Meet Your Expectations With Guarantees: Beyond Worst-Case Synthesis in Quantitative Games

V. Bruyère (UMONS)   E. Filiot (ULB)
M. Randour (UMONS-ULB)   J.-F. Raskin (ULB)

Brussels - 19.11.2013

MF&V seminar
The talk in one slide

Games
→ antagonistic adversary
→ guarantees on worst-case

MDPs
→ stochastic adversary
→ optimize expected value
The talk in one slide

Games
→ antagonistic adversary
→ guarantees on worst-case

MDPs
→ stochastic adversary
→ optimize expected value

BWC synthesis
→ ensure both
The talk in one slide

Games
→ antagonistic adversary
→ guarantees on worst-case

MDPs
→ stochastic adversary
→ optimize expected value

BWC synthesis
→ ensure both

Mean-Payoff

Studied value functions

Shortest Path
1. Context

2. BWC Synthesis

3. Mean-Payoff

4. Shortest Path

5. Conclusion
1 Context

2 BWC Synthesis

3 Mean-Payoff

4 Shortest Path

5 Conclusion
Quantitative games on graphs

- Graph $\mathcal{G} = (S, E, w)$ with $w : E \to \mathbb{Z}$
- Two-player game $G = (\mathcal{G}, S_1, S_2)$
  - $P_1$ states = ○
  - $P_2$ states = □
- Plays have values
  - $f : \text{Plays}(\mathcal{G}) \to \mathbb{R} \cup \{-\infty, \infty\}$
- Players follow strategies
  - $\lambda_i : \text{Prefs}_i(G) \to \mathcal{D}(S)$
  - Finite memory $\Rightarrow$ stochastic Moore machine $\mathcal{M}(\lambda_i) = (\text{Mem}, m_0, \alpha_u, \alpha_n)$
Quantitative games on graphs

- Graph $\mathcal{G} = (S, E, w)$ with $w : E \rightarrow \mathbb{Z}$
- Two-player game $G = (\mathcal{G}, S_1, S_2)$
  - $\mathcal{P}_1$ states = ○
  - $\mathcal{P}_2$ states = □
- Plays have values
  - $f : \text{Plays}(\mathcal{G}) \rightarrow \mathbb{R} \cup \{-\infty, \infty\}$
- Players follow strategies
  - $\lambda_i : \text{Prefs}_i(G) \rightarrow \mathcal{D}(S)$
  - Finite memory $\Rightarrow$ stochastic Moore machine $\mathcal{M}(\lambda_i) = (\text{Mem}, m_0, \alpha_u, \alpha_n)$
Quantitative games on graphs

- Graph $\mathcal{G} = (S, E, w)$ with $w : E \rightarrow \mathbb{Z}$
- Two-player game $G = (\mathcal{G}, S_1, S_2)$
  - $\mathcal{P}_1$ states = $igcirc$
  - $\mathcal{P}_2$ states = $lacksquare$
- Plays have values
  - $f : \text{Plays}(\mathcal{G}) \rightarrow \mathbb{R} \cup \{-\infty, \infty\}$
- Players follow strategies
  - $\lambda_i : \text{Pref}_i(G) \rightarrow \mathcal{D}(S)$
  - Finite memory $\Rightarrow$ stochastic Moore machine $\mathcal{M}(\lambda_i) = (\text{Mem}, m_0, \alpha_u, \alpha_n)$
Quantitative games on graphs

- **Graph** $\mathcal{G} = (S, E, w)$ with $w : E \rightarrow \mathbb{Z}$
- **Two-player game** $G = (\mathcal{G}, S_1, S_2)$
  - $\mathcal{P}_1$ states = $igcirc$
  - $\mathcal{P}_2$ states = $\square$
- Plays have values
  - $f : \text{Plays}(\mathcal{G}) \rightarrow \mathbb{R} \cup \{-\infty, \infty\}$
- Players follow *strategies*
  - $\lambda_i : \text{Prefs}_i(G) \rightarrow \mathcal{D}(S)$
  - Finite memory $\Rightarrow$ stochastic Moore machine $\mathcal{M}(\lambda_i) = (\text{Mem}, m_0, \alpha_u, \alpha_n)$
Quantitative games on graphs

- Graph $\mathcal{G} = (S, E, w)$ with $w : E \rightarrow \mathbb{Z}$
- Two-player game $G = (\mathcal{G}, S_1, S_2)$
  - $\mathcal{P}_1$ states = $\bigcirc$
  - $\mathcal{P}_2$ states = $\Box$
- Plays have values
  - $f : \text{Plays}(\mathcal{G}) \rightarrow \mathbb{R} \cup \{-\infty, \infty\}$
- Players follow strategies
  - $\lambda_i : \text{Prefs}_i(G) \rightarrow \mathcal{D}(S)$
  - Finite memory $\Rightarrow$ stochastic Moore machine
    $\mathcal{M}(\lambda_i) = (\text{Mem}, m_0, \alpha_u, \alpha_n)$
Quantitative games on graphs

- Graph \( \mathcal{G} = (S, E, w) \) with \( w : E \rightarrow \mathbb{Z} \)
- Two-player game \( G = (\mathcal{G}, S_1, S_2) \)
  - \( \mathcal{P}_1 \) states = ◦
  - \( \mathcal{P}_2 \) states = □
- Plays have values
  - \( f : \text{Plays}(\mathcal{G}) \rightarrow \mathbb{R} \cup \{-\infty, \infty\} \)
- Players follow strategies
  - \( \lambda_i : \text{Prefs}_i(G) \rightarrow \mathcal{D}(S) \)
  - Finite memory ⇒ stochastic Moore machine \( \mathcal{M}(\lambda_i) = (\text{Mem}, m_0, \alpha_u, \alpha_n) \)
Quantitative games on graphs

- Graph \( G = (S, E, w) \) with \( w : E \rightarrow \mathbb{Z} \)
- Two-player game \( G = (G, S_1, S_2) \)
  - \( P_1 \) states = \( \bigcirc \)
  - \( P_2 \) states = \( \Box \)

- Plays have values
  - \( f : \text{Plays}(G) \rightarrow \mathbb{R} \cup \{-\infty, \infty\} \)

- Players follow strategies
  - \( \lambda_i : \text{Prefs}_i(G) \rightarrow \mathcal{D}(S) \)
  - Finite memory \( \Rightarrow \) stochastic Moore machine \( \mathcal{M}(\lambda_i) = (\text{Mem}, m_0, \alpha_u, \alpha_n) \)

Then, \( (2, 5, 2)^\omega \)
Markov decision processes

- MDP \( P = (G, S_1, S_\Delta, \Delta) \) with \( \Delta : S_\Delta \to \mathcal{D}(S) \)
  - \( P_1 \) states =  
  - stochastic states = 

- MDP = game + strategy of \( P_2 \)
  - \( P = G[\lambda_2] \)
Markov decision processes

- **MDP** $P = (G, S_1, S_\Delta, \Delta)$ with $\Delta: S_\Delta \to \mathcal{D}(S)$
  - $P_1$ states $\bigcirc$
  - stochastic states $\Box$

