Approximate verification and enumeration problems

Sylvain Peyronnet Michel de Rougemont Yann Strozecki

Université Paris Sud - Paris 11
Equipe ALGO

Bengalore, September 2012, ICTAC

Separation of systems

Representation by polytopes

The (randomized) polytope separator

Motivation

Our aim is to compare two sytems.

Do they have the same behavior?

Motivation

Our aim is to compare two sytems.

Do they have the same behavior?

When they do not, produce one or several witnesses.

Motivation

Our aim is to compare two sytems.

A specification of a protocol and an implementation each represented by a probabilistic automaton.

Do they have the same behavior?

When they do not, produce one or several witnesses.

Motivation

Our aim is to compare two sytems.

A specification of a protocol and an implementation each represented by a probabilistic automaton.

Do they have the same behavior?

Check that these automata accept all words with the same probability.

When they do not, produce one or several witnesses.

Motivation

Our aim is to compare two sytems.

A specification of a protocol and an implementation each represented by a probabilistic automaton.

Do they have the same behavior?

Check that these automata accept all words with the same probability.

When they do not, produce one or several witnesses.

Produce several words accepted with different probabilities.

Motivation

Our aim is to compare two sytems.

A specification of a protocol and an implementation each represented by a probabilistic automaton.

Do they have the same behavior?

Check that these automata accept all words with the same probability.

When they do not, produce one or several witnesses.

Produce several words accepted with different probabilities.

System modelisation

1. **Regular automata.** Equality of language, linear time.
2. **Probabilistic automata.** Equality of language, cubic time.

System modelisation

1. **Regular automata.** Equality of language, linear time.
2. **Probabilistic automata.** Equality of language, cubic time.
3. **Non-deterministic automata**. Equality of language, PSPACE-complete.

System modelisation

1. **Regular automata.** Equality of language, linear time.
2. **Probabilistic automata.** Equality of language, cubic time.
3. **Non-deterministic automata**. Equality of language, PSPACE-complete.
4. **Markov decision processes**. Equivalence: for each sequence of action generated by one MDP with some probability, there is one policy for the second such that the sequence is generated with larger probability. Problem is PSPACE-complete.

System modelisation

1. **Regular automata.** Equality of language, linear time.
2. **Probabilistic automata.** Equality of language, cubic time.
3. **Non-deterministic automata**. Equality of language, PSPACE-complete.
4. **Markov decision processes**. Equivalence: for each sequence of action generated by one MDP with some probability, there is one policy for the second such that the sequence is generated with larger probability. Problem is PSPACE-complete.

System modelisation

1. **Regular automata.** Equality of language, linear time.
2. **Probabilistic automata.** Equality of language, cubic time.
3. **Non-deterministic automata**. Equality of language, PSPACE-complete.
4. **Markov decision processes**. Equivalence: for each sequence of action generated by one MDP with some probability, there is one policy for the second such that the sequence is generated with larger probability. Problem is PSPACE-complete.

Method: representation by an appropriate structured and simple object $\rightarrow$ approximation.

System modelisation

1. **Regular automata.** Equality of language, linear time.
2. **Probabilistic automata.** Equality of language, cubic time.
3. **Non-deterministic automata**. Equality of language, PSPACE-complete.
4. **Markov decision processes**. Equivalence: for each sequence of action generated by one MDP with some probability, there is one policy for the second such that the sequence is generated with larger probability. Problem is PSPACE-complete.

Method: representation by an appropriate structured and simple object $\rightarrow$ approximation.
Sampling of this object $\rightarrow$ randomness.

System modelisation

1. **Regular automata.** Equality of language, linear time.
2. **Probabilistic automata.** Equality of language, cubic time.
3. **Non-deterministic automata**. Equality of language, PSPACE-complete.
4. **Markov decision processes**. Equivalence: for each sequence of action generated by one MDP with some probability, there is one policy for the second such that the sequence is generated with larger probability. Problem is PSPACE-complete.

Method: representation by an appropriate structured and simple object $\rightarrow$ approximation.
Sampling of this object $\rightarrow$ randomness.

Probabilistic automaton

Polynomial representation for the automaton A:

$$P_A = \sum_{w=\sigma_1 \ldots \sigma_n \in \Sigma^n} A(w) X_{1,\sigma_1} X_{2,\sigma_2} \ldots X_{n,\sigma_n}$$

where $A(w)$ is the probability that w is accepted by A.

A and B accept the same language $\iff P_A - P_B \equiv 0$.

Probabilistic automaton

Polynomial representation for the automaton A:

$$P_A = \sum_{w=\sigma_1\ldots\sigma_n\in\Sigma^n} A(w) X_{1,\sigma_1} X_{2,\sigma_2} \ldots X_{n,\sigma_n}$$

where $A(w)$ is the probability that w is accepted by A.

