Computational Approaches for Stochastic Shortest Path on Succinct MDPs

Krishnendu Chatterjee[1] Hongfei Fu[2] Amir Goharshady[1]
Nastaran Okati[3]

[1]IST Austria

[2]Shanghai Jiao Tong University

[3]Ferdowsi University of Mashhad

IJCAI 2018

Succinct MDPs

Succinct MDPs

- A *succinct* MDP is an MDP described implicitly by:

Succinct MDPs

- A *succinct* MDP is an MDP described implicitly by:
  - a set of variables,

Succinct MDPs

- A *succinct* MDP is an MDP described implicitly by:
  - a set of variables,
  - a set of rules that describe how the variables can be updated,

Succinct MDPs

- A *succinct* MDP is an MDP described implicitly by:
  - a set of variables,
  - a set of rules that describe how the variables can be updated,
  - a target set, consisting of valuations to the variables.

Succinct MDPs

- A *succinct* MDP is an MDP described implicitly by:
  - a set of variables,
  - a set of rules that describe how the variables can be updated,
  - a target set, consisting of valuations to the variables.

  At every time step, a rule is non-deterministically chosen to update the variables. This process continues until the target set is reached.

Succinct MDPs

- A *succinct* MDP is an MDP described implicitly by:
  - a set of variables,
  - a set of rules that describe how the variables can be updated,
  - a target set, consisting of valuations to the variables.

  At every time step, a rule is non-deterministically chosen to update the variables. This process continues until the target set is reached.
- We can think of a succinct MDP as a probabilistic program with the following form:

$$\textbf{while } \phi \textbf{ do } Q_1 \square \ldots \square Q_k \textbf{ od}$$

where $\square$ denotes non-determinism and each Q_i is a sequence of assignments to variables.

Succinct MDPs

- A *succinct* MDP is an MDP described implicitly by:
  - a set of variables,
  - a set of rules that describe how the variables can be updated,
  - a target set, consisting of valuations to the variables.

  At every time step, a rule is non-deterministically chosen to update the variables. This process continues until the target set is reached.
- We can think of a succinct MDP as a probabilistic program with the following form:

$$\textbf{while } \phi \textbf{ do } Q_1 \Box \ldots \Box Q_k \textbf{ od}$$

  where $\Box$ denotes non-determinism and each Q_i is a sequence of assignments to variables.
- Example: **while** $x \geq 1$ **do** $x := x + r \Box x := x - 1$ **od**

Another Example

```
while x ≥ 1 do
  □ if (0.4) { x := x + 1 reward=1 }
    else     { x := x − 1 }

  □ if (0.3) { x := x + 1 reward=1 }
    else     { x := x − 1 }
```

Figure: Gambler's Ruin as a Succinct MDP

Stochastic Shortest Path

- Fix an initial valuation $\mathbf{v}$ for the program variables.
- Let σ be a policy that at any point of time, given the history of the program, chooses one of the Q_i's to be executed.
- We define $R^{\infty}(\mathbf{v}, \sigma)$ as the expected sum of rewards collected by the program before termination, if the program starts with the valuation $\mathbf{v}$ and follows the policy σ.

Stochastic Shortest Path

- Fix an initial valuation $\mathbf{v}$ for the program variables.
- Let σ be a policy that at any point of time, given the history of the program, chooses one of the Q_i's to be executed.
- We define $R^{\infty}(\mathbf{v}, \sigma)$ as the expected sum of rewards collected by the program before termination, if the program starts with the valuation $\mathbf{v}$ and follows the policy σ.
- We define $\text{infval}(\mathbf{v}) = \inf_{\sigma} R^{\infty}(\mathbf{v}, \sigma)$, and $\text{supval}(\mathbf{v}) = \sup_{\sigma} R^{\infty}(\mathbf{v}, \sigma)$, where the inf and sup are taken over all policies that guarantee finite expected termination time.
- We are looking for methods to obtain upper and lower bounds for both infval and supval.

LUPFs and LLPFs

- We focus on supval, the approach for infval is similar.