- MDP = game + strategy of $P_2$
  - $P = G[\lambda_2]$

- **Important**: we allow $E \setminus E_\Delta \neq \emptyset$, $E_\Delta = \{(s_1, s_2) \in E \mid s_1 \in S_\Delta \Rightarrow \Delta(s_1)(s_2) > 0\}$
Markov chains

- \( \text{MC } M = (G, \delta) \text{ with } \delta: S \rightarrow \mathcal{D}(S) \)
- \( \text{MC} = \text{MDP} + \text{strategy of } \mathcal{P}_1 \)
  \( = \text{game } + \text{both strategies} \)
  \( \triangleright M = P[\lambda_1] = G[\lambda_1, \lambda_2] \)
Markov chains

- **MC** $M = (G, \delta)$ with $\delta : S \rightarrow D(S)$
- **MC** = MDP + strategy of $P_1$
  $= \text{game} + \text{both strategies}$
  $\triangleright M = P[\lambda_1] = G[\lambda_1, \lambda_2]$
- Event $A \subseteq \text{Plays}(G)$
  $\triangleright$ probability $Pr_{s_{\text{init}}}(A)$
- Measurable $f : \text{Plays}(G) \rightarrow \mathbb{R} \cup \{-\infty, \infty\}$
  $\triangleright$ expected value $E_{s_{\text{init}}}^M(f)$
Classical interpretations

- **System** trying to ensure a specification \(= \mathcal{P}_1\)
  - whatever the actions of its **environment**
Classical interpretations

- **System** trying to ensure a specification $= P_1$
  - whatever the actions of its environment

- The environment can be seen as
  - antagonistic
    - two-player game, worst-case threshold problem for $\mu \in \mathbb{Q}$
    - $\exists \lambda_1 \in \Lambda_1, \forall \lambda_2 \in \Lambda_2, \forall \pi \in \text{Outs}_G(s_{\text{init}}, \lambda_1, \lambda_2), f(\pi) \geq \mu$
Classical interpretations

- **System** trying to ensure a specification $= P_1$
  - whatever the actions of its **environment**

- The environment can be seen as
  - *antagonistic*
    - two-player game, *worst-case* threshold problem for $\mu \in \mathbb{Q}$
    - $\exists \lambda_1 \in \Lambda_1, \forall \lambda_2 \in \Lambda_2, \forall \pi \in \text{Outs}_G(s_{\text{init}}, \lambda_1, \lambda_2), f(\pi) \geq \mu$
  - *fully stochastic*
    - MDP, *expected value* threshold problem for $\nu \in \mathbb{Q}$
    - $\exists \lambda_1 \in \Lambda_1, \mathbb{E}_{s_{\text{init}}}^{P[\lambda_1]}(f) \geq \nu$
1 Context

2 BWC Synthesis

3 Mean-Payoff

4 Shortest Path

5 Conclusion
What if you want both?

In practice, we want both

1. nice expected performance in the everyday situation,
2. strict (but relaxed) performance guarantees even in the event of very bad circumstances.
Example: going to work

- **Weights** = minutes
- **Goal**: minimize our expected time to reach “work”
- **But**, important meeting in one hour! Requires strict guarantees on the worst-case reaching time.
Example: going to work

- Optimal expectation strategy: take the car.
  - $E = 33$, $WC = 71 > 60$.

- Optimal worst-case strategy: bicycle.
  - $E = WC = 45 < 60$. 
Example: going to work

Optimal expectation strategy: take the car.
- $E = 33$, $WC = 71 > 60$.

Optimal worst-case strategy: bicycle.
- $E = WC = 45 < 60$.

Sample BWC strategy: try train up to 3 delays then switch to bicycle.
- $E \approx 37.56$, $WC = 59 < 60$. 
Beyond worst-case synthesis

Formal definition

Given a game $G = (G, S_1, S_2)$, with $G = (S, E, w)$ its underlying graph, an initial state $s_{\text{init}} \in S$, a finite-memory stochastic model $\lambda_{\text{stoch}}^2 \in \Lambda_2$ of the adversary, represented by a stochastic Moore machine, a measurable value function $f : \text{Plays}(G) \rightarrow \mathbb{R} \cup \{-\infty, \infty\}$, and two rational thresholds $\mu, \nu \in \mathbb{Q}$, the beyond worst-case (BWC) problem asks to decide if $P_1$ has a finite-memory strategy $\lambda_1 \in \Lambda_1$ such that

$$\forall \lambda_2 \in \Lambda_2, \forall \pi \in \text{Outs}_G(s_{\text{init}}, \lambda_1, \lambda_2), f(\pi) > \mu$$

(1)

$$\mathbb{E}_{s_{\text{init}}}^{G}[\lambda_1, \lambda_{\text{stoch}}^2](f) > \nu$$

(2)

and the BWC synthesis problem asks to synthesize such a strategy if one exists.
Beyond worst-case synthesis

Formal definition

Given a game $G = (G, S_1, S_2)$, with $G = (S, E, w)$ its underlying graph, an initial state $s_{\text{init}} \in S$, a finite-memory stochastic model $\lambda_{\text{stoch}}^2 \in \Lambda_2^F$ of the adversary, represented by a stochastic Moore machine, a measurable value function $f : \text{Plays}(G) \to \mathbb{R} \cup \{-\infty, \infty\}$, and two rational thresholds $\mu, \nu \in \mathbb{Q}$, the beyond worst-case (BWC) problem asks to decide if $\mathcal{P}_1$ has a finite-memory strategy $\lambda_1 \in \Lambda_1^F$ such that

$$\begin{align*}
\forall \lambda_2 \in \Lambda_2, \forall \pi \in \text{Outs}_G(s_{\text{init}}, \lambda_1, \lambda_2), f(\pi) &> \mu \\
E_{s_{\text{init}}}^{G[\lambda_1, \lambda_{\text{stoch}}^2]}(f) &> \nu
\end{align*}$$

and the BWC synthesis problem asks to synthesize such a strategy if one exists.

Notice the highlighted parts!
1. Context

2. BWC Synthesis

3. Mean-Payoff

4. Shortest Path

5. Conclusion
Mean-payoff value function

\[ MP(\pi) = \lim_{n \to \infty} \inf \left[ \frac{1}{n} \cdot \sum_{i=0}^{i=n-1} w((s_i, s_{i+1})) \right] \]

- Sample play \( \pi = 2, -1, -4, 5, (2, 2, 5)^\omega \)
  - \( MP(\pi) = 3 \sim \text{prefix-independent} \)
Mean-payoff value function

\[ MP(\pi) = \lim_{n \to \infty} \inf \left[ \frac{1}{n} \cdot \sum_{i=0}^{i=n-1} w\left(s_i, s_{i+1}\right) \right] \]

Sample play \( \pi = 2, -1, -4, 5, (2, 2, 5)^\omega \)

\( MP(\pi) = 3 \sim \text{prefix-independent} \)

Games: worst-case threshold problem
[LL69, EM79, ZP96, Jur98, GS09]

Memoryless optimal strategies exist for both players and the problem is in \( \text{NP} \cap \text{coNP} \).