A and B accept the same language $\iff P_A - P_B \equiv 0$.

Theorem (Kiefer et al. 2011)

Let A and B be two automata with at most n states and m transitions. We can decide with probability $1-\varepsilon$ whether $A \equiv B$ in $O(nm\log(\varepsilon^{-1}))$ arithmetic operations.

Probabilistic automaton

Polynomial representation for the automaton A:

$$P_A = \sum_{w=\sigma_1\ldots\sigma_n \in \Sigma^n} A(w) X_{1,\sigma_1} X_{2,\sigma_2} \ldots X_{n,\sigma_n}$$

where $A(w)$ is the probability that w is accepted by A.

A and B accept the same language $\iff P_A - P_B \equiv 0$.

Theorem (Kiefer et al. 2011)

Let A and B be two automata with at most n states and m transitions. We can decide with probability $1-\varepsilon$ whether $A \equiv B$ in $O(nm\log(\varepsilon^{-1}))$ arithmetic operations.

When A and B are not equivalent, a minimal counter-example can be produced in the same time.

Probabilistic automaton

Polynomial representation for the automaton A:

$$P_A = \sum_{w=\sigma_1\ldots\sigma_n\in\Sigma^n} A(w)X_{1,\sigma_1}X_{2,\sigma_2}\ldots X_{n,\sigma_n}$$

where $A(w)$ is the probability that w is accepted by A.

A and B accept the same language $\iff P_A - P_B \equiv 0$.

Theorem (Kiefer et al. 2011)

Let A and B be two automata with at most n states and m transitions. We can decide with probability $1-\varepsilon$ whether $A \equiv B$ in $O(nm\log(\varepsilon^{-1}))$ arithmetic operations.
When A and B are not equivalent, a minimal counter-example can be produced in the same time.

Feature: the algorithm is probabilistic but the counter-examples are always produced in the same order.

Probabilistic automaton

Polynomial representation for the automaton A:

$$P_A = \sum_{w=\sigma_1\ldots\sigma_n\in\Sigma^n} A(w) X_{1,\sigma_1} X_{2,\sigma_2} \ldots X_{n,\sigma_n}$$

where $A(w)$ is the probability that w is accepted by A.

A and B accept the same language $\iff P_A - P_B \equiv 0$.

Theorem (Kiefer et al. 2011)

Let A and B be two automata with at most n states and m transitions. We can decide with probability $1-\varepsilon$ whether $A \equiv B$ in $O(nm\log(\varepsilon^{-1}))$ arithmetic operations.
When A and B are not equivalent, a minimal counter-example can be produced in the same time.

Feature: the algorithm is probabilistic but the counter-examples are always produced in the same order.

Separation of systems

Representation by polytopes

The (randomized) polytope separator

Representation of a polytope

We represent our system by polytopes of potentially large dimension.

Different way of representing a polytope:

1. Convex hull of a set of given points: $\mathcal{V}$-polytope.

Representation of a polytope

We represent our system by polytopes of potentially large dimension.

Different way of representing a polytope:

1. Convex hull of a set of given points: $\mathcal{V}$-polytope.
2. A set of linear inequalities: $\mathcal{H}$-polytope.

Representation of a polytope

We represent our system by polytopes of potentially large dimension.

Different way of representing a polytope:

1. Convex hull of a set of given points: $\mathcal{V}$-polytope.
2. A set of linear inequalities: $\mathcal{H}$-polytope.
3. A polynomial time algorithm to test if a point is in the polytope: strong membership oracle (SMO).

Representation of a polytope

We represent our system by polytopes of potentially large dimension.

Different way of representing a polytope:

1. Convex hull of a set of given points: $\mathcal{V}$-polytope.
2. A set of linear inequalities: $\mathcal{H}$-polytope.
3. A polynomial time algorithm to test if a point is in the polytope: strong membership oracle (SMO).

Remark: Deciding whether a point is in a $\mathcal{V}$ or $\mathcal{H}$-polytope can be done in polynomial time.

Representation of a polytope

We represent our system by polytopes of potentially large dimension.

Different way of representing a polytope:

1. Convex hull of a set of given points: $\mathcal{V}$-polytope.
2. A set of linear inequalities: $\mathcal{H}$-polytope.
3. A polynomial time algorithm to test if a point is in the polytope: strong membership oracle (SMO).

Remark: Deciding whether a point is in a $\mathcal{V}$ or $\mathcal{H}$-polytope can be done in polynomial time.

Density vectors of words and languages

Density vector (or k-gram) of a word:

$\mathsf{ustat}_k(w)$ contains the frequency at which the words of size k appear in the word w.

$\Sigma = \{a, b\}, w = aabab$
$\mathsf{ustat}_2(aabab) = (aa : \frac{1}{4}, ab : \frac{1}{2}, ba : \frac{1}{4}, bb : 0)$

Density vectors of words and languages

Density vector (or k-gram) of a word:

$\mathsf{ustat}_k(w)$ contains the frequency at which the words of size k appear in the word w.

