On the Approximability of Weighted Model Integration over DNF Structures [1]

Ralph Abboud, İsmail İlkan Ceylan, Radoslav Dimitrov

09 July 2020

MCW@SAT2020

Weighted Model Counting

ϕ propositional formula

Weighted Model Counting

ϕ propositional formula

$w: \mathcal{A}(\phi) \rightarrow \mathbb{R}$ weight function

Weighted Model Counting

ϕ propositional formula

$w: \mathcal{A}(\phi) \to \mathbb{R}$ weight function

$$\mathrm{WMC}(\phi) \qquad \sum_{\nu \vDash \phi} w(\phi)$$

Weighted Model Counting

ϕ propositional formula

$w: \mathcal{A}(\phi) \rightarrow \mathbb{R}$ weight function

---

$\mathrm{WMC}(\phi)$ $\sum_{\nu \vDash \phi} w(\phi)$

Applications

Probabilistic Graphical Models

Probabilistic Logic Programming

Probabilistic Databases

Probabilistic Knowledge Bases

Weighted Model Integration

- Weighted model counting only applies over discrete variables.

Weighted Model Integration

- Weighted model counting only applies over discrete variables.

- Inspired by SMT, Weighted model integration generalizes weighted model counting over discrete and continuous variables.

Weighted Model Integration

- Weighted model counting only applies over discrete variables.

- Inspired by SMT, Weighted model integration generalizes weighted model counting over discrete and continuous variables.

$\mathbf{X}$ set of n real variables

Weighted Model Integration

- Weighted model counting only applies over discrete variables.

- Inspired by SMT, Weighted model integration generalizes weighted model counting over discrete and continuous variables.

$\mathbf{X}$ set of n real variables

$\mathbf{V}$ set of m Boolean variables

Weighted Model Integration

$c_1 x_1 + \ldots + c_i\, x_i \bowtie c$

LRA Atom, $x_i \in \mathrm{X}$, $\bowtie \in \{<, \leq, >, \geq, =, \neq\}$

Weighted Model Integration

$c_1 x_1 + \ldots + c_i\, x_i \bowtie c$ LRA Atom, $x_i \in \mathrm{X}$, $\bowtie \in \{<, \leq, >, \geq, =, \neq\}$

$\mathrm{atoms}(\mathrm{X}, \mathrm{V})$ propositional and LRA atoms over $\mathrm{X} \cup \mathrm{V}$

Weighted Model Integration

$c_1 x_1 + \ldots + c_i\, x_i \bowtie \mathrm{c}$ LRA Atom, $x_i \in \mathrm{X}$, $\bowtie \in \{<, \leq, >, \geq, =, \neq\}$

$\mathrm{atoms}(\mathrm{X}, \mathrm{V})$ propositional and LRA atoms over $\mathrm{X} \cup \mathrm{V}$

ϕ propositional formula over $X \cup \mathrm{V}$

Weighted Model Integration

$c_1 x_1 + \ldots + c_i\, x_i \bowtie \mathrm{c}$ — LRA Atom, $x_i \in \mathrm{X}$, $\bowtie \in \{<, \leq, >, \geq, =, \neq\}$

$\mathrm{atoms}(\mathrm{X}, \mathrm{V})$ — propositional and LRA atoms over $\mathrm{X} \cup \mathrm{V}$

ϕ — propositional formula over $X \cup \mathrm{V}$

$w\!: \mathbb{R}^n \times \mathbb{B}^m \rightarrow \mathbb{R}$ — weight function over $X \cup \mathrm{V}$

Weighted Model Integration

$c_1 x_1 + \ldots + c_i\, x_i \bowtie \mathrm{c}$	LRA Atom, $x_i \in \mathrm{X}$, $\bowtie \in \{<, \leq, >, \geq, =, \neq\}$
$\mathrm{atoms}(\mathrm{X}, \mathrm{V})$	propositional and LRA atoms over $\mathrm{X} \cup \mathrm{V}$
ϕ	propositional formula over $X \cup \mathrm{V}$
$w\colon\ \mathbb{R}^n \times \mathbb{B}^m \to \mathbb{R}$	weight function over $X \cup \mathrm{V}$

$\mathrm{WMI}(\phi)$

$$\sum_{v} \int_{x_\phi} w(x, v)\, dx$$

where v is a Boolean assignment over V, x_ϕ denotes valuations of X satisfying ϕ.

Weighted Model Integration: Weight Function

Weighted Model Integration: Weight Function

$w\colon \mathbb{R}^n \times \mathbb{B}^m \to \mathbb{R}$ weight function over $X \cup V$

Weighted Model Integration: Weight Function

$w: \mathbb{R}^n \times \mathbb{B}^m \rightarrow \mathbb{R}$ weight function over $X \cup V$

Weighted Model Integration: Weight Function

$w\colon \mathbb{R}^n \times \mathbb{B}^m \rightarrow \mathbb{R}$ weight function over $X \cup V$

For weight functions in WMI, it is common to factorize [2] w as a product of m Boolean literal weights and a density function over real variables, i.e.,:

$$w(x, v) = w_x(x) \prod_{i=1}^{m} w_b(p_i).$$

Special Cases

$(x \vee y) \wedge (y \vee z)$

Conjunctive Normal Form (CNF)

Special Cases

$(x \vee y) \wedge (y \vee z)$ Conjunctive Normal Form (CNF)

$(x \wedge y) \vee (y \wedge z)$ Disjunctive Normal Form (DNF)

Special Cases

$(x \vee y) \wedge (y \vee z)$ Conjunctive Normal Form (CNF)

$(x \wedge y) \vee (y \wedge z)$ Disjunctive Normal Form (DNF)

$\Rightarrow$ Special cases WMI(CNF), WMI(DNF)

Special Cases

$(x \vee y) \wedge (y \vee z)$ Conjunctive Normal Form (CNF)

$(x \wedge y) \vee (y \wedge z)$ Disjunctive Normal Form (DNF)

➡ Special cases WMI(CNF), WMI(DNF)

Both WMI and WMC are #P-hard for exact solving. Hence, we study WMI within the context of approximate solving.

Approximation Hardness

How hard is it to approximate WMI(CNF) and WMI(DNF)?

Approximation Hardness

How hard is it to approximate WMI(CNF) and WMI(DNF)?

	WMC		WMI
CNF	NP-hard [3]	→	NP-hard
DNF	FPRAS [4]	→	?

Result:

Show that WMI(DNF) admits an FPRAS for concave weight functions

Result builds on existing FPRAS algorithms for WMC(DNF) and volume computation for the union of convex bodies

ApproxUnion: Volume of Union of Convex Bodies [5]

Consider k convex bodies. We wish to compute the volume of their union.

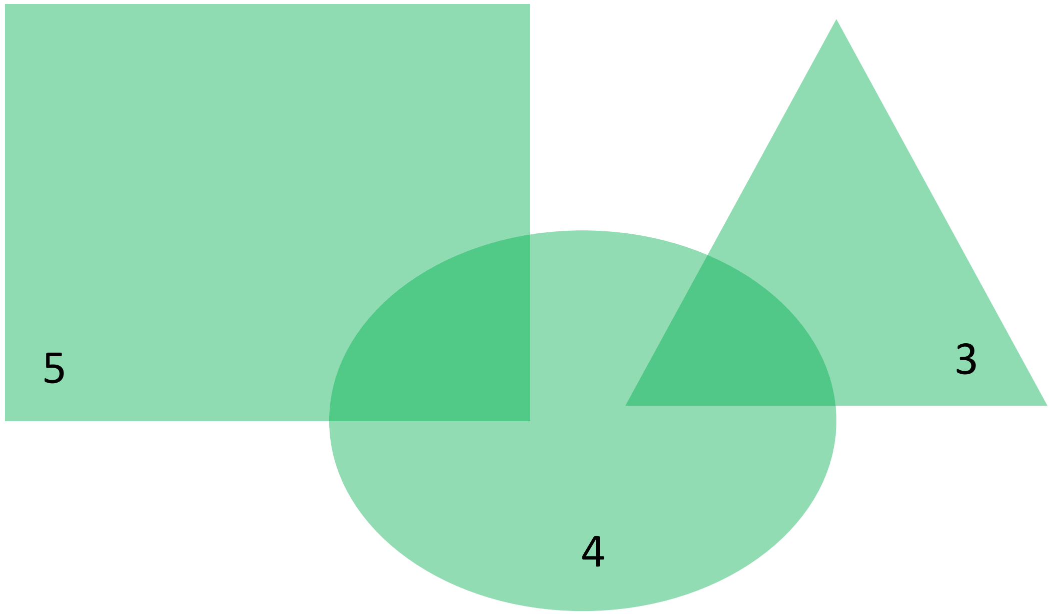
ApproxUnion: Volume of Union of Convex Bodies [5]

Consider k convex bodies. We wish to compute the volume of their union.