MDPs: expected value threshold problem [Put94, FV97]

Memoryless optimal strategies exist and the problem is in \( \text{P} \).
The BWC problem for the mean-payoff is in $\text{NP} \cap \text{coNP}$ and at least as hard as deciding the winner in mean-payoff games.

Additional modeling power for free!
BWC MP problem: overview

Theorem (algorithm & complexity)

The BWC problem for the mean-payoff is in $\text{NP} \cap \text{coNP}$ and at least as hard as deciding the winner in mean-payoff games.

Additional modeling power for free!

Theorem (memory bounds)

Memory of pseudo-polynomial size may be necessary and is always sufficient to satisfy the BWC problem for the mean-payoff: polynomial in the size of the game and the stochastic model, and polynomial in the weight and threshold values.
Algorithm: overview

Algorithm 1 BWC_MP\( (G^i, \lambda_{1}^i, \mu^i, v^i, s_{\text{init}}^i) \)

**Require:** \( G^i = (G^i, S^i_1, S^i_2) \) a game, \( G^i = (S^i, E^i, \omega^i) \) its underlying graph, \( \lambda_{1}^i \in \Lambda_{1}^{F}(G^i) \) a finite-memory stochastic model of the adversary, \( \mathcal{M}(\lambda_{2}^i) = (\text{Mem}, m_{0}, \alpha_{u}, \alpha_{n}) \) its Moore machine, \( \mu^i = \theta_{b}, v^i \in \mathbb{Q}, \mu^i < v^i \), resp. the worst-case and the expected value thresholds, and \( s_{\text{init}}^i \in S^i \) the initial state

**Ensure:** The answer is YES if and only if \( \mathcal{P}_{1} \) has a finite-memory strategy \( \lambda_1 \in \Lambda_{1}^{F}(G^i) \) satisfying the BWC problem from \( s_{\text{init}}^i \) for the thresholds pair \( (\mu^i, v^i) \) and the mean-payoff value function

\[
\{\text{Preprocessing}\}
\]

1: if \( \mu^i \neq 0 \) then
2: Modify the weight function of \( G^i \) s.t. \( \forall e \in E^i, w^i_{\text{new}}(e) := b \cdot w^i(e) - a \), and consider the new thresholds pair \( (0, v := b \cdot v^i - a) \)
3: Compute \( S_{\text{WC}} := \{ s \in S^i | \exists \lambda_1 \in \Lambda_{1}(G^i), \forall \lambda_2 \in \Lambda_{2}(G^i), \forall \pi \in \text{Outs}_{G^i}(s, \lambda_1, \lambda_2), \text{MP}(\pi) > 0 \} \)
4: if \( s_{\text{init}}^i \not\in S_{\text{WC}} \) then
5: return NO
6: else
7: Let \( G^w := G^i \mid S_{\text{WC}} \) be the subgame induced by worst-case winning states
8: Build \( G := G^w \otimes \mathcal{M}(\lambda_{2}^i) = (\tilde{G}, S_{1}, S_{2}), \tilde{G} = (S, E, \omega), S \subseteq (S_{\text{WC}} \times \text{Mem}) \), the game obtained by product with the Moore machine, and \( s_{\text{init}} := (s_{\text{init}}^i, m_{0}) \) the corresponding initial state
9: Let \( \lambda_{2}^{\text{stoch}} \in \Lambda_{2}^{M}(G) \) be the memoryless transcription of \( \lambda_{2}^i \) on \( G \)
10: Let \( P := G[\lambda_{2}^{\text{stoch}}] = (\tilde{G}, S_{1}, S_{\Delta} = S_{2}, \Delta = \lambda_{2}^{\text{stoch}}) \) be the MDP obtained from \( G \) and \( \lambda_{2}^{\text{stoch}} \)

\[
\{\text{Main algorithm}\}
\]

11: Compute \( U_{w} \), the set of maximal winning end-components of \( P \)
12: Build \( P^e = (G', S_{1}, S_{\Delta}^e, \Delta), \) where \( G' = (S, E, w') \) and \( w' \) is defined as follows:

\[
\forall e = (s_{1}, s_{2}) \in E, w'(e) := \begin{cases} w(e) & \text{if } \exists U \in U_{w} \text{ s.t. } \{s_{1}, s_{2}\} \subseteq U \\ 0 & \text{otherwise} \end{cases}
\]

13: Compute the maximal expected value \( v^* \) from \( s_{\text{init}} \) in \( P^e \)
14: if \( v^* > v \) then
15: return YES
16: else
17: return NO
Algorithm: overview

Algorithm 1 \text{BWC\_MP}(G^i, \lambda^i_1, \mu^i, v^i, s^i_{\text{init}})

\textbf{Require:} \(G^i = (G^i, S_1^i, S_2^i)\) a game, \(G^i = (S^i, E^i, w^i)\) its underlying graph, \(\lambda^i_1 \in \Lambda^i_2(G^i)\) a finite-memory stochastic model of the adversary, \(M(\lambda^i_2) = (\text{Mem}, m_0, a_u, a_n)\) its Moore machine, \(\mu^i = \frac{\delta}{b}, v^i \in \mathbb{Q}, \mu^i < v^i\), resp. the worst-case and the expected value thresholds, and \(s^i_{\text{init}} \in S^i\) the initial state

\textbf{Ensure:} The answer is YES if and only if \(P_1\) has a finite-memory strategy \(\lambda_1 \in \Lambda^i_1(G^i)\) satisfying the BWC problem from \(s^i_{\text{init}}\), for the thresholds pair \((\mu^i, v^i)\) and the mean-payoff value function

\begin{enumerate}
\item \textbf{Preprocessing}
\begin{enumerate}
\item if \(\mu^i \neq 0\) then
\item Modify the weight function of \(G^i\) s.t. \(\forall e \in E^i, w^i_{\text{new}}(e) := b \cdot w^i(e) - a\), and consider the new thresholds pair \((0, v := b \cdot v^i - a)\)
\item Compute \(S_{\text{WC}} := \{s \in S^i | \exists \lambda_1 \in \Lambda^i_1(G^i), \forall \lambda_2 \in \Lambda^i_2(G^i), \forall \pi \in \text{Outs}_{G^i}(s, \lambda_1, \lambda_2), \text{MP}(\pi) > 0\}\)
\item if \(s^i_{\text{init}} \notin S_{\text{WC}}\) then
\item return NO
\end{enumerate}
\item else
\begin{enumerate}
\item Let \(G^w := G^i \mid S_{\text{WC}}\) be the subgame induced by worst-case winning states
\item Build \(G := G^w \otimes M(\lambda^i_1) = (G, S_1, S_2), G = (S, E, w), S \subseteq (S_{\text{WC}} \times \text{Mem})\), the game obtained by product with the Moore machine, and \(s^i_{\text{init}} := (s^i_{\text{init}}, m_0)\) the corresponding initial state
\item Let \(\lambda^i_2^\text{stoch} \in \Lambda^i_M(G)\) be the memoryless transcription of \(\lambda^i_2\) on \(G\)
\item Let \(P := G[\lambda^i_2^\text{stoch}] = (G, S_1, S_\Delta = S_2, \Delta = \lambda^i_2^\text{stoch})\) be the MDP obtained from \(G\) and \(\lambda^i_2^\text{stoch}\)
\end{enumerate}
\end{enumerate}