$\Sigma = \{a, b\}, w = aabab$
$\mathsf{ustat}_2(aabab) = (aa : \frac{1}{4}, ab : \frac{1}{2}, ba : \frac{1}{4}, bb : 0)$

Statistics of an automaton

A is an automaton.
$H = \{\mathsf{ustat}_k(w) \mid A \text{ accepts } w\}$.

A accepts $a(ab)^+a^+$, that is $w = a(ab)^m b^n$.
$H = \{(aa : \frac{1}{2m+n}, ab : \frac{m}{2m+n}, ba : \frac{m-1}{2m+n}, bb : \frac{n}{2m+n})\}_{m,n\in\mathbb{N}}$

Density vectors of words and languages

Density vector (or k-gram) of a word:

$\mathsf{ustat}_k(w)$ contains the frequency at which the words of size k appear in the word w.

$\Sigma = \{a, b\}, w = aabab$
$\mathsf{ustat}_2(aabab) = (aa : \frac{1}{4}, ab : \frac{1}{2}, ba : \frac{1}{4}, bb : 0)$

Statistics of an automaton

A is an automaton.
$H = \{\mathsf{ustat}_k(w) \mid A \text{ accepts } w\}$.

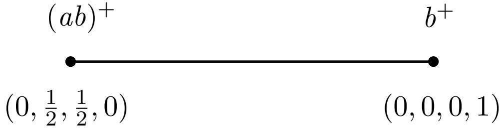
A accepts $a(ab)^+a^+$, that is $w = a(ab)^m b^n$.
$H = \{(aa : \frac{1}{2m+n}, ab : \frac{m}{2m+n}, ba : \frac{m-1}{2m+n}, bb : \frac{n}{2m+n})\}_{m,n\in\mathbb{N}}$

Geometrical embedding of an automaton

$$H_k(A) = \mathrm{ConvexHull}\left\{\mathsf{ustat}_k(w) \mid \ \text{for each } w \text{ compatible loop of } A^k\right\}$$

Geometrical embedding of an automaton

$$H_k(A) = \mathrm{ConvexHull}\left\{\mathsf{ustat}_k(w) \mid \text{ for each } w \text{ compatible loop of } A^k\right\}$$

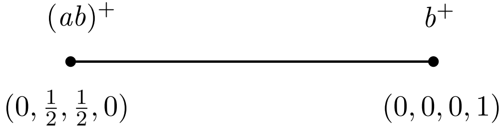


Theorem (Fischer, Magniez and De Rougemont, 2006)

Let A be an automaton with n states, we can construct a $\mathcal{V}$-representation (extremal vertices) of $H_k(A)$ of size $O(n^{|\Sigma|^k})$.

Geometrical embedding of an automaton

$$H_k(A) = \mathrm{ConvexHull}\left\{\mathsf{ustat}_k(w) \mid \text{ for each } w \text{ compatible loop of } A^k\right\}$$

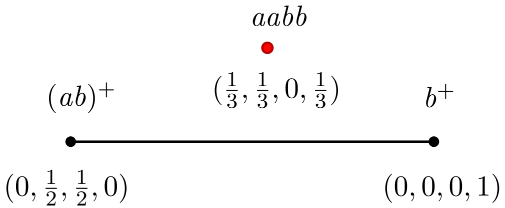


Theorem (Fischer, Magniez and De Rougemont, 2006)

Let A be an automaton with n states, we can construct a $\mathcal{V}$-representation (extremal vertices) of $H_k(A)$ of size $O(n^{|\Sigma|^k})$.

Geometrical embedding of an automaton

$$H_k(A) = \text{ConvexHull}\left\{\mathsf{ustat}_k(w) \mid \text{ for each } w \text{ compatible loop of } A^k\right\}$$



Theorem (Fischer, Magniez and De Rougemont, 2006)

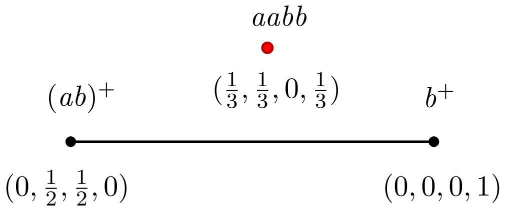
Let A be an automaton with n states, we can construct a $\mathcal{V}$-representation (extremal vertices) of $H_k(A)$ of size $O(n^{|\Sigma|^k})$.

Proposition

Let A be an automaton with n states and w a word it accepts, then $d(\mathsf{ustat}_k(w), H_k(A)) \leq \frac{n}{|w|-k+1}$.