1 Compute individual body volumes

ApproxUnion: Volume of Union of Convex Bodies [5]

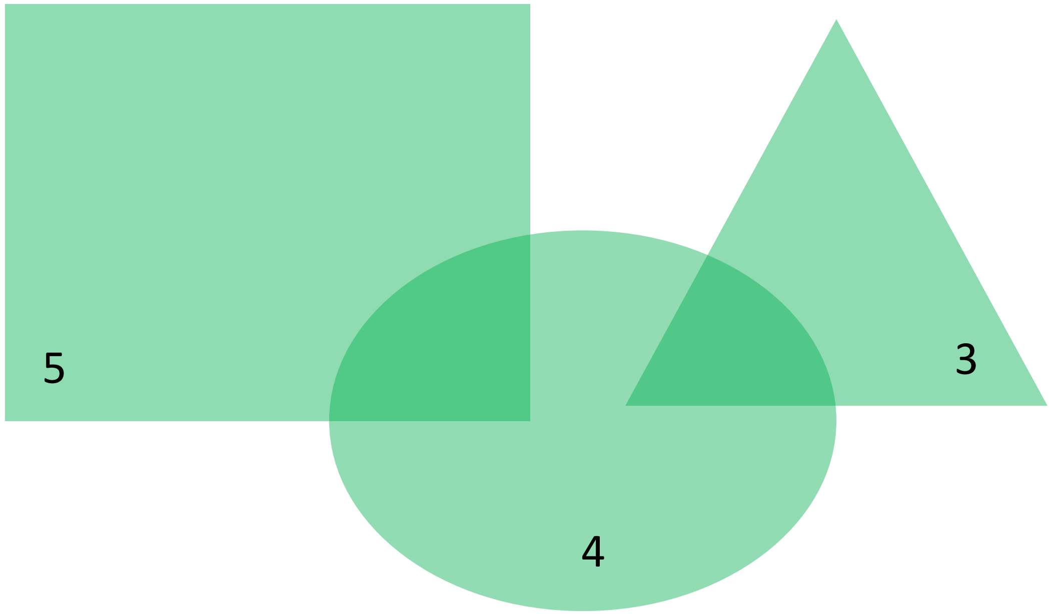
Consider k convex bodies. We wish to compute the volume of their union.



1 Compute individual body volumes

ApproxUnion: Volume of Union of Convex Bodies [5]

Consider k convex bodies. We wish to compute the volume of their union.

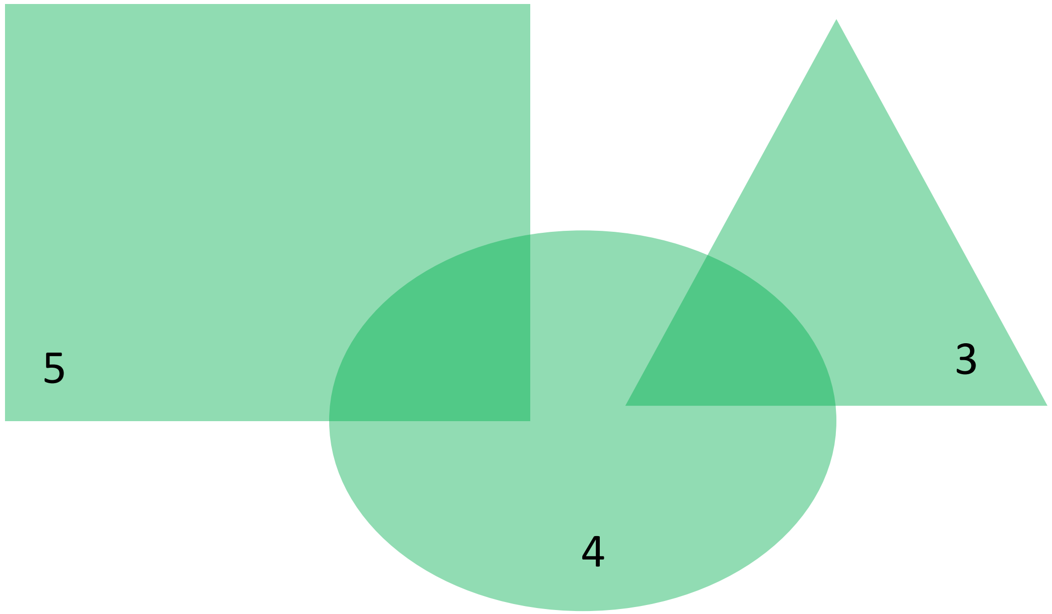


1 Compute individual body volumes

Run until Step 4 executes T times:

ApproxUnion: Volume of Union of Convex Bodies [5]

Consider k convex bodies. We wish to compute the volume of their union.



1 Compute individual body volumes

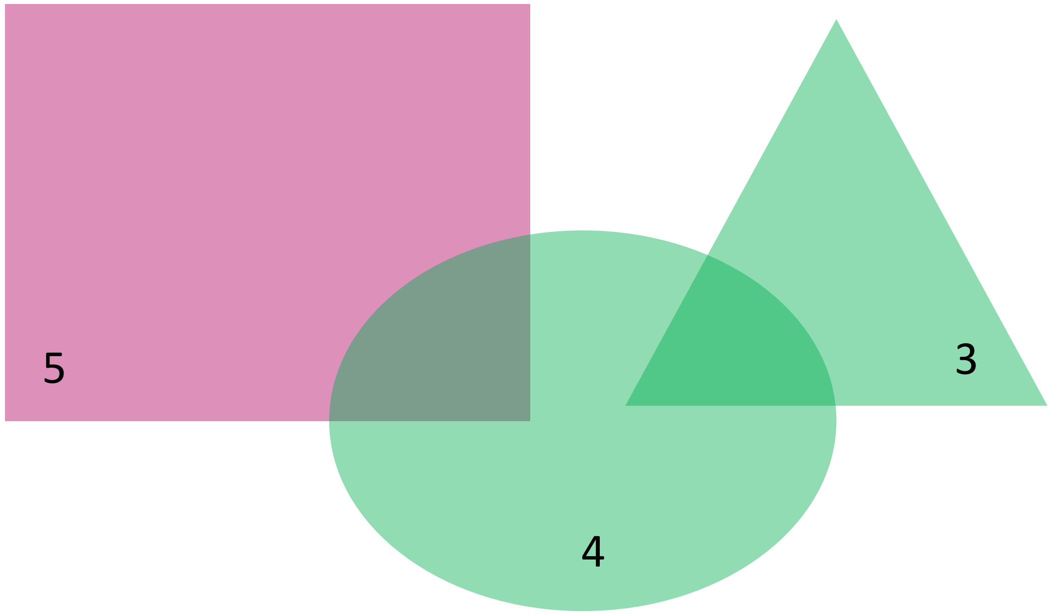
Run until Step 4 executes T times:

2 Randomly sample body

sampling probabilities proportional to body volume

ApproxUnion: Volume of Union of Convex Bodies [5]

Consider k convex bodies. We wish to compute the volume of their union.



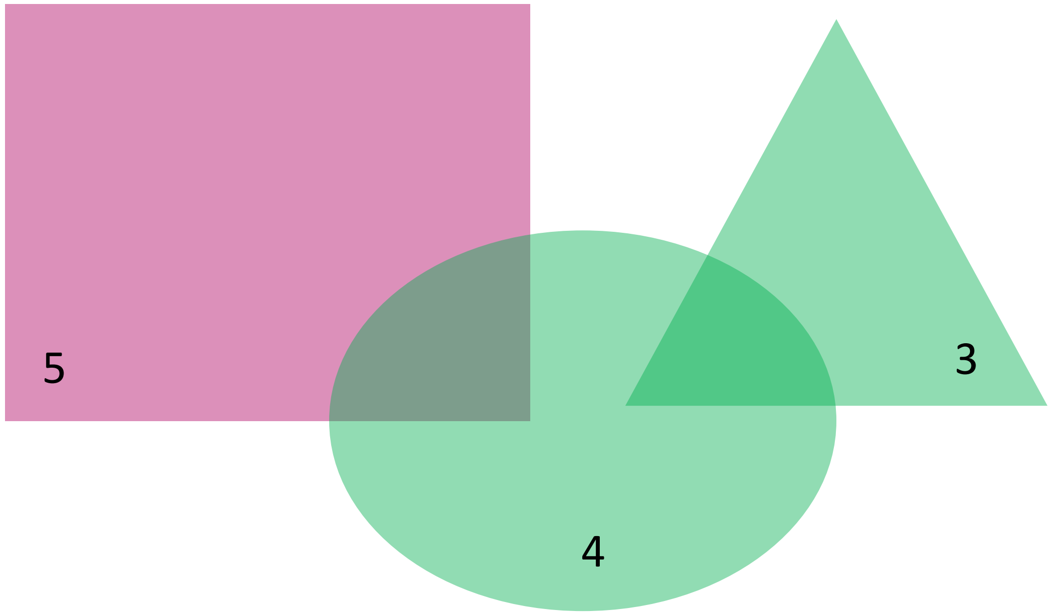
1 Compute individual body volumes

Run until Step 4 executes T times:

2 Randomly sample body
sampling probabilities proportional to body volume

ApproxUnion: Volume of Union of Convex Bodies [5]

Consider k convex bodies. We wish to compute the volume of their union.