\textbf{Main algorithm}

\begin{enumerate}
\item Compute \(U_w\) the set of maximal winning end-components of \(P\)
\item Build \(P' = (G', S_1, S_\Delta, \Delta)\), where \(G' = (S, E, w')\) and \(w'\) is defined as follows:
\[
\forall e = (s_1, s_2) \in E, w'(e) := \begin{cases} w(e) & \text{if } \exists U \in U_w \text{ s.t. } \{s_1, s_2\} \subseteq U \\ 0 & \text{otherwise} \end{cases}
\]
\item Compute the maximal expected value \(v^*\) from \(s^i_{\text{init}}\) in \(P'\)
\begin{enumerate}
\item if \(v^* > v\) then
\item return YES
\item else
\item return NO
\end{enumerate}
\end{enumerate}
Algorithm: overview

Algorithm 1 BWC-MP($G^i, \lambda_2^i, \mu^i, v^i, s_{init}$)

Require: $G^i = (G^i, S_1^i, S_2^i)$ a game, $G = (S', E^i, w^i)$ its underlying graph, $\lambda_2^i \in \Lambda_2^i(G^i)$ a finite-memory stochastic model of the adversary, $\mathcal{M}(\lambda_2^i) = (\text{Mem}, m_0, \alpha_u, \alpha_v)$ its Moore machine, $\mu^i = \frac{\mu}{b}, v^i \in \mathbb{Q}, \mu^i < v^i$, resp. the worst-case and the expected value thresholds, and $s_{init}^i \in S'$ the initial state

Ensure: The answer is YES if and only if $\mathcal{P}_1$ has a finite-memory strategy $\lambda_1 \in \Lambda_1^i(G^i)$ satisfying the BWC problem from $s_{init}^i$, for the thresholds pair $(\mu^i, v^i)$ and the mean-payoff value function

{Preprocessing}
1: if $\mu^i \neq 0$ then
2: Modify the weight function of $G^i$ s.t. $\forall e \in E^i$, $w_{new}(e) := b \cdot w^i(e) - a$, and consider the new thresholds pair $(0, v := b \cdot v^i - a)$
3: Compute $S_{WC} := \{ s \in S' | \exists \lambda_1 \in \Lambda_1(G^i), \forall \lambda_2 \in \Lambda_2(G^i), \forall \pi \in \text{Outs}_{G^i}(s, \lambda_1, \lambda_2), \text{MP}(\pi) > 0 \}$
4: if $s_{init}^i \not\in S_{WC}$ then
5: return NO
6: else
7: Let $G^w := G^i | S_{WC}$ be the subgame induced by worst-case winning states
8: Build $G := G^w \otimes \mathcal{M}(\lambda_2^i) = (G, S_1, S_2), G = (S, E, w), S \subseteq (S_{WC} \times \text{Mem})$, the game obtained by product with the Moore machine, and $s_{init} := (s_{init}^i, m_0)$ the corresponding initial state
9: Let $\lambda_2^{stoch} \in \Lambda_2^M(G)$ be the memoryless transcription of $\lambda_2^i$ on $G$
10: Let $P := G[\lambda_2^{stoch}] = (G, S_1, S_{\Delta} = S_2, \Delta = \lambda_2^{stoch})$ be the MDP obtained from $G$ and $\lambda_2^{stoch}$

{Main algorithm}
11: Compute $U_w$ the set of maximal winning end-components of $P$
12: Build $P' = (G', S_1, S_{\Delta}, \Delta)$, where $G' = (S, E, w')$ and $w'$ is defined as follows:

$$\forall e = (s_1, s_2) \in E, w'(e) := \begin{cases} w(e) & \text{if } \exists U \in U_w \text{ s.t. } \{s_1, s_2\} \subseteq U \\ 0 & \text{otherwise} \end{cases}$$

13: Compute the maximal expected value $v^*$ from $s_{init}$ in $P'$
14: if $v^* > v$ then
15: return YES
16: else
17: return NO
Algorithm: overview

Algorithm 1 BWC\_MP\((G^i, \lambda^i_1, \mu^i, v^i, s^i_{\text{init}})\)

Require: \(G^i = (G^i, S^i_1, S^i_2)\) a game, \(G^i = (S^i, E^i, w^i)\) its underlying graph, \(\lambda^i_1 \in \Lambda^F_1(G^i)\) a finite-memory stochastic model of the adversary, \(\mathcal{M}(\lambda^i_1) = (\text{Mem}, m_0, \alpha_0, \alpha_n)\) its Moore machine, \(\mu^i = \frac{g}{b}, v^i \in \mathbb{Q}, \mu^i < v^i\), resp. the worst-case and the expected value thresholds, and \(s^i_{\text{init}} \in S^i\) the initial state.

Ensure: The answer is YES if and only if \(\mathcal{P}_1\) has a finite-memory strategy \(\lambda_1 \in \Lambda^F_1(G^i)\) satisfying the BWC problem from \(s^i_{\text{init}}\), for the thresholds pair \((\mu^i, v^i)\) and the mean-payoff value function.

\[
\{\text{Preprocessing}\}
\]

1: if \(\mu^i \neq 0\) then
2: Modify the weight function of \(G^i\) s.t. \(\forall e \in E^i, w^i_{\text{new}}(e) := b \cdot w^i(e) - a\), and consider the new thresholds pair \((0, v := b \cdot v^i - a)\)
3: Compute \(S_{\text{WC}} := \{s \in S^i \mid \exists \lambda_1 \in \Lambda_1(G^i), \forall \lambda_2 \in \Lambda_2(G^i), \forall \pi \in \text{Outs}_{G^i}(s, \lambda_1, \lambda_2), MP(\pi) > 0\}\)
4: if \(s^i_{\text{init}} \not\in S_{\text{WC}}\) then
5: return NO
6: else
7: Let \(G^w := G^i \mid S_{\text{WC}}\) be the subgame induced by worst-case winning states.
8: Build \(G := G^w \otimes M(\lambda^i_1) = (G, S_1, S_2), G = (S, E, w), S \subseteq (S_{\text{WC}} \times \text{Mem})\), the game obtained by product with the Moore machine, and \(s^i_{\text{init}} := (s^i_{\text{init}}, m_0)\) the corresponding initial state.
9: Let \(\lambda^2_{\text{stoch}} \in \Lambda^M_2(G)\) be the memoryless transcription of \(\lambda^i_2\) on \(G\).
10: Let \(P := G[\lambda^2_{\text{stoch}}] = (G, S_1, S_\Delta = S_2, \Delta = \lambda^2_{\text{stoch}})\) be the MDP obtained from \(G\) and \(\lambda^2_{\text{stoch}}\).