Geometrical embedding of an automaton

$$H_k(A) = \mathrm{ConvexHull}\left\{\textsf{ustat}_k(w) \mid \text{ for each } w \text{ compatible loop of } A^k\right\}$$



Theorem (Fischer, Magniez and De Rougemont, 2006)

Let A be an automaton with n states, we can construct a $\mathcal{V}$-representation (extremal vertices) of $H_k(A)$ of size $O(n^{|\Sigma|^k})$.

Proposition

Let A be an automaton with n states and w a word it accepts, then $d(\textsf{ustat}_k(w), H_k(A)) \leq \frac{n}{|w|-k+1}$.

Distance between words

The edit distance with moves between v and w, denoted by $d(v, w)$, is the minimal number of operations to transform v into w.

An operation is:

- an insertion of a letter
- a deletion of a letter
- a move of a substring to another position

Distance between words

The edit distance with moves between v and w, denoted by $d(v, w)$, is the minimal number of operations to transform v into w.

An operation is:

- an insertion of a letter
- a deletion of a letter
- a move of a substring to another position

Theorem (Fischer, Magniez and De Rougemont, 2006)

Let v, w be two words of size n (large enough), then

$$d(v, w) \leq \frac{n}{k^2} \Rightarrow ||\mathsf{ustat}_k(v) - \mathsf{ustat}_k(w)||_1 \leq \frac{6.1}{k}$$

$$d(v, w) \geq \frac{5n}{k} \Rightarrow ||\mathsf{ustat}_k(v) - \mathsf{ustat}_k(w)||_1 \geq \frac{6.5}{k}$$

Conclusion: A (large) word w is far from any word accepted by A if and only if $\mathsf{ustat}_k(w)$ is far from $H_k(A)$.

Distance between words

The edit distance with moves between v and w, denoted by $d(v, w)$, is the minimal number of operations to transform v into w.

An operation is:

- an insertion of a letter
- a deletion of a letter
- a move of a substring to another position

Theorem (Fischer, Magniez and De Rougemont, 2006)

Let v, w be two words of size n (large enough), then

$$d(v, w) \leq \frac{n}{k^2} \Rightarrow ||\mathsf{ustat}_k(v) - \mathsf{ustat}_k(w)||_1 \leq \frac{6.1}{k}$$

$$d(v, w) \geq \frac{5n}{k} \Rightarrow ||\mathsf{ustat}_k(v) - \mathsf{ustat}_k(w)||_1 \geq \frac{6.5}{k}$$

Conclusion: A (large) word w is far from any word accepted by A if and only if $\mathsf{ustat}_k(w)$ is far from $H_k(A)$.

Separation of systems

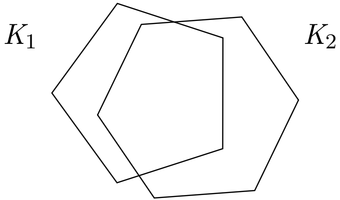
Representation by polytopes

The (randomized) polytope separator

Polytope separation

Two polytopes K_1 and K_2 given as SMOs.

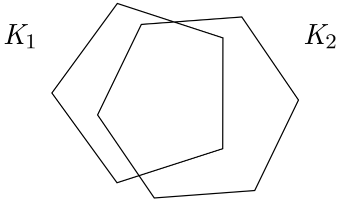
Problem:



Polytope separation

Two polytopes K_1 and K_2 given as SMOs.

Problem: Are they equals?



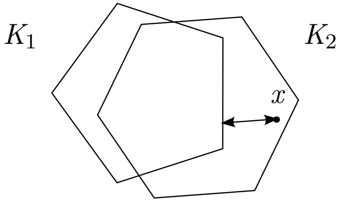
Polytope separation

Two polytopes K_1 and K_2 given as SMOs.

Problem: Are they L_1 far?

The parameters: the dimension of K_1 and K_2 and

$$d_{vol}(K_1, K_2) = \frac{\mathcal{V}(K_1 \triangle K_2)}{\mathcal{V}(K_1) + \mathcal{V}(K_2)}$$



$$d(x, K_1) = \min_{y \in K_1} ||x - y||_1$$

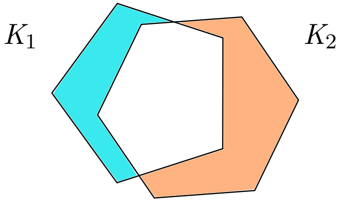
Polytope separation

Two polytopes K_1 and K_2 given as SMOs.

Problem: Are they L_1 far?

The parameters: the dimension of K_1 and K_2 and

$$d_{vol}(K_1, K_2) = \frac{\mathcal{V}(K_1 \triangle K_2)}{\mathcal{V}(K_1) + \mathcal{V}(K_2)}$$



Refined problem: Generate points uniformly in $K_1 \triangle K_2$.