1 Compute individual body volumes

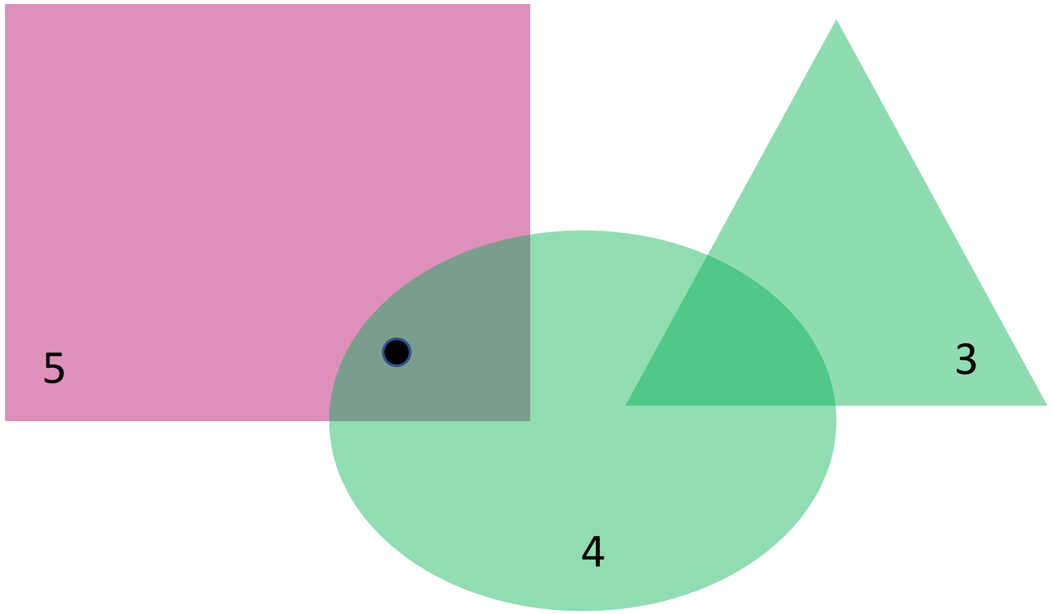
Run until Step 4 executes T times:

2 Randomly sample body
sampling probabilities proportional to body volume

3 Uniformly sample point in body

ApproxUnion: Volume of Union of Convex Bodies [5]

Consider k convex bodies. We wish to compute the volume of their union.



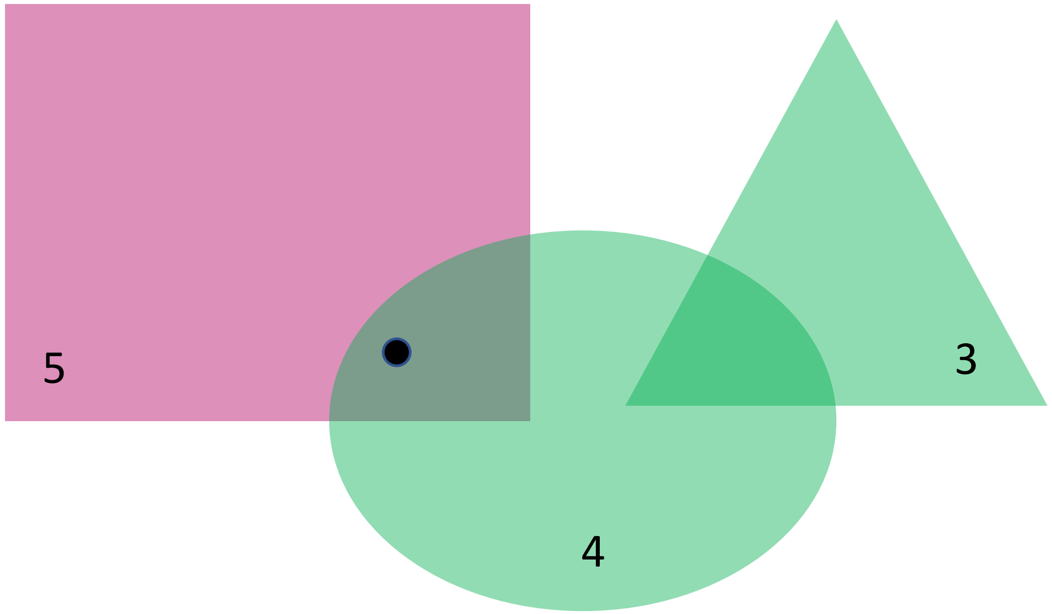
1. Compute individual body volumes

Run until Step 4 executes T times:

2. Randomly sample body
   sampling probabilities proportional to body volume
3. Uniformly sample point in body

ApproxUnion: Volume of Union of Convex Bodies [5]

Consider k convex bodies. We wish to compute the volume of their union.



1 Compute individual body volumes

Run until Step 4 executes T times:

2 Randomly sample body

sampling probabilities proportional to body volume

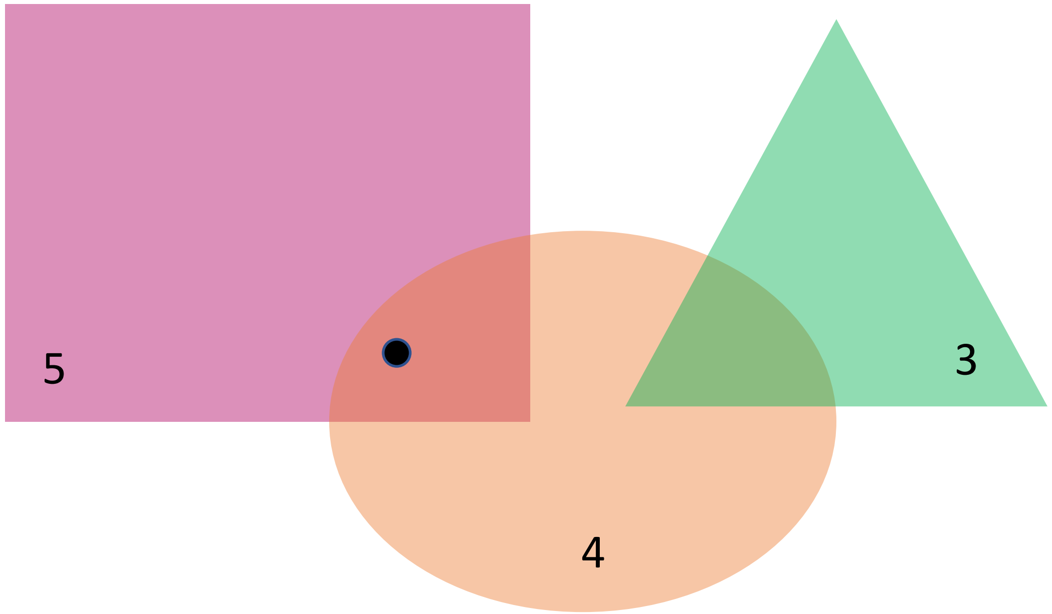
3 Uniformly sample point in body

4 Uniformly sample another body

If sampled point in this body, Success!
Repeat Step 2, Else repeat Step 4.

ApproxUnion: Volume of Union of Convex Bodies [5]

Consider *k* convex bodies. We wish to compute the volume of their union.



1 Compute individual body volumes

Run until Step 4 executes T times:

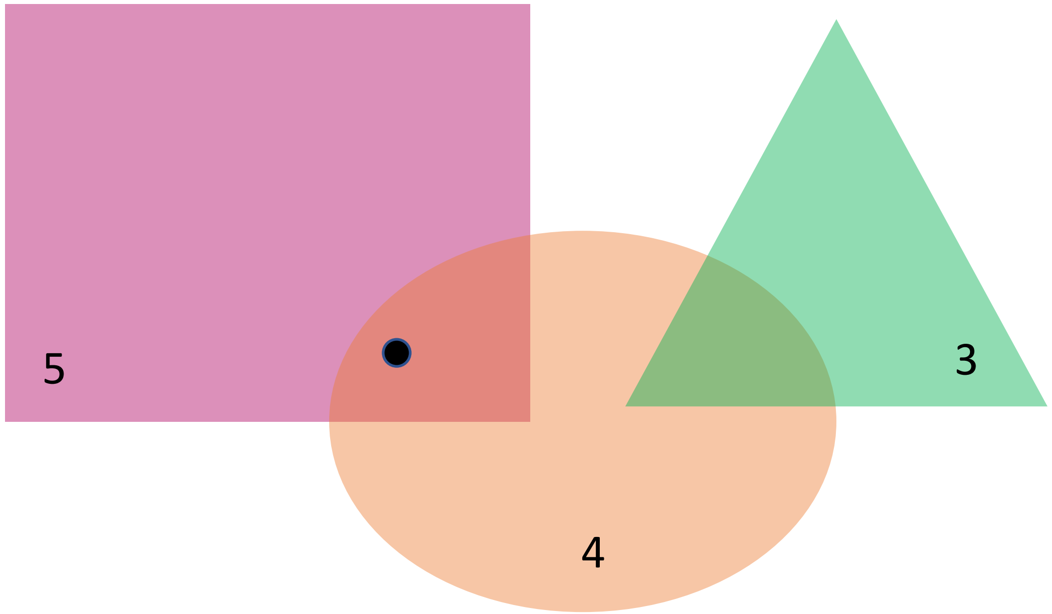
2 Randomly sample body
sampling probabilities proportional to body volume

3 Uniformly sample point in body

4 Uniformly sample another body
If sampled point in this body, Success!
Repeat Step 2, Else repeat Step 4.

ApproxUnion: Volume of Union of Convex Bodies [5]

Consider k convex bodies. We wish to compute the volume of their union.



