e.g., (Skwirzynski 1981; Lazar 1982)), where a bundle of items with limited lifespan should be delivered with a given frequency f. A high level of local instability of the frequency causes a high probability of early/late deliveries that are both undesirable (early deliveries lead to wasting the items that are not consumed before expiration, and late deliveries lead to a shortage of items).
- *dependability*, i.e., an upper bound on failure frequency (see, e.g., (Boussemart and Limnios 2004; Boussemart, Limnios, and Fillion 2002)). If this bound is locally violated with considerable probability, a user may interpret this as a violation of the dependability guarantee. For example, consider a device supposed to fail at most once in a month *on average* during the device lifetime. If the device fails twice in two weeks with probability 0.2 (which is possible *without* violating the guarantee on the long-run average failure frequency), the device is likely to be perceived as *unreliable*.

The above list of examples is not exhaustive. Scenarios documenting the importance of local satisfaction/stability can be found in every application area involving long-run average objectives.

Our Contribution Example 1 shows that optimizing the local satisfaction of long-run average objectives is non-

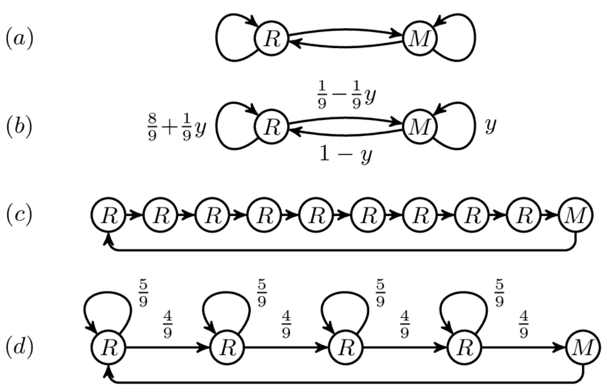

Figure 1: For the graph (a), the memoryless strategy σ_y of (b) achieves $\mathbb{I}_{\sigma_y} = \nu = (0.9, 0.1)$ for all $y \in [0, 1)$, but $\mathit{Prob}^{\sigma_y}[\mathit{Freq}_{10}{=}\nu] \leq 0.43$ for all $y \in [0, 1)$. The deterministic finite-memory strategy π of (c) achieves $\mathbb{I}_\pi = \nu$ and $\mathit{Prob}^\pi[\mathit{Freq}_{10}{=}\nu] = 1$ at the cost of large memory. The randomized finite-memory strategy η of (d) achieves $\mathbb{I}_\eta = \nu$ and $\mathit{Prob}^\eta[\mathit{Freq}_{10}{=}\nu] \approx 0.74$ with less memory.

trivial even for small *graphs* (i.e., MDPs with no probabilistic choice) and optimal strategies may require memory of considerable size. In this work, we formalize the notion of local satisfaction, examine its computational hardness, and design an efficient strategy synthesis algorithm for maximizing the local satisfaction of a given objective in a given MDP. The algorithm is evaluated on examples of nontrivial size. To the best of our knowledge, this is the first systematic study of the local stability of invariant distributions along runs in MDPs and the associated algorithmic problems. More concretely, our results can be summarized as follows:

I. We introduce an abstract class of long-run average objectives and precisely formulate the local optimization problem for a given objective and MDPs. We show that computing an optimal strategy is NP-hard even for *graphs*.

II. We design a dynamic algorithm LocalEval for evaluating the local satisfaction of a given objective *Obj* achieved by a given finite-memory strategy σ. We show that, on the one hand, LocalEval substantially outperforms a naive algorithm based on depth-first search, but, on the other hand, LocalEval is not sufficiently efficient for purposes of automatic differentiation and gradient descent.

III. We propose an efficient algorithm LocalSynt for synthesizing a finite-memory strategy σ maximizing the local satisfaction of a given *Obj* in a given MDP. LocalSynt is based on isolating three crucial features of σ that influence the local satisfaction of *Obj*:

F1. The "appropriateness" of $\mathbb{I}_\sigma$ for satisfying *Obj*.

F2. The "regularity" of σ, i.e., the stochastic stability of renewal times for certain families of states.

F3. The "level of determinism" of σ.

Subsequently, we design highly efficient evaluation functions for F1–F3 and optimize them jointly by gradient descent. We experimentally confirm the scalability of LocalSynt and the expected impact of different F1–F3 prioritization on the properties of the constructed strategies.

Related Work The *steady-state strategy synthesis problem*, i.e., the task of constructing a strategy for a given MDP achieving a given invariant distribution, has been solved in (Brázdil et al. 2011) (see also (Brázdil et al. 2014)) even for a more general class of multiple mean-payoff objectives. The constructed strategies may require infinite memory in general and can be computed in polynomial time. The problem of constructing a *memoryless randomized* strategy achieving a given steady-state distribution has been considered in (Akshay et al. 2013) for a subclass of *ergodic* MDPs and in (Velasquez 2019; Atia et al. 2020) for general MDPs. A polynomial-time strategy synthesis algorithm based on linear programming is given in both cases. The problem of computing a *deterministic* strategy achieving a given invariant distribution has been shown NP-hard and solvable by integer programming in (Velasquez et al. 2023). More recently, steady-state strategy synthesis under LTL constraints has been solved in (Křetínský 2021).

Optimizing *expected window mean-payoff* for MDP (Bordais, Guha, and Raskin 2019) is perhaps most related to the problem studied in this paper. Here, each MDP state is assigned a payoff collected when visiting the state. The task is to ensure that the average reward per visited state (mean-payoff) in a window of length ℓ sliding along a run reaches a given threshold within the window length. This can be seen as enforcing a form of "local stability" of the mean payoff along a run. The problem is solvable in time polynomial in the size of MDP and ℓ, and the algorithm relies on previous results achieved for 2-player games (Chatterjee et al. 2015). This technique is not applicable in our setting (recall that the studied problem is NP-hard even for graphs).

In a broader perspective, there are also works studying the trade-offs between the overall expected performance (mean payoff) and some forms of stability measured by variances of appropriate random variables (Brázdil et al. 2017).

The Model

We assume familiarity with basic notions of probability theory (probability distribution, expected value, conditional variance, etc.) and Markov chain theory. The set of all probability distributions over a finite set A is denoted by $\mathit{Dist}(A)$.