Polytope separation

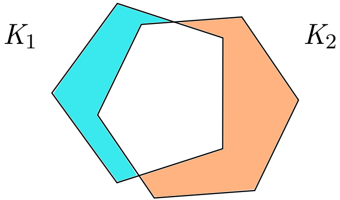
Two polytopes K_1 and K_2 given as SMOs.

Problem: Are they L_1 far?

The parameters: the dimension of K_1 and K_2 and

$$d_{vol}(K_1, K_2) = \frac{\mathcal{V}(K_1 \triangle K_2)}{\mathcal{V}(K_1) + \mathcal{V}(K_2)}$$

Refined problem: Generate points uniformly in $K_1 \triangle K_2$.



A solution for $\mathcal{H}$-polytopes

K_1, K_2 $\mathcal{H}$-polytopes: $\begin{cases} K_1 = \cap_i E_i \\ K_2 = \cap_j F_j \end{cases}$

each E_i or F_j is a half space given by a linear inequality.

Let $\bar{E}$ be the complement of the halfspace E.

$K_2 \setminus K_1 = \cup_j (\bar{F}_j \cap K_1)$

A solution for $\mathcal{H}$-polytopes

K_1, K_2 $\mathcal{H}$-polytopes: $\begin{cases} K_1 = \cap_i E_i \\ K_2 = \cap_j F_j \end{cases}$

each E_i or F_j is a half space given by a linear inequality.

Let $\bar{E}$ be the complement of the halfspace E.
$K_2 \setminus K_1 = \cup_j (\bar{F}_j \cap K_1)$

For each j, check whether the system $\bar{F}_j \cap K_1$ has a solution. If it is the case output one solution.

A solution for $\mathcal{H}$-polytopes

K_1, K_2 $\mathcal{H}$-polytopes: $\begin{cases} K_1 = \cap_i E_i \\ K_2 = \cap_j F_j \end{cases}$

each E_i or F_j is a half space given by a linear inequality.

Let $\bar{E}$ be the complement of the halfspace E.

$K_2 \setminus K_1 = \cup_j (\bar{F}_j \cap K_1)$

For each j, check whether the system $\bar{F}_j \cap K_1$ has a solution. If it is the case output one solution.

Test whether $K_1 \neq K_2$ in time polynomial in the number of inequalities defining K_1 and K_2.

A solution for $\mathcal{H}$-polytopes

K_1, K_2 $\mathcal{H}$-polytopes: $\begin{cases} K_1 = \cap_i E_i \\ K_2 = \cap_j F_j \end{cases}$
each E_i or F_j is a half space given by a linear inequality.

Let $\bar{E}$ be the complement of the halfspace E.
$K_2 \setminus K_1 = \cup_j (\bar{F}_j \cap K_1)$

For each j, check whether the system $\bar{F}_j \cap K_1$ has a solution. If it is the case output one solution.

Test whether $K_1 \neq K_2$ in time polynomial in the number of inequalities defining K_1 and K_2.

Hausdorff distance for $\mathcal{V}$-polytopes

Stronger aim : compute the Hausdorff distance between two polytopes K_1, K_2.

$$d_h(K_1, K_2) = \max_{x \in K_1} d(x, K_2)$$

Facts:

- $d(x, K_2)$ can be computed in polynomial time (LP) when K_2 is a $\mathcal{V}$ or $\mathcal{H}$-polytope.

Hausdorff distance for $\mathcal{V}$-polytopes

Stronger aim : compute the Hausdorff distance between two polytopes K_1, K_2.

$$d_h(K_1, K_2) = \max_{x \in K_1} d(x, K_2)$$

Facts:

- $d(x, K_2)$ can be computed in polynomial time (LP) when K_2 is a $\mathcal{V}$ or $\mathcal{H}$-polytope.
- $\max_{x \in K_1} d(x, K_2)$ is attained by x an extremal point of K_1.

Hausdorff distance for $\mathcal{V}$-polytopes

Stronger aim : compute the Hausdorff distance between two polytopes K_1, K_2.

$$d_h(K_1, K_2) = \max_{x \in K_1} d(x, K_2)$$

Facts:

- $d(x, K_2)$ can be computed in polynomial time (LP) when K_2 is a $\mathcal{V}$ or $\mathcal{H}$-polytope.
- $\max_{x \in K_1} d(x, K_2)$ is attained by x an extremal point of K_1.

Conclusion:

- In P for $\mathcal{V}$-polytopes.

Hausdorff distance for $\mathcal{V}$-polytopes

Stronger aim : compute the Hausdorff distance between two polytopes K_1, K_2.

$$d_h(K_1, K_2) = \max_{x \in K_1} d(x, K_2)$$

Facts:

- $d(x, K_2)$ can be computed in polynomial time (LP) when K_2 is a $\mathcal{V}$ or $\mathcal{H}$-polytope.
- $\max_{x \in K_1} d(x, K_2)$ is attained by x an extremal point of K_1.