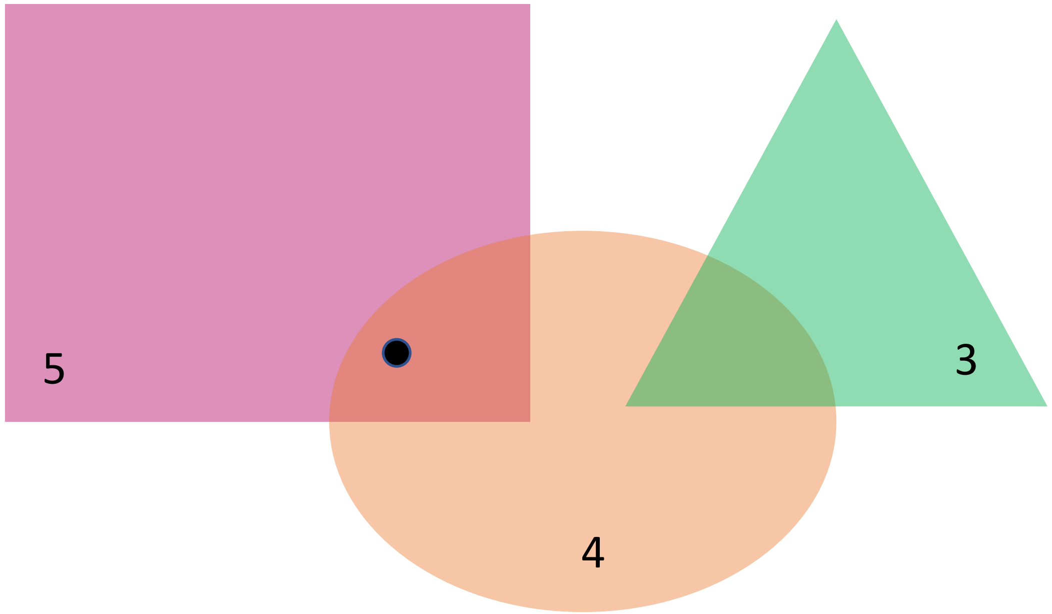
1. Compute individual body volumes

Run until Step 4 executes T times:

2. Randomly sample body
   sampling probabilities proportional to body volume
3. Uniformly sample point in body
4. Uniformly sample another body
   If sampled point in this body, Success! Repeat Step 2, Else repeat Step 4.

ApproxUnion: Volume of Union of Convex Bodies [5]

Consider *k* convex bodies. We wish to compute the volume of their union.



1 Compute individual body volumes

Run until Step 4 executes T times:

2 Randomly sample body
sampling probabilities proportional to body volume

3 Uniformly sample point in body

4 Uniformly sample another body
If sampled point in this body, Success!
Repeat Step 2, Else repeat Step 4.

Number of successes then yields an unbiased estimator for volume

ApproxUnion is an FPRAS

1. Volume Computation: FPRAS with error ϵ_V, confidence δ_V

2. Point Sampling: FPRAS with error ϵ_S, confidence δ_S

3. Membership Check: FPRAS with error ϵ_P, confidence δ_P

ApproxUnion is an FPRAS

1. Volume Computation: FPRAS with error ϵ_V, confidence δ_V

2. Point Sampling: FPRAS with error ϵ_S, confidence δ_S

3. Membership Check: FPRAS with error ϵ_P, confidence δ_P

ApproxUnion is an FPRAS with error ε and confidence δ, using $\mathrm{T} = O(k\varepsilon^{-2})$,
for $\varepsilon_V, \varepsilon_S \leq \frac{\varepsilon^2}{47k}, \varepsilon_P \leq \frac{\varepsilon^2}{47k^2}, \delta_V \leq \frac{\delta}{4k}, \delta_S + \delta_P \leq \frac{\delta}{2276 \ln(\frac{8}{\delta})\frac{k}{\varepsilon^2}}$.

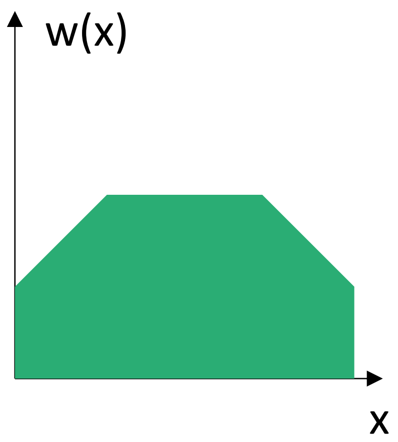
ApproxWMI: An FPRAS for WMI(DNF)

ApproxWMI: An FPRAS for WMI(DNF)

- ApproxWMI applies over WMI(DNF) with concave weight functions, e.g.,:

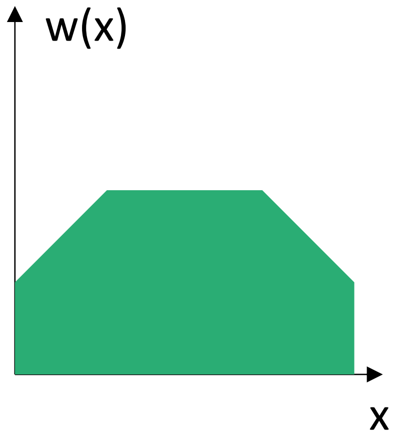
ApproxWMI: An FPRAS for WMI(DNF)

- ApproxWMI applies over WMI(DNF) with concave weight functions, e.g.,:



ApproxWMI: An FPRAS for WMI(DNF)

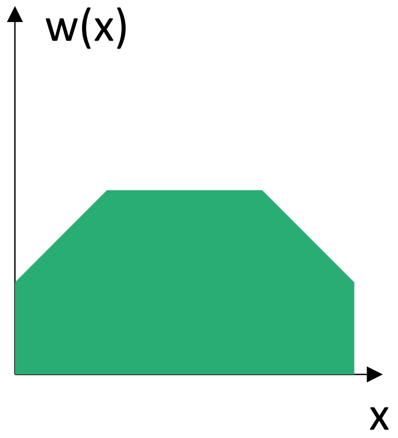
- ApproxWMI applies over WMI(DNF) with concave weight functions, e.g.,:



- ApproxWMI:

ApproxWMI: An FPRAS for WMI(DNF)

- ApproxWMI applies over WMI(DNF) with concave weight functions, e.g.,:

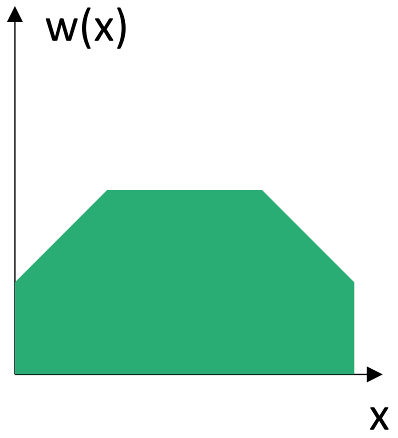


- ApproxWMI:

  1) Computes the weight of every DNF clause: weight function integral over LRA-induced convex polytope, multiplied by Boolean probabilities.

ApproxWMI: An FPRAS for WMI(DNF)

- ApproxWMI applies over WMI(DNF) with concave weight functions, e.g.,:



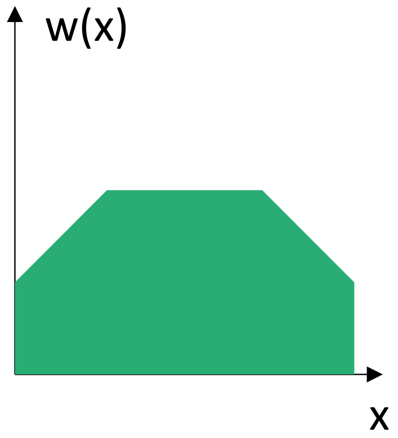
- ApproxWMI:

  1) Computes the weight of every DNF clause: weight function integral over LRA-induced convex polytope, multiplied by Boolean probabilities.

  2) Samples points from a clause as per the weight function.

ApproxWMI: An FPRAS for WMI(DNF)

- ApproxWMI applies over WMI(DNF) with concave weight functions, e.g.,:



- ApproxWMI:

  1) Computes the weight of every DNF clause: weight function integral over LRA-induced convex polytope, multiplied by Boolean probabilities.

  2) Samples points from a clause as per the weight function.

  3) Checks membership of point to another uniformly random clause.