\[
\{\text{Main algorithm}\}
\]

11: Compute \(U_w\) the set of maximal winning end-components of \(P\).
12: Build \(P' = (G', S_1, S_\Delta, \Delta)\), where \(G' = (S, E, w')\) and \(w'\) is defined as follows:

\[
\forall e = (s_1, s_2) \in E, w'(e) := \begin{cases} 
    w(e) \text{ if } \exists U \in U_w \text{ s.t. } \{s_1, s_2\} \subseteq U \\
    0 \text{ otherwise} 
\end{cases}
\]

13: Compute the maximal expected value \(v^*\) from \(s^i_{\text{init}}\) in \(P'\).
14: if \(v^* > v\) then
15: return YES
16: else
17: return NO
Preprocessing: three steps

1. Modify weights and use thresholds \((\mu = 0, \nu)\)
   - simple trick to ease the following technicalities

\[ S_{WC} := \{ s \in S_i \mid \exists \lambda_1 \in \Lambda_1 (G_i), \forall \lambda_2 \in \Lambda_2 (G_i), \forall \pi \in \text{Outs} G_i (s, \lambda_1, \lambda_2), \text{MP}(\pi) > 0 \} \]

\[ G_w := G_i \downharpoonright S_{WC} \]

BWC satisfying strategies must avoid \( S \setminus S_{WC} \): an antagonistic adversary can force WC losing outcomes from there (due to prefix-independence)

\[ \text{Answer} \] No if \( s_{init} \not\in S_{WC} \)

In \( G_w \), \( P_1 \) has a memoryless WC winning strategy from all states
Preprocessing: three steps

1. Modify weights and use thresholds \((\mu = 0, \nu)\)
   - simple trick to ease the following technicalities

2. Remove all worst-case losing states

   \[
   S_{WC} := \left\{ s \in S^i \mid \exists \lambda_1 \in \Lambda_1(G^i), \forall \lambda_2 \in \Lambda_2(G^i), \forall \pi \in \text{Outs}_{G^i}(s, \lambda_1, \lambda_2), \text{MP}(\pi) > 0 \right\}
   \]

   \[
   G^w := G^i \upharpoonright S_{WC}
   \]

   - BWC satisfying strategies must avoid \(S \setminus S_{WC}\): an antagonistic adversary can force WC losing outcomes from there (due to prefix-independence)
   - Answer \(\text{NO}\) if \(s_{init} \not\in S_{WC}\)
Preprocessing: three steps

1. Modify weights and use thresholds ($\mu = 0, \nu$)
   - simple trick to ease the following technicalities

2. Remove all worst-case losing states

$$S_{WC} := \left\{ s \in S^i \mid \exists \lambda_1 \in \Lambda_1(G^i), \forall \lambda_2 \in \Lambda_2(G^i), \forall \pi \in \text{Outs}_{G^i}(s, \lambda_1, \lambda_2), \text{MP}(\pi) > 0 \right\}$$

$$G^w := G^i \downharpoonright S_{WC}$$

- BWC satisfying strategies must avoid $S \setminus S_{WC}$: an antagonistic adversary can force WC losing outcomes from there (due to prefix-independence)
- Answer No if $s_{init} \not\in S_{WC}$
- In $G^w$, $P_1$ has a memoryless WC winning strategy from all states
Preprocessing: three steps

3. Build $G := G^w \otimes M(\lambda_2^i)$, the game obtained by product with the Moore machine
   
   - Corresponding stochastic model $\lambda_2^{\text{stoch}} \in \Lambda_2^M(G)$ is memoryless
Preprocessing: three steps

3. Build $G := G^w \otimes M(\lambda^i_2)$, the game obtained by product with the Moore machine
   
   ▶ Corresponding stochastic model $\lambda_2^{\text{stoch}} \in \Lambda^M(G)$ is memoryless
   
   ▶ Obtain the MDP $P := G[\lambda_2^{\text{stoch}}]$, sharing the same graph
     ■ helps for elegant proofs
Main algorithm: end-components

▷ An **EC** of the MDP $P = G[\lambda_{2}^{\text{stoch}}]$ is a subgraph in which $P_1$ can ensure to stay despite stochastic states [dA97], i.e., a set $U \subseteq S$ s.t.

(i) $(U, E_\Delta \cap (U \times U))$ is strongly connected,

(ii) $\forall s \in U \cap S_\Delta$, $\text{Supp}(\Delta(s)) \subseteq U$, i.e., in stochastic states, all outgoing edges either stay in $U$ or belong to $E \setminus E_\Delta$.

▷ Beware arbitrary adversaries may use edges in $E \setminus E_\Delta$!
Main algorithm: end-components

ECs: $\mathcal{E} = \{ U_1 \}$
Main algorithm: end-components

ECs: $\mathcal{E} = \{U_1, U_2\}$
Main algorithm: end-components

ECs: $\mathcal{E} = \{U_1, U_2, U_3\}$
Main algorithm: end-components

ECs: $\mathcal{E} = \{U_1, U_2, U_3, \{s_5, s_6\}, \{s_6, s_7\}, \{s_1, s_3, s_4, s_5\}\}$
Main algorithm: end-components

ECs: $\mathcal{E} = \{U_1, U_2, U_3, \{s_5, s_6\}, \{s_6, s_7\}, \{s_1, s_3, s_4, s_5\}\}$

Lemma (Long-run appearance of ECs [CY95, dA97])

Let $\lambda_1 \in \Lambda_1(P)$ be an arbitrary strategy of $P_1$. Then, we have that

$$\Pr_{s_{\text{init}}}^{P[\lambda_1]}(\{\pi \in \text{Outs}_{P[\lambda_1]}(s_{\text{init}}) \mid \inf(\pi) \in \mathcal{E}\}) = 1.$$
How to satisfy the BWC problem?

- *Expected value requirement*: reach ECs with the highest achievable expectations and stay in them (optimal expected value in EC [FV97])
How to satisfy the BWC problem?

