Cara: An Isomorphism-Based #SAT Solver

This work is partially published in AAAI 2025

Petr Illner

Department of Theoretical Computer Science and Mathematical Logic, Faculty of Mathematics and Physics
Charles University, Czech Republic

illner3@gmail.com

MODEL COUNTING COMPETITION 2025

Component caching

To avoid redundant work

Input CNF formula

φ: $(\mathbf{x_1} \lor x_2 \lor x_3) \land (\neg\mathbf{x_1} \lor x_2 \lor x_3)$

Component caching

To avoid redundant work

Input CNF formula

φ: $(x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor x_2 \lor x_3)$

$\varphi \mid x_1$

$(x_2 \lor x_3)$

Component caching

To avoid redundant work

Input CNF formula

$\varphi: (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$

$\varphi \mid x_1$

$(x_2 \vee x_3)$

…

$\#(\varphi \mid x_1) = 3$

Component caching

To avoid redundant work

Input CNF formula

$\varphi: (x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor x_2 \lor x_3)$

$\varphi \mid x_1$

$(x_2 \lor x_3)$

…

$\#(\varphi \mid x_1) = 3$

$\varphi \mid \neg x_1$

$(x_2 \lor x_3)$

Component caching

To avoid redundant work

Input CNF formula

$\varphi: (x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor x_2 \lor x_3)$

$\varphi \mid x_1$

$(x_2 \lor x_3)$

…

$\#(\varphi \mid x_1) = \mathbf{3} = \#(\varphi \mid \neg x_1)$

$\varphi \mid \neg x_1$

$(x_2 \lor x_3)$

…

Component caching

To avoid redundant work

Input CNF formula

φ: $(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$

$\varphi \mid x_1$

$(x_2 \vee x_3)$

$\#(\varphi \mid x_1) = \mathbf{3} = \#(\varphi \mid \neg x_1)$

$\varphi \mid \neg x_1$

$(x_2 \vee x_3)$

…

notation: **residual CNF formulae**

…

Caching scheme

Caching scheme

A caching scheme is **correct**[1] if $\forall \psi_1, \psi_2 \colon (r(\psi_1) = r(\psi_2)) \Rightarrow (\psi_1 \Leftrightarrow \psi_2)$

[1] LAGNIEZ, Jean-Marie; MARQUIS, Pierre. Enhanced Caching for# SAT Solving. 2020.

Caching scheme

residual formula ψ → **caching scheme** → representation **r(ψ)** (typically, a string)

A caching scheme is **correct**[1] if $\forall \psi_1, \psi_2 \colon (r(\psi_1) = r(\psi_2)) \Rightarrow \cancel{(\psi_1 \Leftrightarrow \psi_2)}\ (\#\psi_1 = \#\psi_2)$

[1] LAGNIEZ, Jean-Marie; MARQUIS, Pierre. Enhanced Caching for# SAT Solving. 2020.

Caching scheme

residual formula ψ → **caching scheme** → representation $\mathbf{r(\psi)}$ (typically, a string)

A caching scheme is **correct**[1] if $\forall \psi_1, \psi_2 \colon (r(\psi_1) = r(\psi_2)) \Rightarrow \cancel{(\psi_1 \Leftrightarrow \psi_2)}\ (\#\psi_1 = \#\psi_2)$

Key properties of a caching scheme: strength of equivalence detection, representation size, computation time

[1] LAGNIEZ, Jean-Marie; MARQUIS, Pierre. Enhanced Caching for# SAT Solving. 2020.

Caching scheme

residual formula ψ → **caching scheme** → representation $\mathbf{r(\psi)}$ (typically, a string)

A caching scheme is **correct**[1] if $\forall \psi_1, \psi_2 : (r(\psi_1) = r(\psi_2)) \Rightarrow \cancel{(\psi_1 \Leftrightarrow \psi_2)}\ (\#\psi_1 = \#\psi_2)$

Key properties of a caching scheme: strength of equivalence detection,
representation size, computation time

Explicit representation: *standard* (*Cachet*[2]), *basic*, *i* (*D4*[3]),

Implicit representation: *hybrid* (*sharpSAT*[4]), *o*, *i'*,
probabilistic component caching (*GANAK*[5])

[1] LAGNIEZ, Jean-Marie; MARQUIS, Pierre. Enhanced Caching for# SAT Solving. 2020.
[2] SANG, Tian, et al. Combining Component Caching and Clause Learning for Effective Model Counting. SAT, 2004, 4: 7th.
[3] LAGNIEZ, Jean-Marie; MARQUIS, Pierre. An Improved Decision-DNNF Compiler. In: IJCAI. 2017. p. 667-673.
[4] THURLEY, Marc. sharpSAT–counting models with advanced component caching and implicit BCP. In: International Conference on Theory and Applications of Satisfiability Testing. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 424-429.
[5] SHARMA, Shubham, et al. GANAK: A Scalable Probabilistic Exact Model Counter. In: IJCAI. 2019. p. 1169-1176.