ApproxWMI: Computing Clause Weight

ApproxWMI: Computing Clause Weight

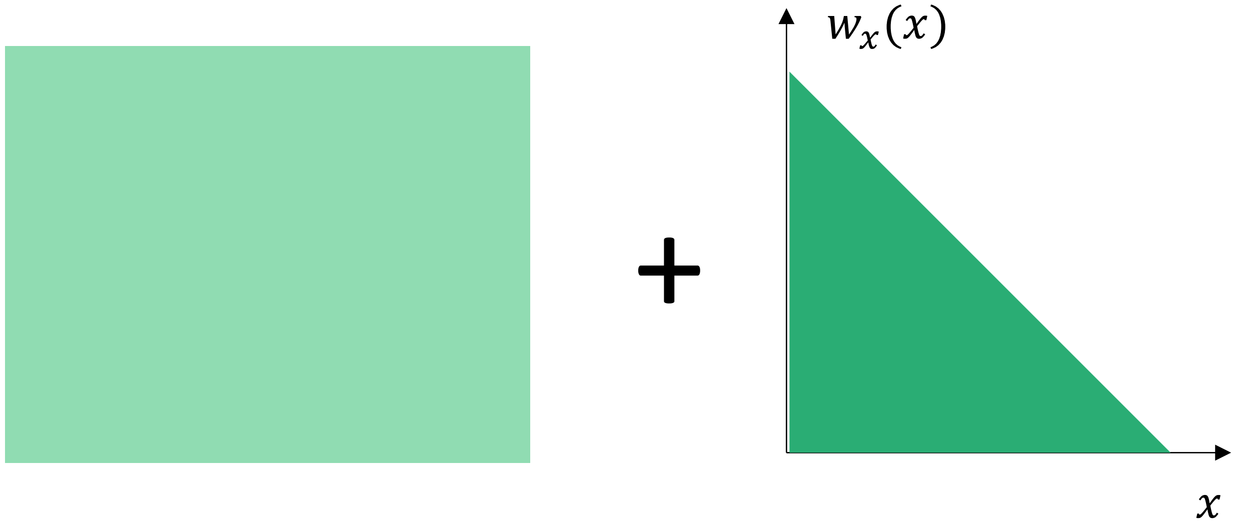
- In a DNF clause, LRA atoms define a convex polytope.

ApproxWMI: Computing Clause Weight

- In a DNF clause, LRA atoms define a convex polytope.
- The integral of $w_x(x)$ over this polytope, multiplied by the probabilities of clause Boolean literals, yields the clause weight.

ApproxWMI: Computing Clause Weight

- In a DNF clause, LRA atoms define a convex polytope.
- The integral of $w_x(x)$ over this polytope, multiplied by the probabilities of clause Boolean literals, yields the clause weight.
- Since $w_x(x)$ is concave, the integral can be computed as the volume of a combined *n+1*-dimensional convex polytope, e.g.,:

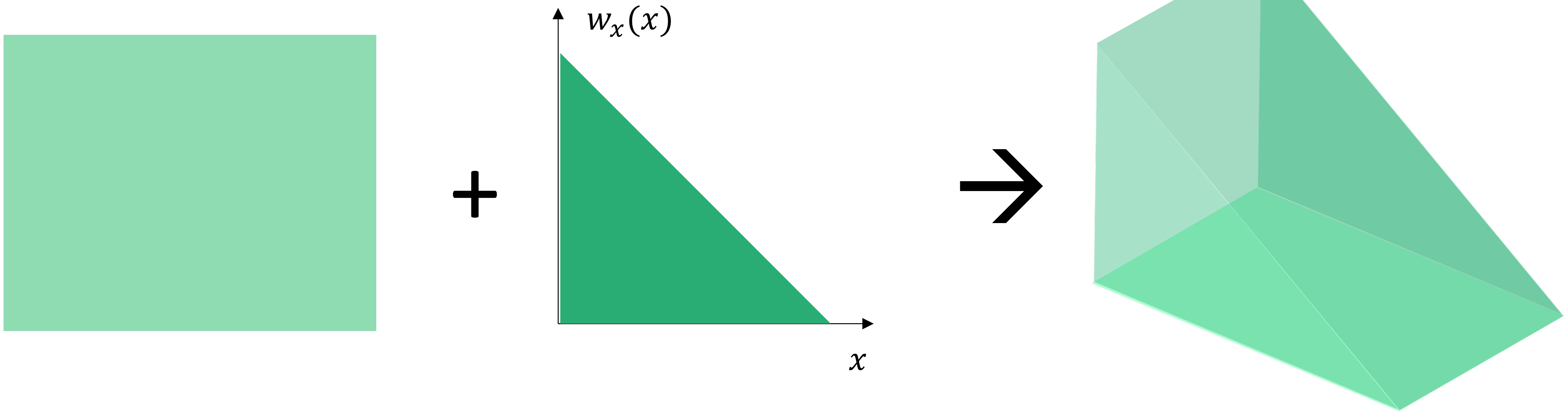


2D Convex Polytope

Concave Weight Function

ApproxWMI: Computing Clause Weight

- In a DNF clause, LRA atoms define a convex polytope.
- The integral of $w_x(x)$ over this polytope, multiplied by the probabilities of clause Boolean literals, yields the clause weight.
- Since $w_x(x)$ is concave, the integral can be computed as the volume of a combined *n+1*-dimensional convex polytope, e.g.,:

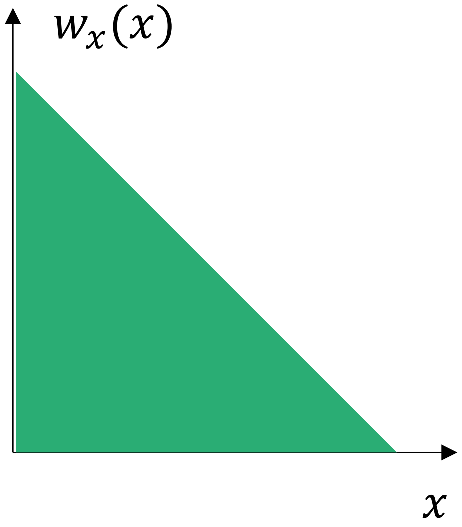


2D Convex Polytope | Concave Weight Function | 3D Convex Polytope

ApproxWMI: Computing Clause Weight

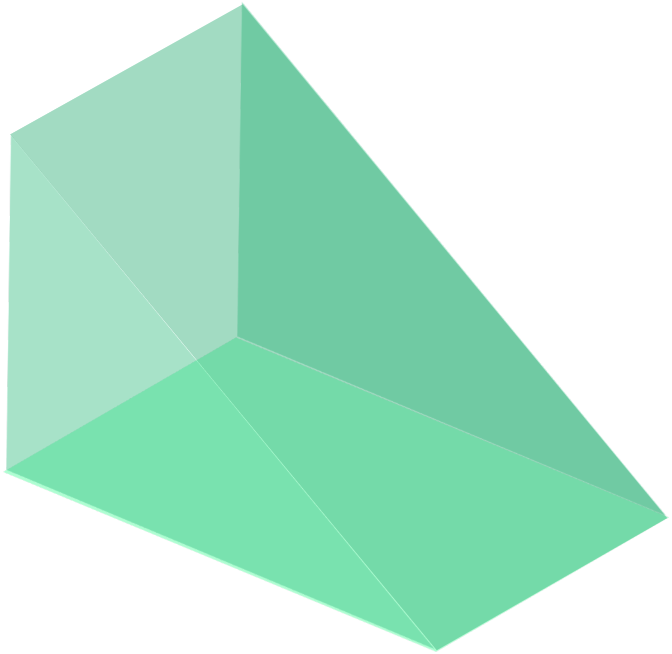
2D Convex Polytope

+



Concave Weight Function

→

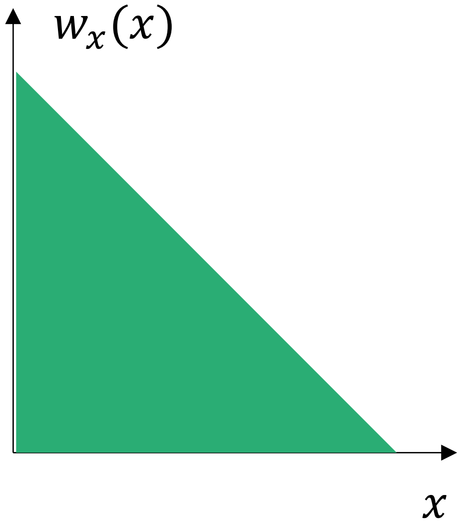


3D Convex Polytope

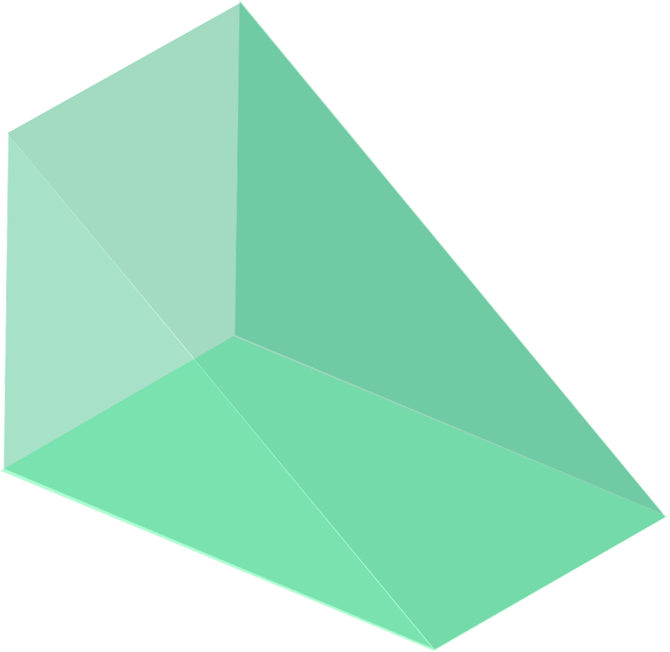
ApproxWMI: Computing Clause Weight

- The volume of this polytope is then computed using standard tools [6].