LUPFs and LLPFs

- We focus on supval, the approach for infval is similar.
- Linear Upper Potential Function (LUPF):
  - Let X be the set of program variables, a function $h : \mathbb{R}^X \to \mathbb{R}$ is an LUPF if it satisfies the following conditions:

LUPFs and LLPFs

- We focus on supval, the approach for infval is similar.
- Linear Upper Potential Function (LUPF):
  - Let X be the set of program variables, a function $h : \mathbb{R}^X \to \mathbb{R}$ is an LUPF if it satisfies the following conditions:
    1. h is linear,

LUPFs and LLPFs

- We focus on supval, the approach for infval is similar.
- Linear Upper Potential Function (LUPF):
  - Let X be the set of program variables, a function $h : \mathbb{R}^X \to \mathbb{R}$ is an LUPF if it satisfies the following conditions:
    1. h is linear,
    2. the value of h at terminating valuations is bounded between two fixed constants K and K',

LUPFs and LLPFs

- We focus on supval, the approach for infval is similar.
- Linear Upper Potential Function (LUPF):
  - Let X be the set of program variables, a function $h : \mathbb{R}^X \to \mathbb{R}$ is an LUPF if it satisfies the following conditions:
    1. h is linear,
    2. the value of h at terminating valuations is bounded between two fixed constants K and K',
    3. For every Q_i and every valuation $\mathbf{v}$ that satisfies the loop guard:

$$h(\mathbf{v}) \geq \mathbb{E}_{\mathbf{u}}(h(Q_i(\mathbf{v}, \mathbf{u}))) + \mathbb{E}_{\mathbf{u}}(R(\mathbf{u}, Q_i))$$

LUPFs and LLPFs

- We focus on supval, the approach for infval is similar.
- Linear Upper Potential Function (LUPF):
  - Let X be the set of program variables, a function $h : \mathbb{R}^X \to \mathbb{R}$ is an LUPF if it satisfies the following conditions:
    1. h is linear,
    2. the value of h at terminating valuations is bounded between two fixed constants K and K',
    3. For every Q_i and every valuation $\mathbf{v}$ that satisfies the loop guard:
    $$h(\mathbf{v}) \geq \mathbb{E}_{\mathbf{u}}(h(Q_i(\mathbf{v}, \mathbf{u}))) + \mathbb{E}_{\mathbf{u}}(R(\mathbf{u}, Q_i))$$
    4. There is a fixed constant M, s.t. at each step of the program, the value of h changes by at most M.

LUPFs and LLPFs

- We focus on supval, the approach for infval is similar.
- Linear Upper Potential Function (LUPF):
  - Let X be the set of program variables, a function $h : \mathbb{R}^X \to \mathbb{R}$ is an LUPF if it satisfies the following conditions:
    1. h is linear,
    2. the value of h at terminating valuations is bounded between two fixed constants K and K',
    3. For every Q_i and every valuation $\mathbf{v}$ that satisfies the loop guard:
    $$h(\mathbf{v}) \geq \mathbb{E}_{\mathbf{u}}(h(Q_i(\mathbf{v}, \mathbf{u}))) + \mathbb{E}_{\mathbf{u}}(R(\mathbf{u}, Q_i))$$
    4. There is a fixed constant M, s.t. at each step of the program, the value of h changes by at most M.
- Linear Lower Potential Function (LLPF):
  - An LLPF h is a function that satisfies the above conditions, except that condition 3 is changed to:
    - For every $\mathbf{v}$ that satisfies the loop guard, there exists a Q_i such that $h(\mathbf{v}) \leq \mathbb{E}_{\mathbf{u}}(h(Q_i(\mathbf{v}, \mathbf{u}))) + \mathbb{E}_{\mathbf{u}}(R(\mathbf{u}, Q_i))$.

Theorem

If h is an LUPF, then $supval(\mathbf{v}) \leq h(\mathbf{v}) - K$ for all valuations $\mathbf{v} \in \mathbb{R}^X$ that satisfy the loop guard.

Theorem

If h is an LLPF, then $supval(\mathbf{v}) \geq h(\mathbf{v}) - K'$ for all valuations $\mathbf{v} \in \mathbb{R}^X$ that satisfy the loop guard.