Markov chains A *Markov chain* is a triple $C = (S, \mathit{Prob}, \mu)$ where S is a finite set of states, $\mathit{Prob}\colon S \times S \to [0, 1]$ is a stochastic matrix such that $\sum_{s' \in S} \mathit{Prob}(s, s') = 1$ for every $s \in S$, and $\mu \in \mathit{Dist}(S)$ is an initial distribution.

A *run* of C is an infinite sequence $w = s_0, s_1, \ldots$ of states. We use $\mathbb{P}_\mu$ to denote the probability measure in the standard probability space over the runs of C determined by *Prob* and μ, and we use $\mathit{Init}(w)$ to denote the initial state of w (i.e., $\mathit{Init}(w) = s_0$).

Let $s, t \in S$. We say that t is *reachable* from s if the probability of visiting t from s is positive, i.e., $\mathit{Prob}^n(s, t) > 0$ for some $n \geq 0$ (recall that Prob^0 is the identity matrix).

Markov decision processes (MDPs) A *Markov decision process (MDP)*[1] is a triple $D{=}(V,E,p)$ where V is a finite set of *vertices* partitioned into subsets (V_N, V_S) of *non-deterministic* and *stochastic* vertices, $E \subseteq V{\times}V$ is a set of *edges* s.t. every vertex has at least one out-going edge, and $p\colon V_S{\to}\mathit{Dist}(V)$ is a *probability assignment* s.t. $p(v)(v'){>}0$ only if $(v,v') \in E$. We say D is a *graph* if $V_S{=}\emptyset$.

Outgoing edges in non-deterministic states are selected by a *strategy*. The most general type of strategy is a *history-dependent randomized (HR)* strategy where the selection is randomized and depends on the whole computational history. Since HR strategies require infinite memory, they are not apt for algorithmic purposes. Therefore, we restrict ourselves to a subclass of *finite-memory randomized (FR)* strategies introduced in the next paragraph.

FR strategies Let $D = (V,E,p)$ be an MDP and $M \neq \emptyset$ a finite set of *memory states*. Intuitively, memory states are used to "remember" some information about the sequence of previously visited vertices. For a given pair (v,m) where v is a currently visited vertex and m a current memory state, a strategy randomly selects a new pair (v',m') such that $(v,v') \in E$. In general, the new memory state m' may *not* be uniquely determined by the chosen v'. If v is stochastic, then v' is selected with probability $p(v)(v')$, and the strategy randomly selects the new memory state m'.

Formally, let $\alpha\colon V \to 2^M$ be a *memory allocation* assigning to every vertex v a non-empty subset of memory states available in V. Let $\overline{V} = \{(v,m) \mid v \in V, m \in \alpha(v)\}$ be the set of *augmented vertices*. A *finite-memory (FR) strategy* is a function $\sigma\colon \overline{V} \to \mathit{Dist}(\overline{V})$ such that for all $(v,m) \in \overline{V}$ where $v \in V_S$ and every $(v,v') \in E$ we have that

$$\sum_{m' \in \alpha(v')} \sigma(v,m)(v',m') \;=\; p(v)(v') \,.$$

An FR strategy is *memoryless* (or *Markovian*) if M is a singleton. In the following, we use $\overline{v}$ to denote an augmented vertex of the form (v,m) for some $m \in \alpha(v)$.

Every FR strategy σ together with a probability distribution $\mu \in \mathit{Dist}(\overline{V})$ determine the Markov chain $D^\sigma = (\overline{V}, \mathit{Prob}, \mu)$ where $\mathit{Prob}(\overline{v},\overline{u}) = \sigma(\overline{v})(\overline{u})$.

Invariant distributions Let $C = (S, \mathit{Prob}, \mu)$ be a Markov chain. A *bottom strongly connected component (BSCC)* of C is a maximal $B \subseteq S$ such that B is strongly connected and closed under reachable states, i.e., for all $s,t \in B$ and $r \in S$ we have that t is reachable from s, and if r is reachable from s, then $r \in B$.

Let B be a BSCC of C. For every $\nu \in \mathit{Dist}(B)$, let B^ν be the Markov chain $(B, \mathit{Prob}_B, \nu)$ where Prob_B is the restriction of Prob to $B{\times}B$. Furthermore, let $\mathbb{I}_B \in \mathit{Dist}(B)$ be the unique *invariant distribution* satisfying $\mathbb{I}_B = \mathbb{I}_B \cdot \mathit{Prob}_B$ (note that $\mathbb{I}_B$ is independent of ν). By ergodic theorem (Norris 1998), $\mathbb{I}_B$ is the limit frequency of visits to the states of B along a run in B^ν. More precisely, let $w = s_0, s_1, \ldots$ be a run of B^ν. For every $n \geq 1$, let $\mathit{Freq}_n(w)\colon B \to [0,1]$ be the state frequency vector computed for the prefix of w of length n, i.e., for every $s \in B$,

$$\mathit{Freq}_n(w)(s) = \#_s(s_0,\ldots,s_{n-1})/n$$

where $\#_s(s_0,\ldots,s_{n-1})$ is the number of occurrences of s in $s_0,\ldots,s_{n-1}$. Let $\mathit{Freq}(w) = \lim_{n\to\infty} \mathit{Freq}_n(w)$. If the limit does not exist, we put $\mathit{Freq}(w) = \vec{0}$. The ergodic theorem says that $\mathbb{P}^\nu[\mathit{Freq}{=}\mathbb{I}_B] = 1$.

Long-run average objectives Let $D = (V,E,p)$ be an MDP. A *long-run average objective* for D is a function $\mathit{Obj}\colon \mathit{Dist}(V) \to \mathbb{R}^{\geq 0}$. Intuitively, for a given frequency of visits to V, the value of *Obj* specifies the "badness" of the frequency, i.e., a higher value of $\mathit{Obj}(\mu)$ indicates that μ is "less appropriate" for achieving the objective encoded by *Obj*. Two representative examples are given below.