2D Convex Polytope



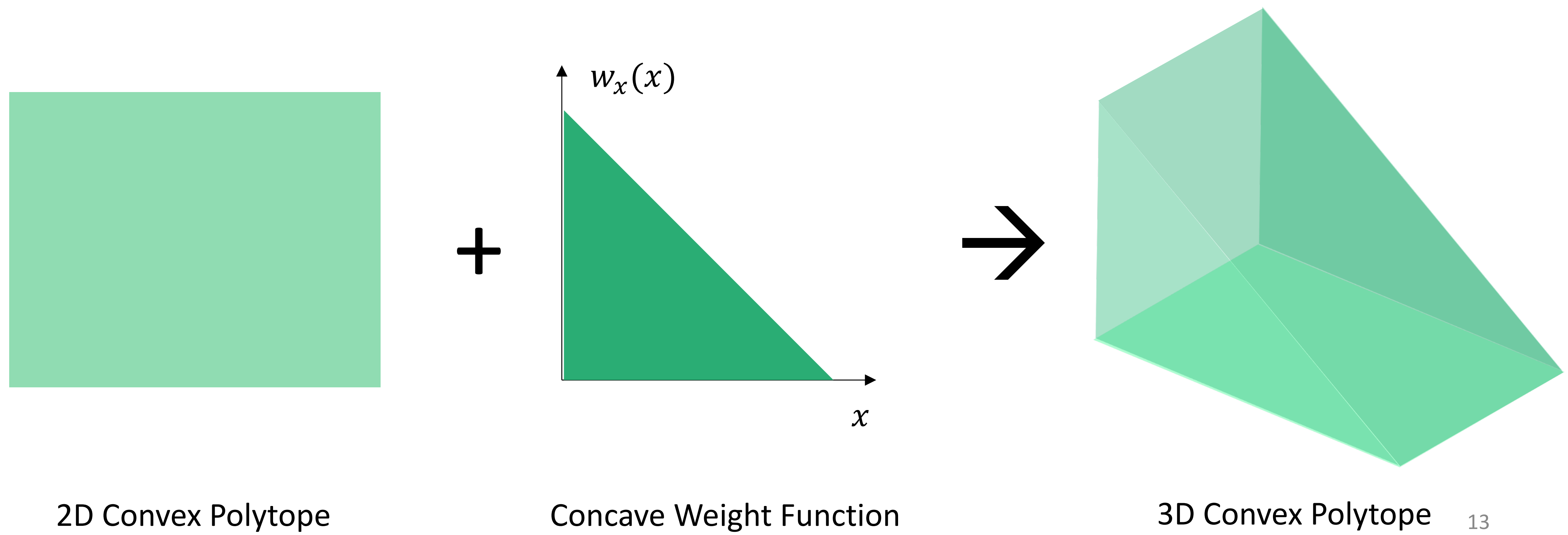
Concave Weight Function



3D Convex Polytope

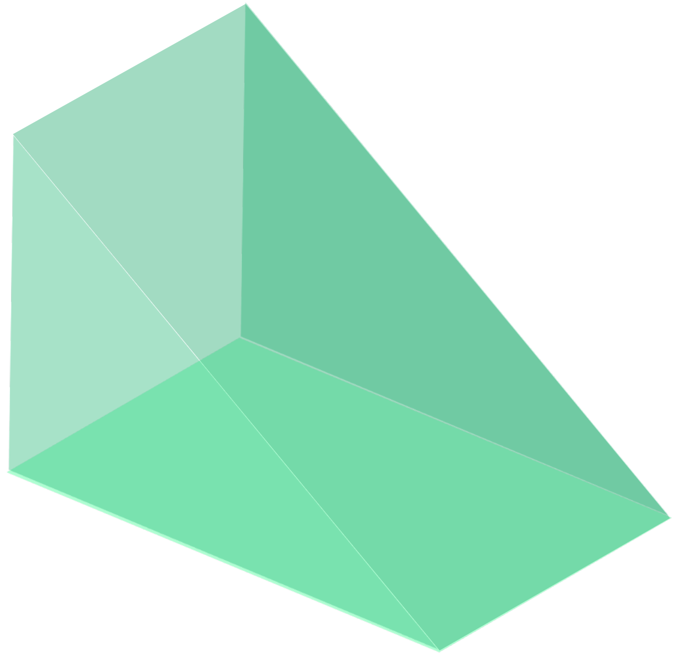
ApproxWMI: Computing Clause Weight

- The volume of this polytope is then computed using standard tools [6].
- This volume is multiplied by Boolean probabilities to return clause weight.



ApproxWMI: Sampling Points and Checking Membership

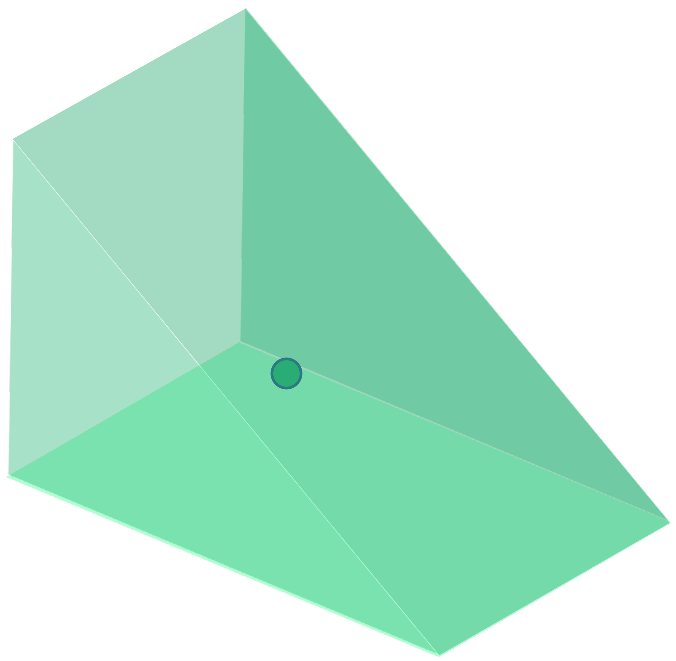
- The *n+1*-dimensional polytope is sampled uniformly, as a proxy for sampling the original polytope according to the weight function.



3D Convex Polytope

ApproxWMI: Sampling Points and Checking Membership

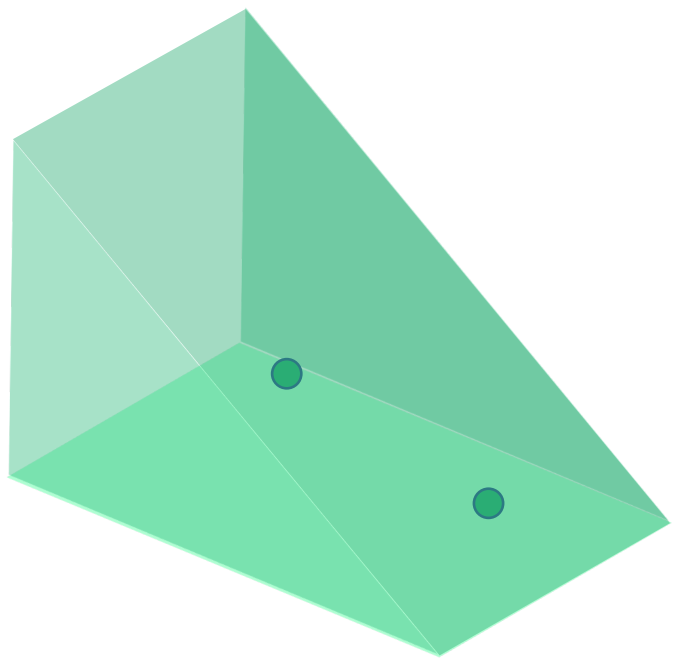
- The *n+1*-dimensional polytope is sampled uniformly, as a proxy for sampling the original polytope according to the weight function.



3D Convex Polytope

ApproxWMI: Sampling Points and Checking Membership

- The *n+1*-dimensional polytope is sampled uniformly, as a proxy for sampling the original polytope according to the weight function.