Theorem

If h is an LUPF, then $supval(\mathbf{v}) \leq h(\mathbf{v}) - K$ for all valuations $\mathbf{v} \in \mathbb{R}^X$ that satisfy the loop guard.

Theorem

If h is an LLPF, then $supval(\mathbf{v}) \geq h(\mathbf{v}) - K'$ for all valuations $\mathbf{v} \in \mathbb{R}^X$ that satisfy the loop guard.

Sketch of Proof. Construct a stochastic process based on h. Show that it forms a supermartingale, and then apply the optional stopping theorem (OST).

Synthesizing LUPFs

while $x \geq 1$ **do**
- $\square$ **if** (0.4) { $x := x + 1$ **reward**=1 }
  else { $x := x - 1$ }
- $\square$ **if** (0.3) { $x := x + 1$ **reward**=1 }
  else { $x := x - 1$ }

Synthesizing LUPFs

```
while x ≥ 1 do
  □ if (0.4) { x := x + 1 reward=1 }
    else     { x := x − 1 }

  □ if (0.3) { x := x + 1 reward=1 }
    else     { x := x − 1 }
```

- Let $h : \mathbb{R} \to \mathbb{R}$ be an LUPF for this example, we have:

$$(1) \quad \exists \lambda_1, \lambda_2 \in \mathbb{R} \quad \forall x \in \mathbb{R} \quad h(x) = \lambda_1 x + \lambda_2$$

$$(2) \quad \exists K, K' \in \mathbb{R} \quad \forall x \in [1,2) \quad K \leq h(x) \leq K'$$

$$(3) \quad \forall x \in [1,\infty) \; h(x) \geq 0.4 \cdot (1 + h(x+1)) + 0.6 \cdot h(x-1)$$

$$(3) \quad \forall x \in [1,\infty) \; h(x) \geq 0.3 \cdot (1 + h(x+1)) + 0.7 \cdot h(x-1)$$

$$(4) \quad \exists M \in [0,\infty) \, \forall x \in [1,\infty) \quad |h(x) - h(x-1)| \leq M$$

$$(4) \quad \textbf{and} \quad |h(x) - h(x+1)| \leq M$$

- By applying Farkas lemma and solving the resulting LP with the goal of minimizing λ_1, we get:
  $\lambda_1 = M = 2, \lambda_2 = K = 0, K' = 4$.

- By applying Farkas lemma and solving the resulting LP with the goal of minimizing λ_1, we get:
  $\lambda_1 = M = 2, \lambda_2 = K = 0, K' = 4$.
- Therefore, we have $\text{supval}(\mathbf{x_0}) \leq 2\mathbf{x_0}$ for all initial valuations $\mathbf{x_0}$ that satisfy the loop guard.
- Hence, in this case the problem was solved by a reduction to linear programming.

Synthesizing LLPFs

while $x \geq 1$ **do**
  $\square$ **if** (0.4) { $x := x + 1$ **reward**=1 }
    else { $x := x - 1$ }

  $\square$ **if** (0.3) { $x := x + 1$ **reward**=1 }
    else { $x := x - 1$ }

Synthesizing LLPFs

while $x \geq 1$ **do**
$\square$ **if** (0.4) { $x := x + 1$ **reward**=1 }
else { $x := x - 1$ }

$\square$ **if** (0.3) { $x := x + 1$ **reward**=1 }
else { $x := x - 1$ }

- This case is a bit more complicated. If h is an LLPF, we must have the exact same conditions as before, except that condition 3 changes to:

$(3')$ $\forall x \in [1, \infty)$

$$h(x) \leq 0.4 \cdot (1 + h(x+1)) + 0.6 \cdot h(x-1)$$

or

$$h(x) \leq 0.3 \cdot (1 + h(x+1)) + 0.7 \cdot h(x-1)$$

Synthesizing LLPFs

while $x \geq 1$ **do**
- $\square$ **if** (0.4) { $x := x + 1$ **reward**=1 }
  else { $x := x - 1$ }
- $\square$ **if** (0.3) { $x := x + 1$ **reward**=1 }
  else { $x := x - 1$ }