Caching scheme i[1] (presented at the Workshop on Counting and Sampling 2021)

$$\varphi : (x_1 \lor \mathbf{x_2} \lor \neg x_4) \land (\mathbf{x_2} \lor x_3 \lor x_4) \land (x_1 \lor \mathbf{x_2} \lor \neg x_4) \land (x_3 \lor x_4 \lor x_5) \land (x_1 \lor \neg x_6)$$

$$\varphi \mid \neg x_2 : \quad (x_1 \lor \neg x_4) \quad \land (x_3 \lor x_4) \land (x_1 \lor \neg x_4) \land (x_3 \lor x_4 \lor x_5) \land (x_1 \lor \neg x_6)$$

[1] LAGNIEZ, Jean-Marie; MARQUIS, Pierre. Enhanced Caching for# SAT Solving. 2020.

Caching scheme i[1] (presented at the Workshop on Counting and Sampling 2021)

$\varphi : (x_1 \lor \mathbf{x_2} \lor \neg x_4) \land (\mathbf{x_2} \lor x_3 \lor x_4) \land (x_1 \lor \mathbf{x_2} \lor \neg x_4) \land (x_3 \lor x_4 \lor x_5) \land (x_1 \lor \neg x_6)$

$\varphi \mid \neg x_2 : \quad (x_1 \lor \neg x_4) \quad \land (x_3 \lor x_4) \land (x_1 \lor \neg x_4) \land \underbrace{(x_3 \lor x_4 \lor x_5) \land (x_1 \lor \neg x_6)}_{\textbf{untouched clauses}}$

[1] LAGNIEZ, Jean-Marie; MARQUIS, Pierre. Enhanced Caching for# SAT Solving. 2020.

Caching scheme i[1] (presented at the Workshop on Counting and Sampling 2021)

$\varphi : (x_1 \lor \mathbf{x_2} \lor \neg x_4) \land (\mathbf{x_2} \lor x_3 \lor x_4) \land (x_1 \lor \mathbf{x_2} \lor \neg x_4) \land (x_3 \lor x_4 \lor x_5) \land (x_1 \lor \neg x_6)$

$\varphi \mid \neg x_2 : \quad (x_1 \lor \neg x_4) \quad \land (x_3 \lor x_4) \land (x_1 \lor \neg x_4) \land \underbrace{(x_3 \lor x_4 \lor x_5) \land (x_1 \lor \neg x_6)}_{\text{untouched clauses}}$

$r(\varphi \mid \neg x_2) = \underbrace{\qquad\qquad}_{\text{sorted residual variables}} \; \underbrace{\qquad\qquad}_{\text{sorted touched residual clauses}}$

[1] LAGNIEZ, Jean-Marie; MARQUIS, Pierre. Enhanced Caching for# SAT Solving. 2020.

Caching scheme i[1] (presented at the Workshop on Counting and Sampling 2021)

$\varphi : (x_1 \lor \mathbf{x_2} \lor \neg x_4) \land (\mathbf{x_2} \lor x_3 \lor x_4) \land (x_1 \lor \mathbf{x_2} \lor \neg x_4) \land (x_3 \lor x_4 \lor x_5) \land (x_1 \lor \neg x_6)$

$\varphi \mid \neg x_2 : \quad (x_1 \lor \neg x_4) \quad \land (x_3 \lor x_4) \land (x_1 \lor \neg x_4) \land \underbrace{(x_3 \lor x_4 \lor x_5) \land (x_1 \lor \neg x_6)}_{\textbf{untouched clauses}}$

$r(\varphi \mid \neg x_2) = \underbrace{1, 3, 4, 5, 6, 0,}_{\textbf{sorted residual variables}} \; \underbrace{1, -4, 0, 1, -4, 0, 3, 4, 0}_{\textbf{sorted touched residual clauses}}$

[1] LAGNIEZ, Jean-Marie; MARQUIS, Pierre. Enhanced Caching for# SAT Solving. 2020.