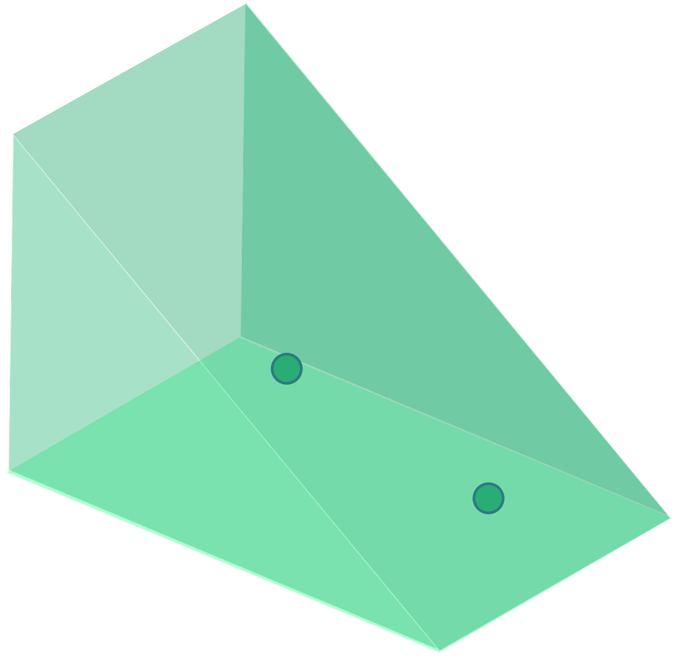


3D Convex Polytope

ApproxWMI: Sampling Points and Checking Membership

- The *n+1*-dimensional polytope is sampled uniformly, as a proxy for sampling the original polytope according to the weight function.

- Membership to LRA polytope is checked by validating point against every LRA atom in a clause.



3D Convex Polytope

ApproxWMI is an FPRAS for WMI(DNF)

1. **ClauseWeight:** FPRAS with error ϵ_V, confidence δ_V
2. **Sampling**: FPRAS with error ϵ_S, confidence δ_S
3. **Evaluate**: FPRAS with error ϵ_P, confidence δ_P

ApproxWMI is an FPRAS for WMI(DNF)

1. **ClauseWeight:** FPRAS with error ϵ_V, confidence δ_V
2. **Sampling**: FPRAS with error ϵ_S, confidence δ_S
3. **Evaluate**: FPRAS with error ϵ_P, confidence δ_P

ApproxWMI is an FPRAS for WMI(DNF) with error ε and confidence δ, using

$\mathrm{T} = O(k\varepsilon^{-2})$, for $\varepsilon_V, \varepsilon_S \leq \frac{\varepsilon^2}{47k}, \varepsilon_P \leq \frac{\varepsilon^2}{47k^2}, \delta_V \leq \frac{\delta}{4k}, \delta_S + \delta_P \leq \frac{\delta}{2276 \ln(\frac{8}{\delta})\frac{k}{\varepsilon^2}}$.

Experimental Setup: Data Generation

Experimental Setup: Data Generation

- Randomly generated DNF instances, with equally many Boolean and real variables. Total number of variables uniformly set between 100 and 1000 in increments of 100.

Experimental Setup: Data Generation

- Randomly generated DNF instances, with equally many Boolean and real variables. Total number of variables uniformly set between 100 and 1000 in increments of 100.

- Clause width uniformly set between 3,5,8, and 13.

Experimental Setup: Data Generation

- Randomly generated DNF instances, with equally many Boolean and real variables. Total number of variables uniformly set between 100 and 1000 in increments of 100.

- Clause width uniformly set between 3,5,8, and 13.

- Number of clauses is roughly the number of variables divided by clause width.

Experimental Setup: Data Generation

- Randomly generated DNF instances, with equally many Boolean and real variables. Total number of variables uniformly set between 100 and 1000 in increments of 100.

- Clause width uniformly set between 3,5,8, and 13.

- Number of clauses is roughly the number of variables divided by clause width.

- Real weight function is a concave polynomial with degree up to 5.

Experimental Setup: ApproxWMI

Experimental Setup: ApproxWMI

- Exact weight computation oracle: Faster due to small clause widths.

Experimental Setup: ApproxWMI

- Exact weight computation oracle: Faster due to small clause widths.
- Hit-and-Run sampler [7], with practically used constant iteration factor.

Experimental Setup: ApproxWMI

- Exact weight computation oracle: Faster due to small clause widths.
- Hit-and-Run sampler [7], with practically used constant iteration factor.

Three algorithm configurations:

Experimental Setup: ApproxWMI

- Exact weight computation oracle: Faster due to small clause widths.
- Hit-and-Run sampler [7], with practically used constant iteration factor.

Three algorithm configurations:

- Target error ϵ: Set to 0.15, 0.25, and 0.35.

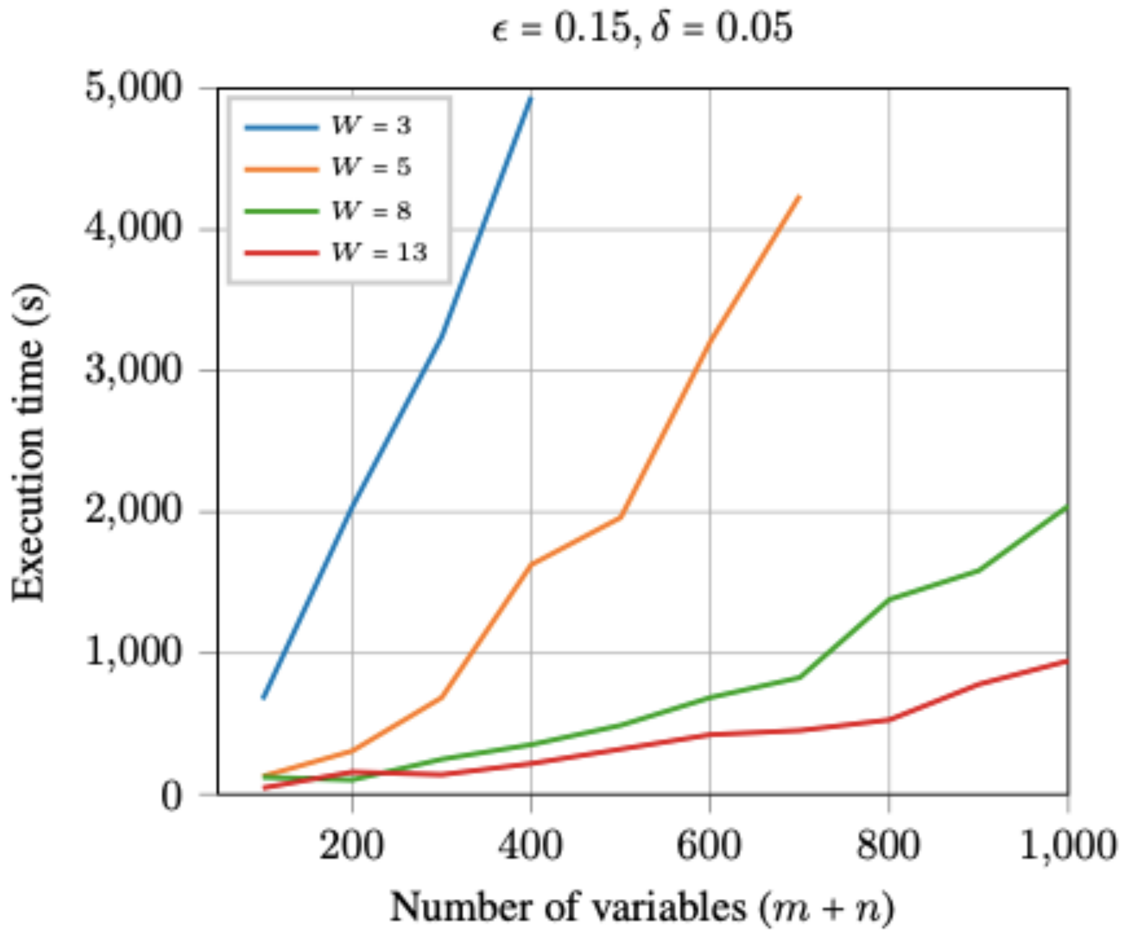
Experimental Setup: ApproxWMI

- Exact weight computation oracle: Faster due to small clause widths.
- Hit-and-Run sampler [7], with practically used constant iteration factor.