Conclusion:

- In P for $\mathcal{V}$-polytopes.
- NP-hard to approximate for $\mathcal{H}$-polytopes.

Hausdorff distance for $\mathcal{V}$-polytopes

Stronger aim : compute the Hausdorff distance between two polytopes K_1, K_2.

$$d_h(K_1, K_2) = \max_{x \in K_1} d(x, K_2)$$

Facts:

- $d(x, K_2)$ can be computed in polynomial time (LP) when K_2 is a $\mathcal{V}$ or $\mathcal{H}$-polytope.
- $\max_{x \in K_1} d(x, K_2)$ is attained by x an extremal point of K_1.

Conclusion:

- In P for $\mathcal{V}$-polytopes.
- NP-hard to approximate for $\mathcal{H}$-polytopes.
- Even harder for SMOs.

Hausdorff distance for $\mathcal{V}$-polytopes

Stronger aim : compute the Hausdorff distance between two polytopes K_1, K_2.

$$d_h(K_1, K_2) = \max_{x \in K_1} d(x, K_2)$$

Facts:

- $d(x, K_2)$ can be computed in polynomial time (LP) when K_2 is a $\mathcal{V}$ or $\mathcal{H}$-polytope.
- $\max_{x \in K_1} d(x, K_2)$ is attained by x an extremal point of K_1.

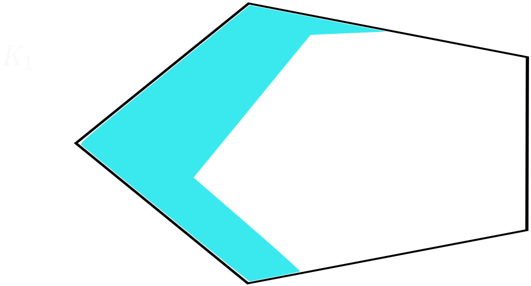
Conclusion:

- In P for $\mathcal{V}$-polytopes.
- NP-hard to approximate for $\mathcal{H}$-polytopes.
- Even harder for SMOs.

Random walk on a polytope

Algorithm used to compute the volume of a polytope [Dyer et al., Lovasz, Simonovitz . . .].

First step: isotropic position i.e. rounding the polytope by linear transformations.

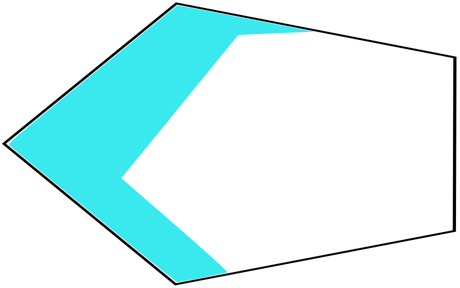


Random walk on a polytope

Algorithm used to compute the volume of a polytope [Dyer et al., Lovasz, Simonovitz . . .].

First step: isotropic position i.e. rounding the polytope by linear transformations.

K_1

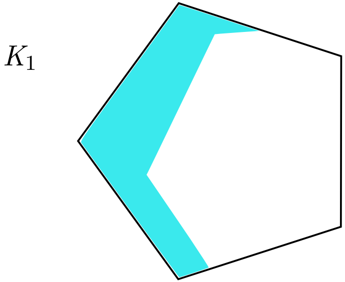


Random walk on a polytope

Algorithm used to compute the volume of a polytope [Dyer et al., Lovasz, Simonovitz ...].

First step: isotropic position i.e. rounding the polytope by linear transformations.

Second step: random walk (ball walk) with polynomial mixing time.

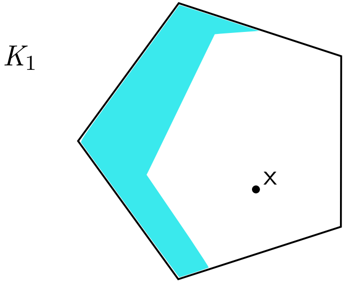


Random walk on a polytope

Algorithm used to compute the volume of a polytope [Dyer et al., Lovasz, Simonovitz ...].

First step: isotropic position i.e. rounding the polytope by linear transformations.

Second step: random walk (ball walk) with polynomial mixing time.

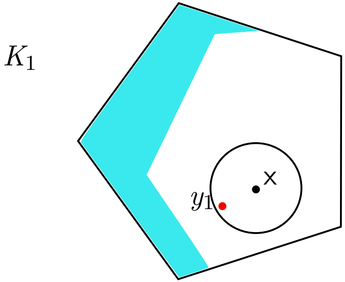


Random walk on a polytope

Algorithm used to compute the volume of a polytope [Dyer et al., Lovasz, Simonovitz . . .].

First step: isotropic position i.e. rounding the polytope by linear transformations.

Second step: random walk (ball walk) with polynomial mixing time.