Caching scheme i[1] (presented at the Workshop on Counting and Sampling 2021)

$$\varphi : (x_1 \vee \mathbf{x_2} \vee \neg x_4) \wedge (\mathbf{x_2} \vee x_3 \vee x_4) \wedge (x_1 \vee \mathbf{x_2} \vee \neg x_4) \wedge (x_3 \vee x_4 \vee x_5) \wedge (x_1 \vee \neg x_6)$$

$$\varphi \mid \neg x_2 : \underbrace{(x_1 \vee \neg x_4)}_{\text{redundant clause}} \wedge (x_3 \vee x_4) \wedge (x_1 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4 \vee x_5) \wedge (x_1 \vee \neg x_6)}_{\text{untouched clauses}}$$

$$r(\varphi \mid \neg x_2) = \underbrace{1, 3, 4, 5, 6, 0,}_{\text{sorted residual variables}} \underbrace{\cancel{1, -4, 0,}\ 1, -4, 0, 3, 4, 0}_{\text{sorted touched non-redundant residual clauses}}$$

[1] LAGNIEZ, Jean-Marie; MARQUIS, Pierre. Enhanced Caching for# SAT Solving. 2020.

Motivation

Residual CNF formulae:

$$\psi_1 : (x_1 \lor x_2) \land (\neg x_1 \lor x_3) \land (x_3 \lor \neg x_4) \land (x_1 \lor \neg x_3 \lor x_4)$$

Motivation

Residual CNF formulae:

$\psi_1 : (x_1 \vee x_2) \wedge (\neg x_1 \vee x_3) \wedge (x_3 \vee \neg x_4) \wedge (x_1 \vee \neg x_3 \vee x_4)$ $\#\psi_1 = \mathbf{6}$

Motivation

Residual CNF formulae:

$\psi_1 : (x_1 \lor x_2) \land (\neg x_1 \lor x_3) \land (x_3 \lor \neg x_4) \land (x_1 \lor \neg x_3 \lor x_4)$ $\#\psi_1 = \mathbf{6}$

$\psi_2 : (x_5 \lor x_6) \land (\neg x_5 \lor x_7) \land (x_7 \lor \neg x_8) \land (x_5 \lor \neg x_7 \lor x_8)$

Motivation

Residual CNF formulae:

$\psi_1 : (x_1 \lor x_2) \land (\neg x_1 \lor x_3) \land (x_3 \lor \neg x_4) \land (x_1 \lor \neg x_3 \lor x_4)$ $\#\psi_1 = \mathbf{6}$

$\psi_2 : (x_5 \lor x_6) \land (\neg x_5 \lor x_7) \land (x_7 \lor \neg x_8) \land (x_5 \lor \neg x_7 \lor x_8)$ $\#\psi_2 = \mathbf{6}$

variable names are NOT relevant

Motivation

Residual CNF formulae:

$\psi_1 : (x_1 \lor x_2) \land (\neg x_1 \lor x_3) \land (x_3 \lor \neg x_4) \land (x_1 \lor \neg x_3 \lor x_4)$ $\#\psi_1 = \mathbf{6}$

$\psi_2 : (x_5 \lor x_6) \land (\neg x_5 \lor x_7) \land (x_7 \lor \neg x_8) \land (x_5 \lor \neg x_7 \lor x_8)$ $\#\psi_2 = \mathbf{6}$

variable names are NOT relevant

$\psi_3 : (\neg x_1 \lor \neg x_2) \land (x_1 \lor \neg x_3) \land (\neg x_3 \lor x_4) \land (\neg x_1 \lor x_3 \lor \neg x_4)$

Motivation

Residual CNF formulae:

$\psi_1 : (x_1 \vee x_2) \wedge (\neg x_1 \vee x_3) \wedge (x_3 \vee \neg x_4) \wedge (x_1 \vee \neg x_3 \vee x_4)$ $\quad \#\psi_1 = 6$