Three algorithm configurations:

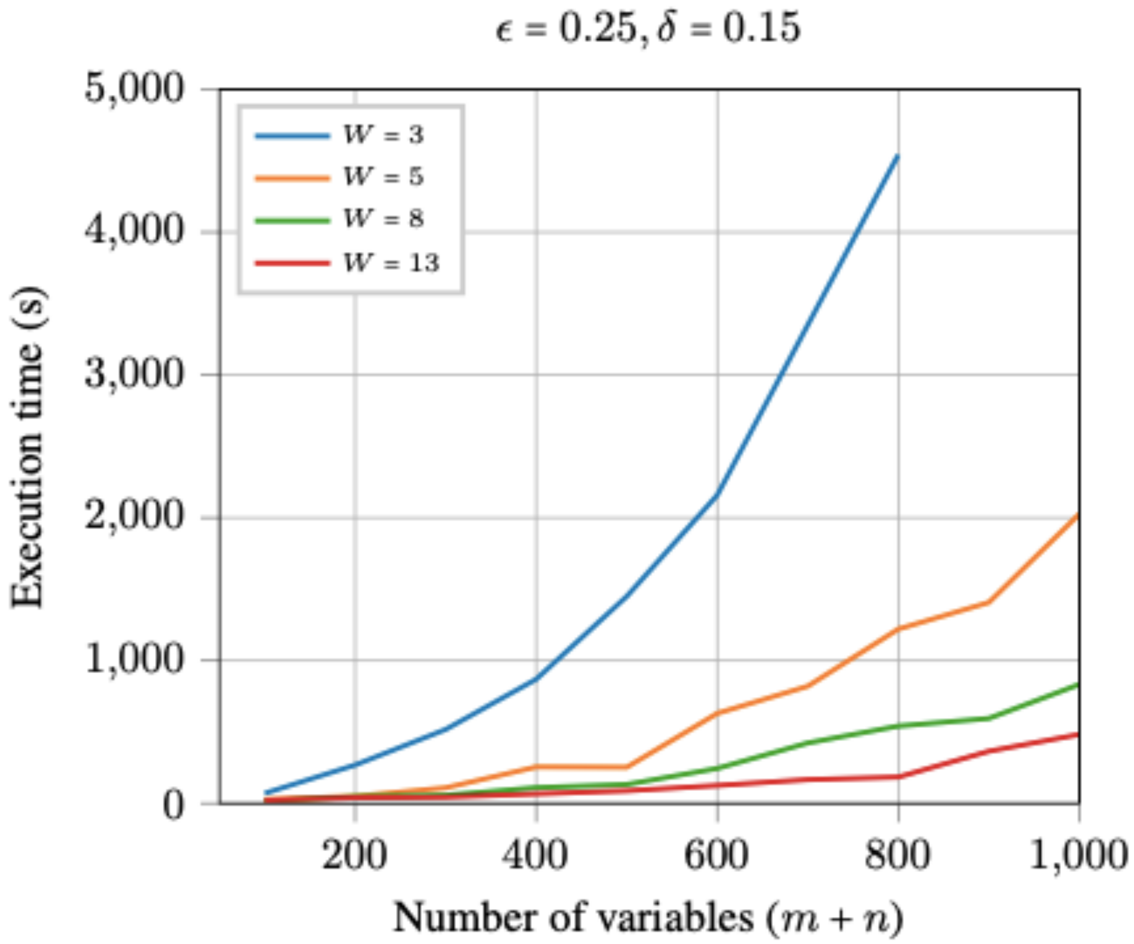
- Target error ϵ: Set to 0.15, 0.25, and 0.35.
- Target confidence δ: Set to 0.05, 0.15 and 0.25.

Results

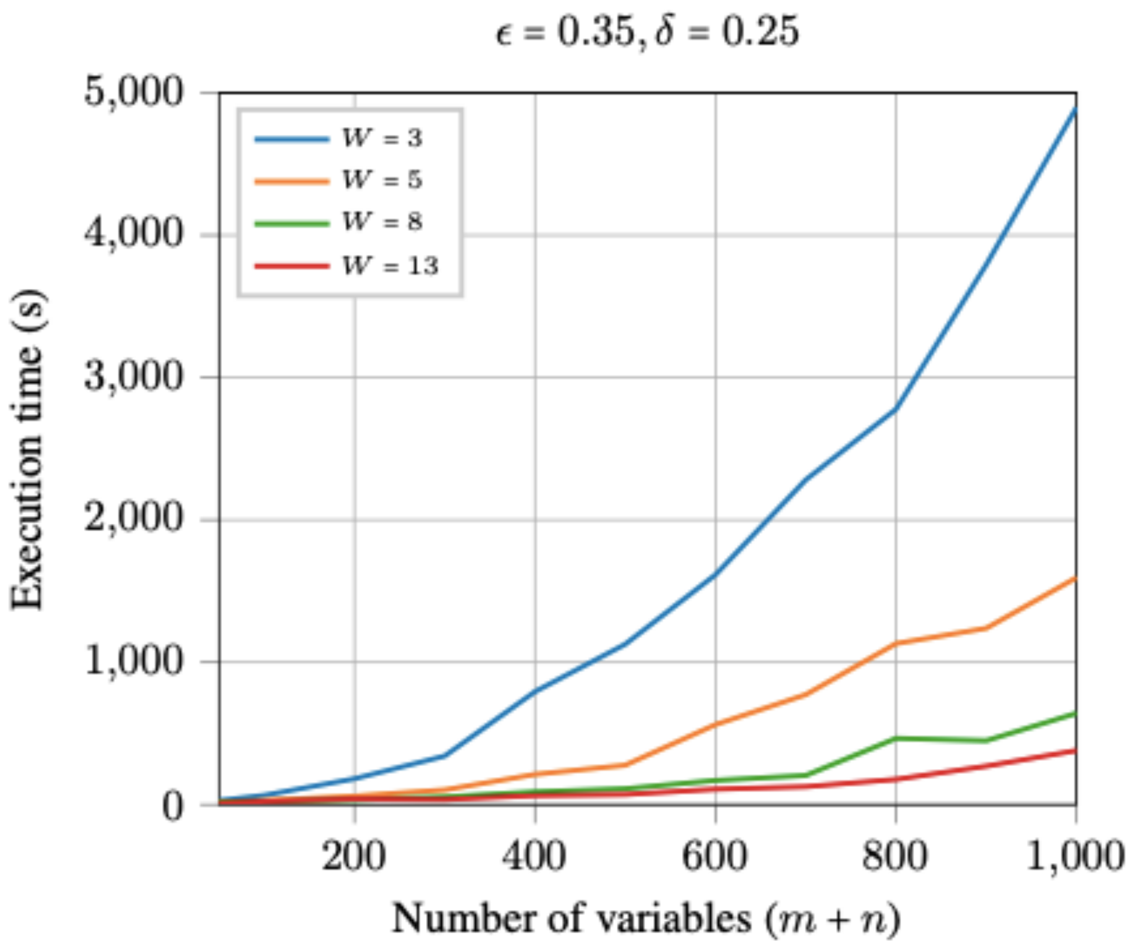


$\epsilon = 0.15, \delta = 0.05$
5,000
4,000
3,000
2,000
1,000
0
Execution time (s)
$W = 3$
$W = 5$
$W = 8$
$W = 13$
200
400
600
800
1,000
Number of variables $(m + n)$

Results


$\epsilon = 0.25, \delta = 0.15$
W = 3
W = 5
W = 8
W = 13
5,000
4,000
3,000
2,000
1,000
0
Execution time (s)
200
400
600
800
1,000
Number of variables (m + n)

Results


$\epsilon = 0.35, \delta = 0.25$
5,000
4,000
3,000
2,000
1,000
0
Execution time (s)
$W = 3$
$W = 5$
$W = 8$
$W = 13$
200
400
600
800
1,000
Number of variables $(m + n)$

Summary

Summary

- ApproxWMI is an FPRAS for WMI(DNF) given concave weight functions.

Summary

- ApproxWMI is an FPRAS for WMI(DNF) given concave weight functions.
- ApproxWMI is scalable, and can efficiently solve instances with up to 1K variables, which is out of reach for existing WMI solvers.

Summary

- ApproxWMI is an FPRAS for WMI(DNF) given concave weight functions.
- ApproxWMI is scalable, and can efficiently solve instances with up to 1K variables, which is out of reach for existing WMI solvers.
- ApproxWMI can be extended to more general factorizations enabling real-Boolean dependency.

Summary

- ApproxWMI is an FPRAS for WMI(DNF) given concave weight functions.
- ApproxWMI is scalable, and can efficiently solve instances with up to 1K variables, which is out of reach for existing WMI solvers.
- ApproxWMI can be extended to more general factorizations enabling real-Boolean dependency.
- ApproxWMI is a useful tool for efficient probabilistic inference in hybrid domains.

Thank You!

UNIVERSITY OF OXFORD

Selected References

[1] Abboud, R.; Ceylan, İ; and Dimitrov, R. 2020. On the approximability of weighted model integration on DNF structures. To appear in *Proc. of KR.*

[2] Martires, P.; Dries, A.; and De Raedt, L. 2019. Exact and approximate weighted model integration with probability density functions using knowledge compilation. In *Proc. of AAAI*, 7825–7833

[3] Roth, D. 1996. On the Hardness of Approximate Reasoning. *AIJ* 82(1-2):273–302

[4] Karp, R. M.; Luby, M.; and Madras, N. 1989. Monte-Carlo approximation algorithms for enumeration problems. *J. Algorithms* 10(3):429–448

[5] Bringmann, K., and Friedrich, T. 2010. Approximating the volume of unions and intersections of high-dimensional geometric objects. *Comput. Geom.* 43(6-7):601–610.

[6] Lovasz, L., and Vempala, S. S. 2006. Simulated annealing in convex bodies and an $O^*(n^4)$ volume algorithm. *JCSS* 72(2):392–417.

[7] Chen, M., and Schmeiser, B. W. 1996. General hit-and-run Monte Carlo sampling for evaluating multidimensional integrals. *Oper. Res. Lett.* 19(4):161–169

UNIVERSITY OF OXFORD