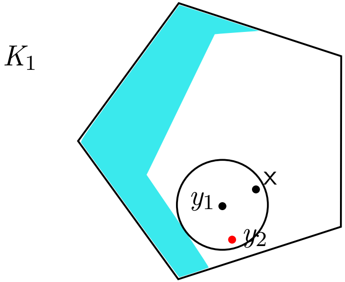


Random walk on a polytope

Algorithm used to compute the volume of a polytope [Dyer et al., Lovasz, Simonovitz . . .].

First step: isotropic position i.e. rounding the polytope by linear transformations.

Second step: random walk (ball walk) with polynomial mixing time.

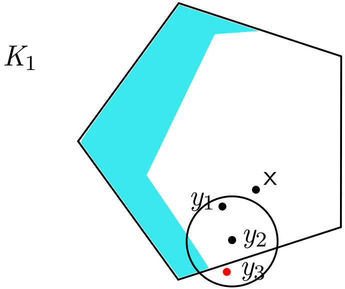


Random walk on a polytope

Algorithm used to compute the volume of a polytope [Dyer et al., Lovasz, Simonovitz . . .].

First step: isotropic position i.e. rounding the polytope by linear transformations.

Second step: random walk (ball walk) with polynomial mixing time.

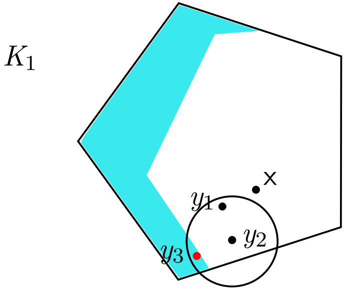


Random walk on a polytope

Algorithm used to compute the volume of a polytope [Dyer et al., Lovasz, Simonovitz . . .].

First step: isotropic position i.e. rounding the polytope by linear transformations.

Second step: random walk (ball walk) with polynomial mixing time.



Polytope separator

Theorem

Let K_1 and K_2 be two polytopes of dimension n, given as SMOs. The Polytope Separator *algorithm outputs a point x such that $x \in K_1 \triangle K_2$ with probability greater than $2/3$. Moreover, the running time of this algorithm is polynomial in n and $d_{vol}(K_1, K_2)^{-1}$.*

- Complexity is polynomial but large: $O(n^5)$ checks whether a point is in the polytope.

Polytope separator

Theorem

Let K_1 and K_2 be two polytopes of dimension n, given as SMOs. The Polytope Separator *algorithm outputs a point x such that $x \in K_1 \triangle K_2$ with probability greater than $2/3$. Moreover, the running time of this algorithm is polynomial in n and $d_{vol}(K_1, K_2)^{-1}$.*

- Complexity is polynomial but large: $O(n^5)$ checks whether a point is in the polytope.
- Use another random walk like hit and run. Should go down to $O(n^4)$ and even $O(n^3)$.

Polytope separator

Theorem

Let K_1 and K_2 be two polytopes of dimension n, given as SMOs. The Polytope Separator *algorithm outputs a point x such that $x \in K_1 \triangle K_2$ with probability greater than $2/3$. Moreover, the running time of this algorithm is polynomial in n and $d_{vol}(K_1, K_2)^{-1}$.*

- Complexity is polynomial but large: $O(n^5)$ checks whether a point is in the polytope.
- Use another random walk like hit and run. Should go down to $O(n^4)$ and even $O(n^3)$.
- Adapt the algorithm to the representations we have in the applications: $\mathcal{V}$-representation or projection of a $\mathcal{H}$-representation.

Polytope separator

Theorem

Let K_1 and K_2 be two polytopes of dimension n, given as SMOs. The Polytope Separator *algorithm outputs a point x such that $x \in K_1 \triangle K_2$ with probability greater than $2/3$. Moreover, the running time of this algorithm is polynomial in n and $d_{vol}(K_1, K_2)^{-1}$.*

- Complexity is polynomial but large: $O(n^5)$ checks whether a point is in the polytope.
- Use another random walk like hit and run. Should go down to $O(n^4)$ and even $O(n^3)$.
- Adapt the algorithm to the representations we have in the applications: $\mathcal{V}$-representation or projection of a $\mathcal{H}$-representation.

Application to non deterministic automata

Theorem

Given two non deterministic automata A and B with alphabet Σ and at most n states, let $\varepsilon = \frac{1}{k}$ we can generate ε-separating words when they exist in time polynomial in N, where $N = n^{|\Sigma|^k}$.

1. Generate the extremal vertices of $H_k(A)$ and $H_k(B)$. There are at most N vertices.

Application to non deterministic automata

Theorem

Given two non deterministic automata A and B with alphabet Σ and at most n states, let $\varepsilon = \frac{1}{k}$ we can generate ε-separating words when they exist in time polynomial in N, where $N = n^{|\Sigma|^k}$.