$\psi_2 : (x_5 \vee x_6) \wedge (\neg x_5 \vee x_7) \wedge (x_7 \vee \neg x_8) \wedge (x_5 \vee \neg x_7 \vee x_8)$ $\quad \#\psi_2 = 6$

variable names are NOT relevant

$\psi_3 : (\neg x_1 \vee \neg x_2) \wedge (x_1 \vee \neg x_3) \wedge (\neg x_3 \vee x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4)$ $\quad \#\psi_3 = 6$

literal polarity is NOT relevant

Isomorphism-aware caching scheme

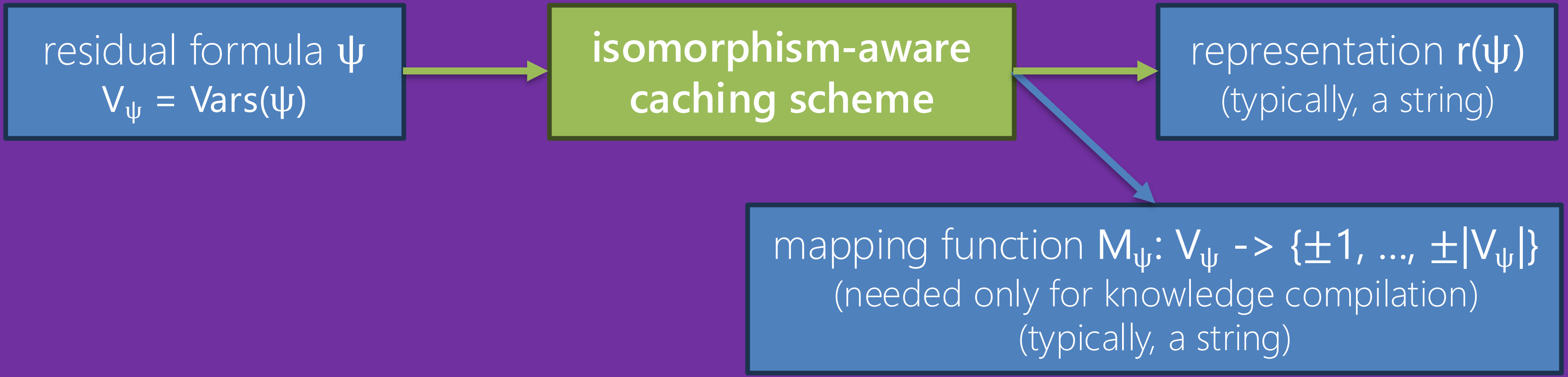
residual formula ψ
V_ψ = Vars(ψ)

isomorphism-aware caching scheme

representation **r(ψ)**
(typically, a string)

mapping function **M_ψ**: V_ψ -> $\{\pm 1, \ldots, \pm|V_\psi|\}$
(needed only for knowledge compilation)
(typically, a string)

Isomorphism-aware caching scheme



An isomorphism-aware caching scheme is **correct** if :

$$\forall \psi_1, \psi_2 : (\mathrm{r}(\psi_1) = \mathrm{r}(\psi_2)) \Rightarrow (M_{\psi 1}(\psi_1) \Leftrightarrow M_{\psi 2}(\psi_2)) \equiv (\#\psi_1 = \#\psi_2)$$

Cara caching scheme[1]

$\psi_1 = (\neg x_3 \lor \neg x_5) \land (x_4 \lor \neg x_6) \land (x_3 \lor x_5 \lor x_6) \land (\neg x_4 \lor x_5 \lor \neg x_6)$

$\#\psi_1 = ?$

$V_{\psi 1} = \{ 3, 4, 5, 6 \}$

[1] ILLNER, Petr. New Compilation Languages Based on Restricted Weak Decomposability. In: Proceedings of the AAAI Conference on Artificial Intelligence. 2025. p. 14987-14996.

Cara caching scheme[1]

$\psi_1 = (\neg x_3 \lor \neg x_5) \land (x_4 \lor \neg x_6) \land (x_3 \lor x_5 \lor x_6) \land (\neg x_4 \lor x_5 \lor \neg x_6)$

$\#\psi_1 = ?$

$V_{\psi 1} = \{ 3, 4, 5, 6 \}$

	C_x	$C_{\neg x}$	μ_x	$\mu_{\neg x}$	ID
x_3					
x_4					
x_5					
x_6					

C_l = the number of clauses containing l

μ_l = the average size of clauses containing l

$C_x < C_{\neg x}$ = sign flip

[1] ILLNER, Petr. New Compilation Languages Based on Restricted Weak Decomposability. In: Proceedings of the AAAI Conference on Artificial Intelligence. 2025. p. 14987-14996.