- For a given $\nu \in \mathit{Dist}(V)$, let $\mathit{Distance}_\nu(\mu) = \|\mu - \nu\|$, where $\|\cdot\|$ is a vector norm (such as L_1 or L_2). Hence, the objective $\mathit{Distance}_\nu$ corresponds to minimizing the distance from a desired frequency vector ν.
- For every $v \in V$, let $\kappa_v \subseteq [0,1]$ be an interval of admissible frequencies of visiting the vertex v. For example, if $\kappa_v = [0, 0.2]$, then v should be visited with frequency at most 0.2. For every $\mu \in \mathit{Dist}(V)$, we put $\mathit{Satisfy}_\kappa(\mu) = 0$ if $\mu(v) \in \kappa_v$ for all $v \in V$. Otherwise, $\mathit{Satisfy}_\kappa(\mu) = 1$. The objective $\mathit{Satisfy}_\kappa$ then corresponds to satisfying the constraints imposed by κ.

In some scenarios, the value of a long-run average objective depends only on the total frequency of visits to "equivalent" vertices. Formally, such equivalence is defined as a *labeling* $\mathcal{L}\colon V \to L$ where equivalent vertices share the same label, and a *labeled* long-run average objective is represented by a function $\mathcal{L}\text{-}\mathit{Obj}\colon \mathit{Dist}(L) \to \mathbb{R}^{\geq 0}$ specifying the "badness" of a given frequency of labels seen along a run. The function $\mathcal{L}$-*Obj* represents the unique objective $\mathit{Obj}\colon \mathit{Dist}(V) \to \mathbb{R}^{\geq 0}$ such that $\mathit{Obj}(\mu) = \mathcal{L}\text{-}\mathit{Obj}(\mu_{\mathcal{L}})$ where $\mu_{\mathcal{L}}(\ell) = \sum_{v \in \mathcal{L}^{-1}(\ell)} \mu(v)$.

In the following sections, we also apply *Obj* to distributions over augmented vertices $\overline{V}$. For every $\mu \in \mathit{Dist}(\overline{V})$, we put $\mathit{Obj}(\mu) = \mathit{Obj}(\nu)$, where $\nu \in \mathit{Dist}(V)$ is defined by $\nu(v) = \sum_{m \in \alpha(v)} \mu(v,m)$.

Local Frequency Measures Let $D = (V,E,p)$ be an MDP and *Obj* a long-run average objective for D.

The "global" satisfaction of *Obj* achieved by an FR strategy σ is measured by $\min_B \mathit{Obj}(\mathbb{I}_B)$ where B ranges over the BSCCs of D^σ. As we already noted in Example 1, it may happen that an FR strategy achieves the optimal $\mathit{Obj}(\mathbb{I}_B)$, but the expected value of *Obj* for a *local* frequency of states sampled from n consecutive states along a run is large. The *local satisfaction* of *Obj* is measured by the *expected badness of the local frequency* defined in the next paragraph.

Let σ be a FR strategy, B a BSCC of D^σ, and μ_B an initial distribution over B. Consider the local frequency sampled from n consecutive states along a run in B, where the sampling starts in a randomly chosen *pivot* state p. The probability of $p = s$ for a given $s \in B$ corresponds to the "global"

[1] Our definition of MDPs is standard in the area of graph games. It is equivalent to the "classical" MDP definition where *actions* are used instead of stochastic vertices (see, e.g., (Puterman 1994)). For our purposes, the adopted definition is more convenient and leads to substantially simpler notation.

frequency of s in a run, which is equal to $\mathbb{I}_B(s)$ independently of μ_B. Hence, the conditional expected badness of the local frequency under the condition $p = s$ is equal to $\mathbb{E}^{\mu_s}[\mathit{Obj}(\mathit{Freq}_n)]$ where μ_s is a distribution over B such that $\mu_s(s) = 1$ and $\mu_s(t) = 0$ for all $t \neq s$. Hence, the *expected badness of the local frequency* is defined as

$$\sum_{s\in B} \mathbb{I}_B(s)\cdot \mathbb{E}^{\mu_s}[\mathit{Obj}(\mathit{Freq}_n)] \quad = \quad \mathbb{E}^{\mathbb{I}_B}[\mathit{Obj}(\mathit{Freq}_n)]$$

We intuitively expect that $\mathbb{E}^{\mathbb{I}_B}[\mathit{Obj}(\mathit{Freq}_n)]$ decreases with increasing time horizon n. This holds if n is increased by a *sufficiently large* $k > 0$. However, for $k = 1$, it may happen that $\mathbb{E}^{\mathbb{I}_B}[\mathit{Obj}(\mathit{Freq}_n)]$ *increases*. We fix this inconvenience by adopting the following definition:

$$\mathit{L\text{-}Badness}^\sigma(\mathit{Obj}, d) \;=\; \min_B \min_{n\le d} \mathbb{E}^{\mathbb{I}_B}[\mathit{Obj}(\mathit{Freq}_n)]$$

That is, for every $d \geq 1$, we consider the best outcome achievable for a time horizon of size *at most* d in a BSCC B of D^σ. Note that $\mathit{L\text{-}Badness}^\sigma(\mathit{Obj}, d)$ is non-increasing in d.

The next theorem shows that the problem of computing an FR strategy σ minimizing $\mathit{L\text{-}Badness}^\sigma(\mathit{Obj}, d)$ is computationally hard even for *graphs* (MDPs with no stochastic vertices) where an optimal FR strategy does not require randomization. A proof is in (Klaška et al. 2023).

Theorem 1. *Let $D = (V, E, p)$ be a graph (i.e., $V_S = \emptyset$), $d \in \mathbb{N}$, and $\nu \in \mathit{Dist}(V)$. The existence of a FR strategy σ such that $\mathbb{P}_{\mathbb{I}_B}[\mathit{Freq}_n{=}\nu] = 1$ for some $n \leq d$ and a BSCC B of D^σ is NP-hard.*

The NP-hardness holds even under the assumption that if such a σ exists, it can be constructed so that $\sigma(\overline{v})$ is a Dirac distribution for every $\overline{v} \in \overline{V}$.