- This case is a bit more complicated. If h is an LLPF, we must have the exact same conditions as before, except that condition 3 changes to:

$$(3') \quad \forall x \in [1, \infty)$$
$$h(x) \leq 0.4 \cdot (1 + h(x+1)) + 0.6 \cdot h(x-1)$$
or
$$h(x) \leq 0.3 \cdot (1 + h(x+1)) + 0.7 \cdot h(x-1)$$

which is equivalent to $\lambda_1 \leq 2$, and hence we get $\text{supval}(x_0) \geq 2x_0$. So, our previous bound is tight.

Theorem (Motzkin)

Let $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{B} \in \mathbb{R}^{k \times n}$ *and* $\mathbf{b} \in \mathbb{R}^m, \mathbf{c} \in \mathbb{R}^k$. *Assume that* $\{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{A}\mathbf{x} \leq \mathbf{b}\} \neq \emptyset$. *Then*

$$\{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{A}\mathbf{x} \leq \mathbf{b}\} \cap \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{B}\mathbf{x} < \mathbf{c}\} = \emptyset$$

iff there exist $\mathbf{y} \in \mathbb{R}^m$ *and* $\mathbf{z} \in \mathbb{R}^k$ *such that* $\mathbf{y}, \mathbf{z} \geq \mathbf{0}$, $\mathbf{1}^{\mathrm{T}} \cdot \mathbf{z} > 0$, $\mathbf{A}^{\mathrm{T}}\mathbf{y} + \mathbf{B}^{\mathrm{T}}\mathbf{z} = \mathbf{0}$ *and* $\mathbf{b}^{\mathrm{T}}\mathbf{y} + \mathbf{c}^{\mathrm{T}}\mathbf{z} \leq 0$.

Theorem (Motzkin)

Let $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{B} \in \mathbb{R}^{k \times n}$ *and* $\mathbf{b} \in \mathbb{R}^m, \mathbf{c} \in \mathbb{R}^k$. *Assume that* $\{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{A}\mathbf{x} \leq \mathbf{b}\} \neq \emptyset$. *Then*

$$\{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{A}\mathbf{x} \leq \mathbf{b}\} \cap \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{B}\mathbf{x} < \mathbf{c}\} = \emptyset$$

iff there exist $\mathbf{y} \in \mathbb{R}^m$ *and* $\mathbf{z} \in \mathbb{R}^k$ *such that* $\mathbf{y}, \mathbf{z} \geq \mathbf{0}$, $\mathbf{1}^{\mathrm{T}} \cdot \mathbf{z} > 0$, $\mathbf{A}^{\mathrm{T}}\mathbf{y} + \mathbf{B}^{\mathrm{T}}\mathbf{z} = \mathbf{0}$ *and* $\mathbf{b}^{\mathrm{T}}\mathbf{y} + \mathbf{c}^{\mathrm{T}}\mathbf{z} \leq 0$.

- By applying Motzkin's theorem, we can reduce the conditions to a *Quadratic Programming* problem.

Complexity

- Using Farkas' Lemma, we reduced the problem of finding the best LUPF to linear programming.
- Using Motzkin's Theorem, we reduced the problem of finding the best LLPF to quadratic programming.
- Both reductions are polynomial in the size of the succinct MDP (i.e., the length of the code).
- Note that the MDPs might have infinitely many states (e.g., if the variables can hold integers) or even uncountably many states (e.g., if the variables hold real values).

Experimental Results

MDP	Parameters	Upper bound	Lower bound	Time
Gambler's Ruin	$p_1 = 0.4$ $p_2 = 0.3$	$2x$	$2x$	153 ms
2D Robot Planning	$p = 0.4$	$5x - 5y$	$5x - 5y$	251 ms
Multi-robot Planning	$p_1 = 0.4$ $p_2 = 0.4$	$2.5x - 2.5y + 5$	$2.5x - 2.5y$	758 ms
Mini-roulette	—	$11x$	$11x$	320 ms
American Roulette	—	$24x$	$24x$	425 ms