- **Expected value requirement**: reach ECs with the highest achievable expectations and stay in them (optimal expected value in EC [FV97])

- **Worst-case requirement**: some ECs may need to be eventually avoided because risky!
Classification of ECs

$U \in \mathcal{W}$, the winning ECs, if $P_1$ can win in $G_\Delta \mid U$, from all states:

$\exists \lambda_1 \in \Lambda_1(G_\Delta \mid U), \forall \lambda_2 \in \Lambda_2(G_\Delta \mid U), \forall s \in U, \forall \pi \in \text{Outs}_{(G_\Delta \mid U)}(s, \lambda_1, \lambda_2), \text{MP}(\pi) > 0$
Classification of ECs

\[
\begin{array}{c}
U_2 \\
\frac{1}{2} \\
1 \\
-1 \\
0 \\
1 \\
\end{array}
\]

\[
\begin{array}{c}
U_3 \\
\frac{1}{2} \\
9 \\
1 \\
0 \\
\frac{1}{2} \\
\end{array}
\]

\[
\begin{array}{c}
U_1 \\
1 \\
\end{array}
\]

\[U \in \mathcal{W}, \text{ the winning ECs, if } \mathcal{P}_1 \text{ can win in } G_\Delta \downarrow U, \text{ from all states:}\]

\[\exists \lambda_1 \in \Lambda_1(G_\Delta \downarrow U), \forall \lambda_2 \in \Lambda_2(G_\Delta \downarrow U), \forall s \in U, \forall \pi \in \text{Outs}(G_\Delta \downarrow U)(s, \lambda_1, \lambda_2), \text{ MP}(\pi) > 0\]

\[\mathcal{W} = \{U_1, U_3, \{s_5, s_6\}, \{s_6, s_7\}\}\]

\[U_2 \text{ losing: from state } s_1, \mathcal{P}_2 \text{ can force the outcome }\]

\[\pi = (s_1s_3s_4)^\omega \text{ of MP}(\pi) = -1/3 < 0\]
Winning ECs: usefulness

Lemma (Long-run appearance of winning ECs)

Let $\lambda_1^f \in \Lambda_1^F$ be a finite-memory strategy of $P_1$ that satisfies the BWC problem for thresholds $(0, \nu) \in \mathbb{Q}^2$. Then, we have that

$$\mathbb{P}_{s_{\text{init}}}^{P[\lambda_1^f]} \left( \left\{ \pi \in \text{Outs}_{P[\lambda_1^f]}(s_{\text{init}}) \mid \text{Inf}(\pi) \in \mathcal{W} \right\} \right) = 1.$$
Winning ECs: usefulness

**Lemma (Long-run appearance of winning ECs)**

Let $\lambda^f_1 \in \Lambda^F_1$ be a finite-memory strategy of $P_1$ that satisfies the BWC problem for thresholds $(0, \nu) \in \mathbb{Q}^2$. Then, we have that

$$\mathbb{P}^{P_{s_{init}}} \left( \left\{ \pi \in \text{Outs}_{P_{\lambda^f_1}}(s_{init}) \mid \inf(\pi) \in \mathcal{W} \right\} \right) = 1.$$

▷ A good finite-memory strategy for the BWC problem should maximize the expected value achievable through winning ECs.
Winning ECs: computation

- Deciding if an EC is winning or not is in \( \text{NP} \cap \text{coNP} \) (worst-case threshold problem)
- \( |E| \leq 2^{|S|} \sim \) exponential \# of ECs
Winning ECs: computation

- Deciding if an EC is winning or not is in NP $\cap$ coNP (worst-case threshold problem)
- $|\mathcal{E}| \leq 2^{|S|} \sim$ exponential $\#$ of ECs
- Considering the maximal ECs does not suffice! See $U_3 \subset U_2$
Winning ECs: computation

- Deciding if an EC is winning or not is in \( \text{NP} \cap \text{coNP} \) (worst-case threshold problem)
- \(|\mathcal{E}| \leq 2^{|S|} \sim \) exponential number of ECs
- Considering the maximal ECs does not suffice! See \( U_3 \subset U_2 \)

But,

- possible to define a recursive algorithm computing the maximal winning ECs, such that \( |U_w| \leq |S| \), in \( \text{NP} \cap \text{coNP} \).
- Uses polynomial number of calls to
  - max. EC decomp. of sub-MDPs (each in \( O(|S|^2) \) [CH12]),
  - worst-case threshold problem (NP \( \cap \) coNP).
- Critical complexity gain for the overall algorithm BWC\_MP!
Winning ECs: what can we expect?

We know we can only benefit from the expectation of winning ECs. But how can we compute it?
Winning ECs: what can we expect?

We know we can only benefit from the expectation of winning ECs. But how can we compute it?

**Theorem (BWC satisfaction from winning ECs)**

Let $U \in \mathcal{W}$ a winning EC, $s_{\text{init}} \in U$ an initial state inside the EC, and $\nu^* \in \mathbb{Q}$ the maximal expected value achievable by $\mathcal{P}_1$ in $P \upharpoonright U$. Then, for all $\varepsilon > 0$, there exists a finite-memory strategy of $\mathcal{P}_1$ that satisfies the BWC problem for the thresholds pair $(0, \nu^* - \varepsilon)$.

▷ We can be arbitrarily close to the optimal expectation of the EC while ensuring the worst-case!
Inside a WEC: combined strategy

Consider the WEC $U_3 \subseteq S$ and $E \setminus E_\Delta = \emptyset$
Inside a WEC: combined strategy

Consider the WEC $U_3 \subseteq S$ and $E \setminus E_\Delta = \emptyset$

Two particular memoryless strategies exist:

1. Optimal expected value strategy $\lambda^e_1 \in \Lambda^{PM}_1(P)$, yielding $\mathbb{E} = 2$
2. Optimal worst-case strategy $\lambda^{wc}_1 \in \Lambda^{PM}_1(G)$, ensuring $\text{MP} = 1 > 0$

Remark: $\nu^* = 2 > \mu^* = 1$
Inside a WEC: combined strategy

Consider the WEC $U_3 \subseteq S$ and $E \setminus E_\Delta = \emptyset$

We define $\lambda_{1}^{cmb} \in \Lambda_1^{PF}$ as follows, for some well-chosen $K, L \in \mathbb{N}$.

**(a)** Play $\lambda_1^e$ for $K$ steps and memorize $\text{Sum} \in \mathbb{Z}$, the sum of weights encountered during these $K$ steps.

**(b)** If $\text{Sum} > 0$, then go to (a).

Else, play $\lambda_1^{wc}$ during $L$ steps then go to (a).
Inside a WEC: combined strategy

Consider the WEC $U_3 \subseteq S$ and $E \setminus E_\Delta = \emptyset$

- **Phase (a):** try to increase the expectation and approach the optimal one
- **Phase (b):** compensate, if needed, losses that occurred in (a)
Combined strategy: parameters

**Key result:** $\exists K, L \in \mathbb{N}$ for any thresholds pair $(0, \nu^* - \varepsilon)$

- plays = sequences of periods starting with phase $(a)$
Combined strategy: parameters

Key result: \( \exists K, L \in \mathbb{N} \) for any thresholds pair \((0, \nu^* - \varepsilon)\)

- plays = sequences of periods starting with phase \((a)\)

- Worst-case requirement
  - \( \forall K, \exists L(K) \text{ s.t. } (a) + (b) \text{ has } \text{MP} \geq 1/(K + L) > 0 \)
  - Periods \((a)\) induce \( \text{MP} \geq 1/K \) (not followed by \((b)\))
  - Weights are integers and period length bounded \(\sim\) inequality remains strict for play
Combined strategy: parameters

**Key result:** \( \exists K, L \in \mathbb{N} \) for any thresholds pair \((0, \nu^* - \varepsilon)\)

- plays = sequences of periods starting with phase \((a)\)

- **Worst-case requirement**
  - \( \forall K, \exists L(K) \text{ s.t. } (a) + (b) \text{ has } MP \geq 1/(K + L) > 0 \)
  - Periods \((a)\) induce \( MP \geq 1/K \) (not followed by \((b)\))
  - Weights are integers and period length bounded \(\sim\) inequality remains strict for play

- **Expected value requirement**
  - When \( K \to \infty, \mathbb{E}(a) \to \nu^* \)
  - We need the *overall contribution* of \((b)\) to tend to zero when \( K \to \infty \)
    - \( \mathbb{P}(b) \) decreases faster than increase of \( L(K) \): exponential vs. polynomial
    - proved using results related to Chernoff bounds and Hoeffding's inequality on MCs [Tra09, GO02]: bound on the probability of being far from the optimal after \( K \) steps of \((a)\)
Witness-and-secure strategy

What if $E \setminus E_\Delta \neq \emptyset$?