Note that Theorem 1 implies NP-hardness of minimizing $\mathit{L\text{-}Badness}^\sigma(\mathit{Obj}, d)$ for $\mathit{Distance}_\nu$ and $\mathit{Satisfy}_\kappa$, because $\mathbb{P}_{\mathbb{I}_B}[\mathit{Freq}_n{=}\nu] = 1$ iff $\mathit{L\text{-}Badness}^\sigma(\mathit{Distance}_\nu, d) = 0$ iff $\mathit{L\text{-}Badness}^\sigma(\mathit{Satisfy}_\kappa, d) = 0$ where $\kappa(v) = [\nu(v), \nu(v)]$ for every $v \in V$.

Evaluating Local Badness

In this section, we design algorithm LocalEval for evaluating $\mathit{L\text{-}Badness}^\sigma(\mathit{Obj}, d)$.

Let $D = (V, E, p)$ be an MDP, σ an FR strategy for D, and $\mathcal{L}\colon V \to L$ a labeling. Furthermore, let $\mathcal{L}\text{-}\mathit{Obj}\colon \mathit{Dist}(L) \to \mathbb{R}^{\geq 0}$ be the desired objective function. Algorithm LocalEval consists of several phases, following the definition of $\mathit{L\text{-}Badness}^\sigma(\mathit{Obj}, d)$: First, we use Tarjan's algorithm (Tarjan 1972) to identify all BSCCs of D^σ. For each BSCC B, the invariant distribution $\mathbb{I}_B$ is computed via the following system of linear equations: For each $\overline{v} \in B$, we have a fresh variable $z_{\overline{v}}$ and equations expressing that $z = z \cdot \mathit{Prob}_B$ and $\sum_{\overline{v}\in B} z_{\overline{v}} = 1$. The vector $\mathbb{I}_B$ is the unique solution of this system.

The core of LocalEval is Algorithm 1 computing $\mathbb{E}^{\mathbb{I}_B}[\mathit{Obj}(\mathit{Freq}_n) \mid \mathit{Init}{=}\overline{v}]$ for all $\overline{v} \in B$ and $n \leq d$ by dynamic programming. Since for all $n \leq d$ we have that

$$\mathbb{E}^{\mathbb{I}_B}[\mathit{Obj}(\mathit{Freq}_n)] = \sum_{\overline{v}\in B} \mathbb{I}_B(\overline{v})\cdot\mathbb{E}^{\mathbb{I}_B}[\mathit{Obj}(\mathit{Freq}_n) \mid \mathit{Init}{=}\overline{v}],$$

the computation of $\mathit{L\text{-}Badness}^\sigma(\mathit{Obj}, d)$ is straightforward.

Algorithm 1: The core procedure of LocalEval

```
for $\overline{v_0} \in B$ do
    $s_0.\overline{v} = \overline{v_0}$
    $s_0.vec = \{0, \ldots, 0\}$
    $s_0.vec[\mathcal{L}(v_0)]$++
    $cur\_map[s_0] = 1.$
    for $n \in \{1, \ldots, d\}$ do
        for $(s, p) \in cur\_map$ do
            $rsl[\overline{v_0}][n]$ += $p \cdot \mathcal{L}\text{-}\mathit{Obj}(s.vec/n)$
        if $n < d$ then
            for $(s, p) \in cur\_map$ do
                for $\overline{v} \in B$ do
                    $s' = s$
                    $s'.\overline{v} = \overline{v}$
                    $s'.vec[\mathcal{L}(v)]$++
                    $next\_map[s']$ += $p \cdot \sigma[s.\overline{v}][\overline{v}]$
            $swap(cur\_map, next\_map)$
            $next\_map.clear()$
```

Algorithm 1 uses two associative arrays (e.g., C++ unordered_map), called cur_map and $next\_map$, to gather information about the probabilities of individual paths. More specifically, the maps are indexed by *states*, where a state consists of an augmented vertex $\overline{v} \in B$, corresponding to the last vertex of a path, and a vector vec of $|L|$ integers, corresponding to the numbers of visits to particular labels. The value associated to a state s is the total probability of all paths corresponding to s. The values $\mathbb{E}^{\mathbb{I}_B}[\mathit{Obj}(\mathit{Freq}_n) \mid \mathit{Init}{=}\overline{v}]$ are gathered in a 2-dimensional array rsl. Further details are given in (Klaška et al. 2023).

Optimizing Local Badness

In this section, we design an algorithm LocalSynt for constructing an FR strategy σ with memory M minimizing $\mathit{L\text{-}Badness}^\sigma(\mathit{Obj}, d)$ for a given MDP D. The main idea behind LocalSynt is to construct and optimize a function *simultaneously* rewarding the following features of σ:

F1. Global satisfaction of *Obj*;

F2. Stochastic stability of renewal times for families of augmented vertices with the same label.

F3. The level of determinism achieved by σ.

Intuitively, F1 ensures that σ achieves *Obj* globally, and F2 in combination with F3 "encourage" the features of σ causing a small difference between the global and the local satisfaction. To understand the significance of F2, realize that the frequency of visits to augmented vertices with the same label is the inverse of the expected *renewal time* for this family. Hence, the local stability of the frequency of visits can be achieved by maximizing the stochastic stability (i.e., minimizing the standard deviation of) the renewal time. To understand the significance of F3, realize that for every *deterministic* FR strategy σ, the value of $\mathit{L\text{-}Badness}^\sigma(\mathit{Obj}, d)$ is equal to $\min_B \mathit{Obj}(\mathbb{I}_B)$ for a *sufficiently large* d. Hence, putting more emphasis on F3 yields strategies where $\sigma(\overline{v})$ is close to a Dirac distribution for many $\overline{v}$, which may be advantageous when d is high.

Measuring F1–F3

In this section, we design efficient measures for F1–F3 and combine these measures into a single function $Comb$.

Let $D = (V, E, p)$ be an MDP, $\mathcal{L}\colon V \to L$ a labeling, and Obj a long-run average objective for D. Furthermore, let σ be an FR strategy with memory M and B a BSCC of D^σ. Recall that $\mathbb{I}_B$ is the invariant distribution of B, and $B^{\mathbb{I}_B}$ is the Markov chain determined by B and the initial distribution $\mathbb{I}_B$. Furthermore, let $RT(w)$ be the least $i \geq 1$ such that $\mathcal{L}(\overline{v}_i) = \mathcal{L}(\overline{v}_0)$. If there is no such i, we put $RT(w) = \infty$. Hence, $RT(w)$ is the number of edges needed to visit an augmented vertex with the same label as $\mathit{Init}(w)$ (i.e., the Renewal Time to the initial label).