1. Generate the extremal vertices of $H_k(A)$ and $H_k(B)$. There are at most N vertices.
2. Find an extremal point x in say $H_k(A)$ which is at maximum distance d of $H_k(B)$ in time $poly(N)$.

Application to non deterministic automata

Theorem

Given two non deterministic automata A and B with alphabet Σ and at most n states, let $\varepsilon = \frac{1}{k}$ we can generate ε-separating words when they exist in time polynomial in N, where $N = n^{|\Sigma|^k}$.

1. Generate the extremal vertices of $H_k(A)$ and $H_k(B)$. There are at most N vertices.
2. Find an extremal point x in say $H_k(A)$ which is at maximum distance d of $H_k(B)$ in time $poly(N)$.
3. If $d \leq \varepsilon$ then A is close to B.

Application to non deterministic automata

Theorem

Given two non deterministic automata A and B with alphabet Σ and at most n states, let $\varepsilon = \frac{1}{k}$ we can generate ε-separating words when they exist in time polynomial in N, where $N = n^{|\Sigma|^k}$.

1. Generate the extremal vertices of $H_k(A)$ and $H_k(B)$. There are at most N vertices.
2. Find an extremal point x in say $H_k(A)$ which is at maximum distance d of $H_k(B)$ in time $poly(N)$.
3. If $d \leq \varepsilon$ then A is close to B.
4. The point x is the ustat$_k$ of some loop accepted by A and this loop w can be found quickly.

Application to non deterministic automata

Theorem

Given two non deterministic automata A and B with alphabet Σ and at most n states, let $\varepsilon = \frac{1}{k}$ we can generate ε-separating words when they exist in time polynomial in N, where $N = n^{|\Sigma|^k}$.

1. Generate the extremal vertices of $H_k(A)$ and $H_k(B)$. There are at most N vertices.
2. Find an extremal point x in say $H_k(A)$ which is at maximum distance d of $H_k(B)$ in time $poly(N)$.
3. If $d \leq \varepsilon$ then A is close to B.
4. The point x is the ustat$_k$ of some loop accepted by A and this loop w can be found quickly.
5. There is a word $uw^n v$ accepted by A and ε-far from the words accepted by B.

Application to non deterministic automata

Theorem

Given two non deterministic automata A and B with alphabet Σ and at most n states, let $\varepsilon = \frac{1}{k}$ we can generate ε-separating words when they exist in time polynomial in N, where $N = n^{|\Sigma|^k}$.

1. Generate the extremal vertices of $H_k(A)$ and $H_k(B)$. There are at most N vertices.
2. Find an extremal point x in say $H_k(A)$ which is at maximum distance d of $H_k(B)$ in time $poly(N)$.
3. If $d \leq \varepsilon$ then A is close to B.
4. The point x is the ustat$_k$ of some loop accepted by A and this loop w can be found quickly.
5. There is a word $uw^n v$ accepted by A and ε-far from the words accepted by B.

Thank you!

Questions?

A word on markov decision processes

- State action frequency: to every strategy the ustat$_k$ of the expectation of the trace when the length of the run tends to ∞.
- Consider the k-product of the MDP. From states in S and actions in Σ to $(S \times \Sigma)^k$.

A word on markov decision processes

- State action frequency: to every strategy the ustat_k of the expectation of the trace when the length of the run tends to ∞.
- Consider the k-product of the MDP. From states in S and actions in Σ to $(S \times \Sigma)^k$.
- Conservation of the limit distribution over the k-product of the MDP. Yields a $\mathcal{H}$-polytope.

A word on markov decision processes

- State action frequency: to every strategy the ustat_k of the expectation of the trace when the length of the run tends to ∞.
- Consider the k-product of the MDP. From states in S and actions in Σ to $(S \times \Sigma)^k$.
- Conservation of the limit distribution over the k-product of the MDP. Yields a $\mathcal{H}$-polytope.
- Projection of the polytope: forget the states of the MDP [De Rougemont and Tracol].

A word on markov decision processes

- State action frequency: to every strategy the ustat_k of the expectation of the trace when the length of the run tends to ∞.
- Consider the k-product of the MDP. From states in S and actions in Σ to $(S \times \Sigma)^k$.
- Conservation of the limit distribution over the k-product of the MDP. Yields a $\mathcal{H}$-polytope.
- Projection of the polytope: forget the states of the MDP [De Rougemont and Tracol].
- **Drawback**: separating action frequency vector only.

A word on markov decision processes

- State action frequency: to every strategy the ustat_k of the expectation of the trace when the length of the run tends to ∞.
- Consider the k-product of the MDP. From states in S and actions in Σ to $(S \times \Sigma)^k$.
- Conservation of the limit distribution over the k-product of the MDP. Yields a $\mathcal{H}$-polytope.
- Projection of the polytope: forget the states of the MDP [De Rougemont and Tracol].
- **Drawback**: separating action frequency vector only.