- arbitrary adversaries can produce bad behaviors
- add the possibility to **react** using a worst-case winning strategy (existing everywhere thanks to the preprocessing)
  - guarantees worst-case
  - no impact on expected value (probability zero)
Back to the algorithm

So we know we should only use WECs and we know how to play $\varepsilon$-optimally when starting in a WEC. *What remains to settle?*
So we know we should only use WECs and we know how to play ϵ-optimally when starting in a WEC. *What remains to settle?*

▷ Determine **which** WECs to reach and **how**!
Back to the algorithm

So we know we should only use WECs and we know how to play $\varepsilon$-optimally when starting in a WEC. What remains to settle?

- Determine **which** WECs to reach and **how**!
- Key idea: define a **global strategy** that will go towards the highest valued WECs and avoid LECs
Global strategy via modified MDP

WEC $U_3 - E = 2$

WEC $U_2 - E = 3$

LEC $U_1 - E = 4$
Global strategy via modified MDP

Modify weights:

\[ \forall e = (s_1, s_2) \in E, \ w'(e) := \begin{cases} w(e) & \text{if } \exists U \in \mathcal{U}_W \text{ s.t. } \{s_1, s_2\} \subseteq U, \\ 0 & \text{otherwise.} \end{cases} \]
Global strategy via modified MDP

2. Compute memoryless optimal expectation strategy $\lambda_1^e$ on $P'$

- the probability to be in a good WEC (here, $U_2$) after $N$ steps tends to one when $N \to \infty$
Global strategy via modified MDP

3 \( \lambda_1^{gb} \in \Lambda_1^{PF}(G) \):
   
   (a) Play \( \lambda_1^e \in \Lambda_1^{PM}(G) \) for \( N \) steps.
   
   (b) Let \( s \in S \) be the reached state.
      
      (b.1) If \( s \in U \in \mathcal{U}_w \), play corresponding \( \lambda_1^{wns} \in \Lambda_1^{PF}(G) \) forever.
      
      (b.2) Else play \( \lambda_1^{wc} \in \Lambda_1^{PM}(G) \) forever.

▷ Parameter \( N \in \mathbb{N} \) can be chosen so that overall expectation is arbitrarily close to optimal in \( P' \), or equivalently, optimal for BWC strategies in \( P \)

▷ Algorithm BWC\_MP answers \textbf{Yes} iff \( \nu^* > \nu \)
Correctness and completeness

Algorithm BWC-MP is

- **correct**: if answer is **YES**, then $\lambda_{1}^{glb}$ satisfies the BWC problem for the given thresholds

- **complete**: if answer is **NO**, then the BWC problem cannot be satisfied by a finite-memory strategy
BWC MP problem: bounds

- **Complexity**
  - algorithm in $\text{NP} \cap \text{coNP}$ (P if MP games proved in P)
  - lower bound via reduction from MP games
BWC MP problem: bounds

- **Complexity**
  - Algorithm in $NP \cap coNP$ ($P$ if MP games proved in $P$)
  - Lower bound via reduction from MP games

- **Memory**
  - Pseudo-polynomial upper bound via global strategy
  - Matching lower bound via family $(G(X))_{X \in \mathbb{N}_0}$ requiring polynomial memory in $W = X + 5$ to satisfy the BWC problem for thresholds $(0, \nu \in ]1, 5/4[)$
  - Need to use $(s_1, s_3)$ infinitely often for $\mathbb{E}$ but need pseudo-poly. memory to counteract $-X$ for the WC requirement
1. Context

2. BWC Synthesis

3. Mean-Payoff

4. Shortest Path

5. Conclusion
Shortest path - truncated sum

- Assume strictly positive integer weights, \( w : E \rightarrow \mathbb{N}_0 \)
- Let \( T \subseteq S \) be a target set that \( P_1 \) wants to reach with a path of bounded value (cf. introductory example)
  - Inequalities are reversed, \( \nu < \mu \)
- \( TS_T(\pi = s_0s_1s_2\ldots) = \sum_{i=0}^{n-1} w((s_i, s_{i+1})) \), with \( n \) the first index such that \( s_n \in T \), and \( TS_T(\pi) = \infty \) if \( \forall n, s_n \notin T \)
Shortest path - truncated sum

- Assume strictly positive integer weights, \( w : E \to \mathbb{N}_0 \)
- Let \( T \subseteq S \) be a target set that \( P_1 \) wants to reach with a path of bounded value (cf. introductory example)
  - Inequalities are reversed, \( \nu < \mu \)
- \( \text{TS}_T(\pi = s_0s_1s_2\ldots) = \sum_{i=0}^{n-1} w((s_i, s_{i+1})) \), with \( n \) the first index such that \( s_n \in T \), and \( \text{TS}_T(\pi) = \infty \) if \( \forall n, s_n \notin T \)

Games: worst-case threshold problem

Memoryless optimal strategies as cycles are to be avoided, and the problem is in P, solvable using attractors and computation of the worst cost.

MDPs: expected value threshold problem [BT91, dA99]

Memoryless optimal strategies exist and the problem is in P.
BWC SP problem: overview

Theorem (algorithm)

The BWC problem for the shortest path can be solved in pseudo-polynomial time: polynomial in the size of the game graph, the Moore machine for the stochastic model of the adversary and the encoding of the expected value threshold, and polynomial in the value of the worst-case threshold.

Theorem (memory bounds)

Pseudo-polynomial memory may be necessary and is always sufficient to satisfy the BWC problem for the shortest path.