Measuring F1 The global *dissatisfaction* of Obj achieved by σ in $B^{\mathbb{I}_B}$ is measured by $\mathit{Obj}(\mathbb{I}_B)$.

Measuring F2 Let $\mathrm{Var}^{\mathbb{I}_B}[RT \mid \mathcal{L}(\mathit{Init}){=}\ell]$ be the conditional variance of the Renewal Time to the initial label under the condition that a run is initiated in an augmented vertex with label ℓ. If the probability of $\mathcal{L}(\mathit{Init}){=}\ell$ is zero, i.e, $\mathbb{I}_B$ assigns zero to all augmented vertices with label ℓ, we treat $\mathrm{Var}^{\mathbb{I}_B}[RT \mid \mathcal{L}(\mathit{Init}){=}\ell]$ as zero. Furthermore, we define the corresponding standard deviation

$$\mathit{SD}(\ell) = \sqrt{\mathrm{Var}^{\mathbb{I}_B}[RT \mid \mathcal{L}(\mathit{Init}){=}\ell]}\,.$$

Stochastic *instability* of renewal times caused by σ in $B^{\mathbb{I}_B}$ is measured by the function

$$\mathit{Penalty}_1(\sigma, B) = \sum_{\ell \in L} \mathbb{I}_B(\ell) \cdot \mathit{SD}(\ell)$$

where $\mathbb{I}_B(\ell)$ is the sum of all $\mathbb{I}_B(\overline{v})$ where $\overline{v} \in B$ and $\mathcal{L}(v) = \ell$. That is, $\mathit{Penalty}_1$ is the weighted sum of all $\mathit{SD}(\ell)$ where the weights correspond to the limit label frequencies.

Measuring F3 The level of *non-determinism* caused by σ in $B^{\mathbb{I}_B}$ is measured by the stochastic instability of Renewal Times separately for each augmented vertex. That is, we put

$$\mathit{Penalty}_2(\sigma, B) = \sum_{\overline{v} \in B} \mathbb{I}_B(\overline{v}) \cdot \sqrt{\mathrm{Var}^{\mathbb{I}_B}[RT \mid \mathit{Init}{=}\overline{v}]}$$

If the probability of $\mathit{Init}{=}\overline{v}$ is zero, we treat the corresponding conditional variance as zero.

Note that for every *deterministic* strategy we have that $\mathrm{Var}^{\mathbb{I}_B}[RT \mid \mathit{Init}{=}\overline{v}] = 0$ for every $\overline{v}$, i.e., $\mathit{Penalty}_2 = 0$. However, $\mathit{Penalty}_1$ is still positive if the expected renewal times for the individual augmented vertices with the same label differ. The only "degenerated" case when $\mathit{Penalty}_1$ and $\mathit{Penalty}_2$ are the same functions is when all vertices have pairwise different labels and every vertex is allocated just one memory state.

Combining the measures Our LocalSynt algorithm attempts to *minimize* the following function $Comb(\sigma)$ over all BSCC B of D^σ:

$$(1{-}\beta{-}\gamma)\mathit{Obj}(\mathbb{I}_B){+}\beta{\cdot}c_1{\cdot}\mathit{Penalty}_1(\sigma,B){+}\gamma{\cdot}c_2\mathit{Penalty}_2(\sigma,B)$$

where $\beta, \gamma \in [0, 1]$ are *weights* such that $\beta + \gamma < 1$ representing the preference among F1–F3. Since the values of $\mathit{Obj}(\mathbb{I}_B)$ may range over very different intervals than $\mathit{Penalty}_1$ and $\mathit{Penalty}_2$, we also use the *normalizing constants* $c_1 = (\mathit{Obj}(\mathbb{I}_B){+}1)/(\mathit{Penalty}_1(\sigma, B){+}1)$ and $c_2 = (\mathit{Obj}(\mathbb{I}_B){+}1)/(\mathit{Penalty}_2(\sigma, B){+}1)$.

Algorithm 2: LocalSynt

```
SolutionParameters ← $RandomInit$
for $i \in \{1, \ldots, \mathrm{Steps}\}$ do
    $\sigma \leftarrow \mathit{Softmax}(\mathrm{SolutionParameters})$
    $Comb(\sigma) \leftarrow \mathit{EvaluateComb}(\sigma)$
    $\nabla Comb(\sigma) \leftarrow \mathit{Gradient}(\sigma)$
    SolutionParameters += $\mathit{Step}(\nabla Comb(\sigma))$
    Save $Comb(\sigma), \sigma$
return $\sigma$ with the least $Comb(\sigma)$
```

Computing $Comb$

In this section, we show that there exist three efficiently constructible systems of linear equations with unique solutions $\vec{x}$, $\vec{y}$ and $\vec{z}$ such that the function $Comb$ is a closed-form expression over the components of $\vec{x}$, $\vec{y}$, and $\vec{z}$ containing only differentiable functions. This allows us to compute the *gradient* of $Comb$ efficiently and apply state-of-the-art methods of differentiable programming to minimize $Comb$ by gradient descent, which is the essence of LocalSynt functionality.

For every $\overline{v} \in B$ and $\ell \in L$ such that $\mathbb{I}_B(\ell) > 0$, let $x_{\overline{v},\ell}$ and $y_{\overline{v},\ell}$ be fresh variables. For every $x_{\overline{v},\ell}$, we add an equation

$$x_{\overline{v},\ell} = \begin{cases} 0 & \text{if } \mathcal{L}(v) = \ell, \\ 1 + \sum_{\overline{u} \in B} \sigma(\overline{v})(\overline{u}) \cdot x_{\overline{u},\ell} & \text{otherwise.} \end{cases}$$

Then the system has a unique solution $\vec{x}$ where $\vec{x}_{\overline{v},\ell}$ is the expected time for visiting an ℓ-labeled augmented vertex from $\overline{v}$. Hence, $\mathbb{E}^{\mathbb{I}_B}[RT \mid \mathit{Init} = \overline{v}] = 1 + \sum_{\overline{u} \in B} \sigma(\overline{v})(\overline{u}) \cdot \vec{x}_{\overline{u},\ell}$, where $\ell = \mathcal{L}(v)$.

Similarly, for every $y_{\overline{v},\ell}$, we add an equation

$$y_{\overline{v},\ell} = \begin{cases} 0 & \text{if } \mathcal{L}(v) = \ell, \\ 1 + \sum_{\overline{u} \in B} \sigma(\overline{v})(\overline{u}) \cdot (2\vec{x}_{\overline{u},\ell} + y_{\overline{u},\ell}) & \text{otherwise.} \end{cases}$$

Note that the above equation is *linear* and uses components of $\vec{x}$ in the coefficients. The system has a unique solution $\vec{y}$ where $\vec{y}_{\overline{v},\ell}$ is the expected *square* of the time for visiting an ℓ-labeled augmented vertex from $\overline{v}$. Hence,

$$\mathbb{E}^{\mathbb{I}_B}[RT^2 \mid \mathit{Init} = \overline{v}] = 1 + \sum_{\overline{u} \in B} \sigma(\overline{v})(\overline{u}) \cdot (2\vec{x}_{\overline{u},\ell} + \vec{y}_{\overline{u},\ell})$$

The vector $\vec{z}$ corresponding to the invariant distribution $\mathbb{I}_B$ is computed the same way as in LocalEval.

Both $\mathbb{E}^{\mathbb{I}_B}[RT \mid \mathcal{L}(\mathit{Init}){=}\ell]$ and $\mathbb{E}^{\mathbb{I}_B}[RT^2 \mid \mathcal{L}(\mathit{Init}){=}\ell]$ are weighted sums of $\mathbb{E}^{\mathbb{I}_B}[RT \mid \mathit{Init} = \overline{v}]$ and $\mathbb{E}^{\mathbb{I}_B}[RT^2 \mid \mathit{Init} = \overline{v}]$ where the weights are expressions over the components of $\vec{z}$. Since $\mathrm{Var}[X \mid Y] = \mathbb{E}[X^2 \mid Y] - \mathbb{E}^2[X \mid Y]$ for all random variables X, Y, the conditional variances $\mathrm{Var}^{\mathbb{I}_B}[RT \mid \mathit{Init}{=}\overline{v}]$ and $\mathrm{Var}^{\mathbb{I}_B}[RT \mid \mathcal{L}(\mathit{Init}){=}\ell]$ are also expressible as closed form expressions over $\vec{x}$, $\vec{y}$, and $\vec{z}$. Hence, $Comb$ also has this property.

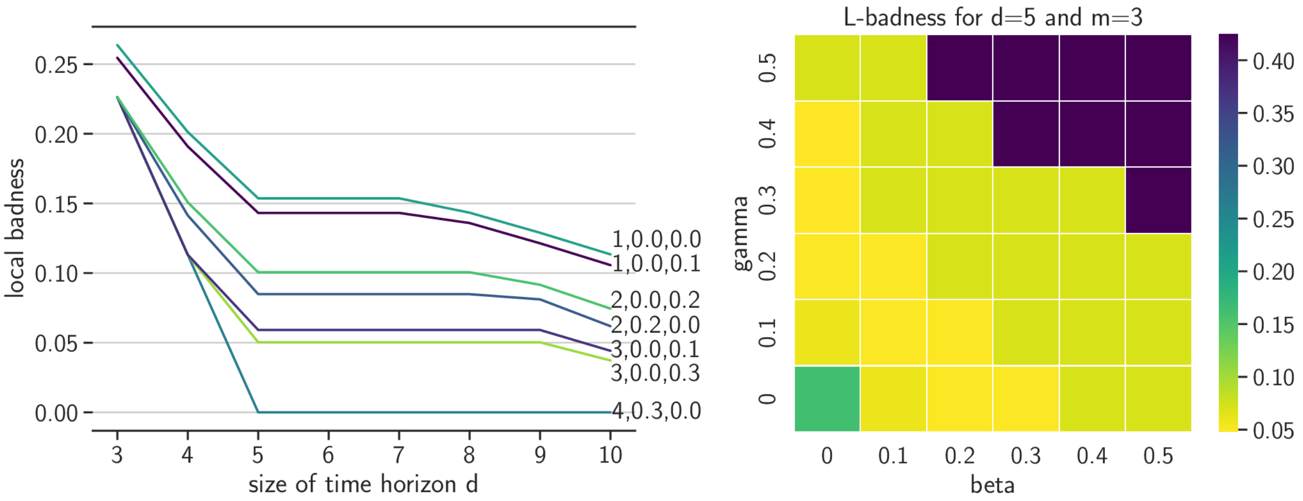

Figure 2: Strategies constructed for the graph of Fig. 1. Adding more memory to the state R helps. Best results are achieved for certain combinations of β and γ where $\beta + \gamma$ does not exceed the threshold around 0.6.

The LocalSynt Algorithm

Our algorithm is based on differentiable programming and gradient descent, and it performs the standard optimization loop shown in Algorithm 2. For every pair of augmented vertices $(\overline{v}, \overline{u})$ such that $(v, u) \in E$, we need a parameter representing $\sigma(\overline{v})(\overline{u})$. Note that if v is stochastic, then the parameter actually represents the probability of selecting the memory state of $\overline{u}$. These parameters are initialized to random values sampled from *LogUniform* distribution (so that we impose no prior knowledge about the solution). Then, they are transformed into probability distributions using the standard *Softmax* function.

The crucial ingredient of LocalSynt is the procedure *EvaluateComb* for computing the value of $Comb$ for the strategy represented by the parameters. This procedure allows to compute $Comb(\sigma)$, and also the gradient of $Comb(\sigma)$ at the point corresponding to σ by automatic differentiation. After that, we update the point representing the current σ in the direction of the steepest descent. The intermediate solutions and the corresponding $Comb$ values are stored, and the best solution found within Steps optimization steps is returned. Our implementation uses PyTorch framework (Paszke et al. 2019) and its automatic differentiation with Adam optimizer (Kingma and Ba 2015)).

Observe that LocalSynt is equally efficient for general MDPs and graphs. The only difference is that stochastic vertices generate fewer parameters.