Theorem (complexity lower bound)

The BWC problem for the shortest path is $\text{NP-hard}$. 
Pseudo-polynomial algorithm: sketch

1. Start from $G = (G, S_1, S_2)$, $G = (S, E, w)$, $T = \{s_3\}$, $M(\lambda^{stoch}_{2})$, $\mu = 8$, and $\nu \in \mathbb{Q}$
Pseudo-polynomial algorithm: sketch

1. Start from $G = (\mathcal{G}, S_1, S_2), \mathcal{G} = (S, E, w), T = \{s_3\}, \mathcal{M}(\lambda_2^{\text{stoch}}), \mu = 8$, and $\nu \in \mathbb{Q}$

2. Build $G'$ by unfolding $\mathcal{G}$, tracking the current sum up to the worst-case threshold $\mu$, and integrating it in the states of $G'$. 
Pseudo-polynomial algorithm: sketch

Beyond Worst-Case Synthesis

Bruyère, Filiot, Randour, Raskin
Pseudo-polynomial algorithm: sketch

\begin{equation*}
\nu^* = \frac{9}{2}
\end{equation*}
Pseudo-polynomial algorithm: sketch

Beyond Worst-Case Synthesis Bruyère, Filiot, Randour, Raskin 36 / 44
Pseudo-polynomial algorithm: sketch
Pseudo-polynomial algorithm: sketch

Beyond Worst-Case Synthesis
Bruyère, Filiot, Randour, Raskin
Pseudo-polynomial algorithm: sketch
Pseudo-polynomial algorithm: sketch

\[ \nu^* = \frac{9}{2} \]

\[ s_1, 5 \]

\[ s_1, 2 \]

\[ s_1, 7 \]

\[ s_1, 4 \]

\[ s_1, 6 \]

\[ s_2, 7 \]

\[ s_1, T \]
Pseudo-polynomial algorithm: sketch

3. Compute $R$, the attractor of $T$ with cost $< \mu = 8$
4. Consider $G_\mu = G' | R$
Pseudo-polynomial algorithm: sketch

3. Compute $R$, the attractor of $T$ with cost $< \mu = 8$

4. Consider $G_\mu = G' \upharpoonright R$
Pseudo-polynomial algorithm: sketch

5. Consider $P = G_\mu \otimes \mathcal{M}(\lambda_{2}^{\text{stoch}})$

6. Compute memoryless optimal expectation strategy

7. If $\nu^* < \nu$, answer YES, otherwise answer NO

Here, $\nu^* = 9/2$
Memory bounds

- Upper bound provided by synthesized strategy
- Lower bound given by family of games \((G(\mu))_{\mu \in \{13+k \cdot 4 | k \in \mathbb{N}\}}\) requiring memory linear in \(\mu\)
  \(\leadsto\) play \((s_1, s_2)\) exactly \(\left\lfloor \frac{\mu}{4} \right\rfloor\) times and then switch to \((s_1, s_3)\) to minimize expected value while ensuring the worst-case
Complexity lower bound: NP-hardness

- Truly-polynomial algorithm very unlikely...

- Reduction from the $K^{th}$ largest subset problem
  - commonly thought to be outside NP as natural certificates are larger than polynomial [JK78, GJ79]
Complexity lower bound: NP-hardness

- Truly-polynomial algorithm very unlikely...

- Reduction from the $K^{th}$ largest subset problem

  - commonly thought to be outside NP as natural certificates are larger than polynomial [JK78, GJ79]

$K^{th}$ largest subset problem

Given a finite set $A$, a size function $h: A \rightarrow \mathbb{N}_0$ assigning strictly positive integer values to elements of $A$, and two naturals $K, L \in \mathbb{N}$, decide if there exist $K$ distinct subsets $C_i \subseteq A$, $1 \leq i \leq K$, such that $h(C_i) = \sum_{a \in C_i} h(a) \leq L$ for all $K$ subsets.

- Build a game composed of two gadgets
Random subset selection gadget

- Stochastically generates paths representing subsets of $A$: an element is selected in the subset if the upper edge is taken when leaving the corresponding state.

- All subsets are equiprobable.
Choice gadget

- $s_e$ leads to lower expected values but may be dangerous for the worst-case requirement
- $s_{wc}$ is always safe but induces a higher expected cost
Crux of the reduction

Establish that there exist values for thresholds and weights s.t.

(i) an optimal (i.e., minimizing the expectation while guaranteeing a given worst-case threshold) strategy for $P_1$ consists in choosing state $s_e$ only when the randomly generated subset $C \subseteq A$ satisfies $h(C) \leq L$;

(ii) this strategy satisfies the BWC problem if and only if there exist $K$ distinct subsets that verify this bound.
1. Context

2. BWC Synthesis

3. Mean-Payoff

4. Shortest Path

5. Conclusion
In a nutshell

- BWC framework combines worst-case and expected value requirements
  - a natural wish in many practical applications
  - few existing theoretical support
In a nutshell

- BWC framework combines worst-case and expected value requirements
  ▶ a natural wish in many practical applications
  ▶ few existing theoretical support

- Mean-payoff: additional modeling power for no complexity cost (decision-wise)

- Shortest path: harder than the worst-case, pseudo-polynomial with NP-hardness result
In a nutshell

- BWC framework combines worst-case and expected value requirements
  - a natural wish in many practical applications
  - few existing theoretical support

- Mean-payoff: additional modeling power for no complexity cost (decision-wise)

- Shortest path: harder than the worst-case, pseudo-polynomial with NP-hardness result

- In both cases, pseudo-polynomial memory is both sufficient and necessary
  - but strategies have natural representations based on states of the game and simple integer counters
Beyond BWC synthesis?

Possible future works include

- study of other quantitative objectives,

- extension of our results to more general settings (multi-dimension [CDHR10, CRR12], decidable classes of games with imperfect information [DDG+10], etc),

- application of the BWC problem to various practical cases.
Beyond BWC synthesis?

Possible future works include

- study of other quantitative objectives,

- extension of our results to more general settings (multi-dimension [CDHR10, CRR12], decidable classes of games with imperfect information [DDG+10], etc),

- application of the BWC problem to various practical cases.

Thanks!
Do not hesitate to discuss with us!
References I

D.P. Bertsekas and J.N. Tsitsiklis.
An analysis of stochastic shortest path problems.

Generalized mean-payoff and energy games.

K. Chatterjee, L. Doyen, M. Randour, and J.-F. Raskin.
Looking at mean-payoff and total-payoff through windows.

K. Chatterjee and M. Henzinger.
An $\mathcal{O}(n^2)$ time algorithm for alternating Büchi games.

K. Chatterjee, M. Randour, and J.-F. Raskin.
Strategy synthesis for multi-dimensional quantitative objectives.

C. Courcoubetis and M. Yannakakis.
The complexity of probabilistic verification.
References II

L. de Alfaro.
Formal verification of probabilistic systems.

L. de Alfaro.
Computing minimum and maximum reachability times in probabilistic systems.

Energy and mean-payoff games with imperfect information.

A. Ehrenfeucht and J. Mycielski.
Positional strategies for mean payoff games.

J. Filar and K. Vrieze.
Competitive Markov Decision Processes.

M.R. Garey and D.S. Johnson.

P.W. Glynn and D. Ormoneit.
Hoeffding’s inequality for uniformly ergodic Markov chains.
References III

Games through nested fixpoints.

D.B. Johnson and S.D. Kashdan.
Lower bounds for selection in X + Y and other multisets.

M. Jurdziński.
Deciding the winner in parity games is in UP ∩ co-UP.

T.M. Liggett and S.A. Lippman.
Stochastic games with perfect information and time average payoff.

M.L. Puterman.

M. Tracol.
Fast convergence to state-action frequency polytopes for MDPs.

U. Zwick and M. Paterson.
The complexity of mean payoff games on graphs.
References IV