Experiments

The system setup was as follows: CPU: AMD Ryzen 93900X (12 cores); RAM: 32GB; Ubuntu 20.04. To separate the probabilistic choice introduced by the constructed strategies from the internal probabilistic choice performed in stochastic vertices, we perform our experiments on graphs.

Experiment I

In our first experiment, we aim to analyze the impact of the β, γ coefficients in $Comb$ and the size of available memory on the structure and performance of the resulting strategy σ.

We use the graph D of Fig. 1(a) and the objective $Distance_\nu$ with L_2 norm where $\nu(R) = \frac{4}{5}$ and $\nu(M) = \frac{1}{5}$. In our FR strategies, we allocate $m \leq 4$ memory states to the vertex R and one memory state to the vertex M. The coefficients β, γ range over $[0, 0.5]$ with a discrete step 0.1. For every choice of β, γ, and m, we run LocalSynt 40 times with Steps set to 800 and return the strategy σ with the *least* value of $Comb$ found. Then, we use the LocalEval algorithm to compute *L-Badness*$^\sigma(Distance_\nu, d)$ for $d \in \{3, \dots, 10\}$.

Discussion The plot of Fig. 2 (left) shows that

1. increasing the size of memory m leads to better performance (smaller *L-Badness*$^\sigma(Distance_\nu, d)$);
2. setting $\beta{=}\gamma{=}0$ produces worse strategies (for every m) than setups with even small positive values of β, γ;
3. setting $\beta + \gamma \geq 0.5$ leads to very bad strategies.

The outcomes 1. and 2. are in full accordance with the intuition presented in the section "Optimizing Local Badness". Outcome 3. is also easy to explain—when β or γ is too large, the algorithm LocalSynt concentrates on maximizing stochastic stability of renewal times or achieving determinism and "ignores" the L_2 distance from ν. For example, the worst strategy of Fig. 2 (left) obtained for $m = 2$, $\beta = 0.5$, $\gamma = 0$ "regularly alternates" between R and M, i.e., the renewal times of R and M are equal to 2 and have *zero variance*. This leads to local frequency $(\frac{1}{2}, \frac{1}{2})$, which is "far" from the desired ν.

We also provide plots of *L-Badness*$^\sigma(Distance_\nu, d)$ where m and d are fixed and β, γ range over $[0, 0.5]$. The plot for $d = 5$ and $m = 3$ is shown in Fig. 2 (right). All these plots (see (Klaška et al. 2023)) consistently show that

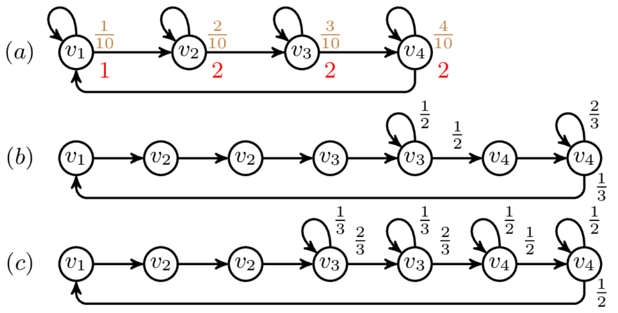

Figure 3: The structure of D_4 (a), π_4 (b), and ϱ_4 (c).

the best outcomes are achieved for certain combinations of β and γ where $\beta+\gamma$ is positive but below 0.6 (in Fig. 2 (right), the best outcomes are in yellow).

Experiment II

Here we aim to analyze the scalability of LocalEval and LocalSynt and demonstrate that LocalSynt can produce sophisticated strategies for instances of non-trivial size. Since the running time of LocalSynt depends on the *number of parameters*, i.e., the number of augmented edges, we need to consider a scalable instance.

For every $n \geq 2$, let D_n be a graph with vertices $v_1, \ldots, v_n$ and edges (v_i, v_i) and $(v_i, v_{(i \bmod n)+1})$ for every $i \leq n$. Every v_i is assigned $\min\{i, \lceil \frac{n}{2} \rceil\}$ memory states. The desired frequency ν is defined by $\nu(v_i) = i/s$ where $s = \frac{n(n+1)}{2}$. The structure of D_4 is shown in Fig. 3(a), together with ν (brown) and memory allocation (red).

We consider the objective $Distance_\nu$ with the L_2 norm, and we aim to optimize $\textit{L-Badness}^\sigma(Distance_\nu, d)$ where $d = s$ (the least d such that ν is achievable in d consecutive states.) To evaluate the scalability of LocalEval and LocalSynt we run LocalEval 5 times for different choice of β and γ and evaluate $\textit{L-Badness}^\sigma(Distance_\nu, d)$ using LocalEval and also a naive algorithm based on depth-first search (see (Klaška et al. 2023) for a more detailed description of the naive algorithm). In Table 2, for every n we report the number of parameters, the size of d, the average time of one Step of LocalSynt (i.e., one iteration of the main **for** loop of LocalSynt), one run of LocalEval, and one run of the naive evaluation algorithm (in secs).

To evaluate the quality of strategies constructed by LocalSynt, we consider two natural strategies π_n and ϱ_n (see Fig. 3 (b) and (c)). Both strategies perform an "ideal" number of self-loops on $v_1, \ldots, v_{\lceil n/2 \rceil}$ where the memory suffices. On the other vertices, π_n performs $\lceil n/2 \rceil - 1$ self-loops deterministically and then selects randomly between the self-loop and the edge to the next vertex, while ϱ_n performs a random choice in every visit. The probabilities are computed so that $\mathbb{I} = \nu$. Hence, both π_n and ϱ_n represent an "educated guess" for a high-quality strategy.

For all $n \in \{2, \ldots, 8\}$, we run LocalSynt 40 times with Steps set to 800 for all $\beta, \gamma \in \langle 0, 0.5 \rangle$ with a discrete step 0.1, always collecting the strategy σ_n with the minimal *Comb* value. The outcomes are shown in Table 1. Interestingly, σ_n *significantly outperforms* both π_n and ϱ_n for all $n \geq 4$. The strategy σ_n cannot be found "ad-hoc"; in most cases, the associated invariant distribution is different from ν, which means that global satisfaction "traded" for local satisfaction. We also report the values of β and γ for which the best strategy σ_n was found by LocalSynt.

	$\textit{L-Badness}(Distance_\nu, d)$				
n	π_n	ϱ_n	σ_n	β	γ
2	0.15713	0.15713	0.15713	0.2	0.0
3	0.11479	0.10255	0.11473	0.1	0.1
4	0.19416	0.17131	0.10540	0.0	0.2
5	0.14277	0.11762	0.10540	0.0	0.2
6	0.17491	0.13985	0.08016	0.0	0.2
7	0.13781	0.10456	0.10022	0.0	0.2
8	0.15609	0.11436	0.10012	0.0	0.2

Table 1: Strategy σ_n outperforms π_n and ϱ_n.

n	*Par*	d	*Step*	LocalEval	*Naive*
4	25	10	2.12E-03	2.21E-04	2.74E-03
5	61	15	2.71E-03	4.56E-03	1.21E+02
6	79	21	2.21E-03	4.13E-01	timeout
7	150	28	2.44E-03	1.98E+01	timeout
8	182	36	2.50E-03	timeout	timeout
10	350	55	2.97E-03	timeout	timeout
12	599	78	6.43E-03	timeout	timeout
14	945	105	1.88E-02	timeout	timeout
16	1404	136	3.91E-02	timeout	timeout
18	1992	171	1.05E-01	timeout	timeout
20	2725	210	2.15E-01	timeout	timeout

Table 2: Running times in *seconds*, timeout = 900 secs.

Discussion Table 2 shows that LocalSynt can easily process instances with thousands of parameters, while the scalability limits of LocalEval are reached for $d \approx 30$. Hence, LocalEval cannot be used for strategy synthesis based on gradient descent because LocalEval would have to be invoked hundreds of times in a single run. Table 1 shows that LocalSynt can construct sophisticated strategies for non-trivial instances. Details are in (Klaška et al. 2023).

Conclusions

The results demonstrate that non-trivial instances of the local satisfaction problem for long-run average objectives can be solved efficiently despite the NP-hardness of this problem. Experiment II also shows that the best strategy for D_n is obtained by setting $\beta = 0.0$ and $\gamma = 0.2$. Although LocalEval cannot evaluate the strategies obtained for large n's, there is a good chance that these strategies are better than the ones constructed ad-hoc. This indicates how to overcome the scalability issues for other parameterized instances.

Acknowledgments

Research was sponsored by the Army Research Office and accomplished under Grant Number W911NF-21-1-0189. *Disclaimer.* The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

Vojtěch Kůr received funding from the European Union's Horizon Europe program under the Grant Agreement No. 101087529. Vít Musil was supported by the Czech Science Foundation grant GA23-06963S.

References

Akshay, S.; Bertrand, N.; Haddad, S.; and Hélouët, L. 2013. The Steady-State Control; Problem for Markov Decision Processes. In *Proceedings of 10th Int. Conf. on Quantitative Evaluation of Systems (QEST'13)*, volume 8054 of *Lecture Notes in Computer Science*, 290–304. Springer.

Atia, G.; Beckus, A.; Alkhouri, I.; and Velasquez, A. 2020. Steady-State Policy Synthesis in Multichain Markov Decision Processes. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2020)*, 4069–4075.

Bordais, B.; Guha, S.; and Raskin, J.-F. 2019. Expected Window Mean-Payoff. In *Proceedings of FST&TCS 2019*, volume 150 of *Leibniz International Proceedings in Informatics*, 32:1–32:15. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

Boussemart, M.; and Limnios, N. 2004. Markov Decision Processes with Asymptotic Average Failure Rate Constraint. *Communications in Statistics – Theory and Methods*, 33(7): 1689–1714.

Boussemart, M.; Limnios, N.; and Fillion, J. 2002. Non-Ergodic Markov Decision Processes with a Constraint on the Asymptotic Failure Rate: General Class of Policies. *Stochastic Models*, 18(1): 173–191.

Brázdil, T.; Brožek, V.; Chatterjee, K.; Forejt, V.; and Kučera, A. 2011. Two Views on Multiple Mean-Payoff Objectives in Markov Decision Processes. In *Proceedings of LICS 2011*. IEEE Computer Society Press.

Brázdil, T.; Brožek, V.; Chatterjee, K.; Forejt, V.; and Kučera, A. 2014. Markov Decision Processes with Multiple Long-run Average Objectives. *Logical Methods in Computer Science*, 10(1): 1–29.

Brázdil, T.; Chatterjee, K.; Forejt, V.; and Kučera, A. 2017. Trading performance for stability in Markov decision processes. *Journal of Computer and System Sciences*, 84: 144–170.

Chatterjee, K.; Doyen, L.; Randour, M.; and Raskin, J.-F. 2015. Looking at Mean-Payoff and Total-Payoff through Windows. *Information and Computation*, 242: 25–52.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of ICLR 2015*.

Klaška, D.; Kučera, A.; Kůr, V.; Musil, V.; and Řehák, V. 2023. Optimizing Local Satisfaction of Long-Run Average Objectives in Markov Decision Processes. arXiv:2312.12325.

Křetínský, J. 2021. LTL-Constrained Steady-State Policy Synthesis. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2021)*, 4104–4111.

Lazar, A. 1982. Optimal Flow Control of a Class of Queueing Networks in Equilibrium. *IEEE Transactions on Automatic Control*, 28(11): 1001–1007.

Norris, J. 1998. *Markov Chains*. Cambridge University Press.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, 8024–8035. Curran Associates, Inc.

Puterman, M. 1994. *Markov Decision Processes*. Wiley.

Skwirzynski, J. 1981. New Concepts in Multi-User Communication. *Springer Science & Business Media*, 43.

Tarjan, R. 1972. Depth-First Search and Linear Graph Algorithms. *SIAM Journal of Computing*, 1(2).

Velasquez, A. 2019. Steady-State Policy Synthesis for Verifiable Control. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2019)*, 5653–5661.

Velasquez, A.; Alkhouri, I.; Subramani, K.; Wojciechowski, P.; and Atia, G. 2023. Optimal Deterministic Controller Synthesis from Steady-State Distributions. *Journal of Automated Reasoning*, 67(7).