Cara caching scheme[1]

$\psi_1 = (\neg x_3 \lor \neg x_5) \land (x_4 \lor \neg x_6) \land (x_3 \lor x_5 \lor x_6) \land (\neg x_4 \lor x_5 \lor \neg x_6)$

$\#\psi_1 = ?$

$V_{\psi 1} = \{ 3, 4, 5, 6 \}$

	C_x	$C_{\neg x}$	μ_x	$\mu_{\neg x}$	ID
x_3					3
x_4					
x_5					
x_6					

C_l = the number of clauses containing l

μ_l = the average size of clauses containing l

$C_x < C_{\neg x}$ = sign flip

[1] ILLNER, Petr. New Compilation Languages Based on Restricted Weak Decomposability. In: Proceedings of the AAAI Conference on Artificial Intelligence. 2025. p. 14987-14996.

Cara caching scheme[1]

$\psi_1 = (\neg x_3 \lor \neg x_5) \land (x_4 \lor \neg x_6) \land (\underline{\mathbf{x_3}} \lor x_5 \lor x_6) \land (\neg x_4 \lor x_5 \lor \neg x_6)$

$\#\psi_1 = ?$

$V_{\psi 1} = \{ 3, 4, 5, 6 \}$

	C_x	$C_{\neg x}$	μ_x	$\mu_{\neg x}$	ID
x_3					3
x_4					
x_5					
x_6					

C_l = the number of clauses containing l

μ_l = the average size of clauses containing l

$C_x < C_{\neg x}$ = sign flip

[1] ILLNER, Petr. New Compilation Languages Based on Restricted Weak Decomposability. In: Proceedings of the AAAI Conference on Artificial Intelligence. 2025. p. 14987-14996.

Cara caching scheme[1]

$$\psi_1 = (\neg x_3 \lor \neg x_5) \land (x_4 \lor \neg x_6) \land (\underline{\mathbf{x_3}} \lor x_5 \lor x_6) \land (\neg x_4 \lor x_5 \lor \neg x_6)$$

$\#\psi_1 = ?$

$V_{\psi 1} = \{ 3, 4, 5, 6 \}$

	C_x	$C_{\neg x}$	μ_x	$\mu_{\neg x}$	ID
x_3	1		3		3
x_4					
x_5					
x_6					

C_l = the number of clauses containing l

μ_l = the average size of clauses containing l

$C_x < C_{\neg x}$ = sign flip

[1] ILLNER, Petr. New Compilation Languages Based on Restricted Weak Decomposability. In: Proceedings of the AAAI Conference on Artificial Intelligence. 2025. p. 14987-14996.

Cara caching scheme[1]

$\psi_1 = (\underline{\neg \mathbf{x_3}} \vee \neg x_5) \wedge (x_4 \vee \neg x_6) \wedge (x_3 \vee x_5 \vee x_6) \wedge (\neg x_4 \vee x_5 \vee \neg x_6)$

$\#\psi_1 = ?$

$V_{\psi 1} = \{ 3, 4, 5, 6 \}$

	C_x	$C_{\neg x}$	μ_x	$\mu_{\neg x}$	ID
x_3	1		3		3
x_4					
x_5					
x_6					

C_l = the number of clauses containing l

μ_l = the average size of clauses containing l

$C_x < C_{\neg x}$ = sign flip

[1] ILLNER, Petr. New Compilation Languages Based on Restricted Weak Decomposability. In: Proceedings of the AAAI Conference on Artificial Intelligence. 2025. p. 14987-14996.

Cara caching scheme[1]

$\psi_1 = (\underline{\neg \mathbf{x_3}} \lor \neg x_5) \land (x_4 \lor \neg x_6) \land (x_3 \lor x_5 \lor x_6) \land (\neg x_4 \lor x_5 \lor \neg x_6)$

$\#\psi_1 = ?$

$V_{\psi 1} = \{ 3, 4, 5, 6 \}$

	C_x	$C_{\neg x}$	μ_x	$\mu_{\neg x}$	ID
x_3	1	**1**	3	**2**	3
x_4					
x_5					
x_6					

C_l = the number of clauses containing l

μ_l = the average size of clauses containing l

$C_x < C_{\neg x}$ = sign flip

[1] ILLNER, Petr. New Compilation Languages Based on Restricted Weak Decomposability. In: Proceedings of the AAAI Conference on Artificial Intelligence. 2025. p. 14987-14996.

Cara caching scheme[1]

$\psi_1 = (\neg x_3 \lor \neg x_5) \land (x_4 \lor \neg x_6) \land (x_3 \lor x_5 \lor x_6) \land (\neg x_4 \lor x_5 \lor \neg x_6)$

$\#\psi_1 = ?$

$V_{\psi 1} = \{ 3, 4, 5, 6 \}$

	C_x	$C_{\neg x}$	μ_x	$\mu_{\neg x}$	ID
x_3	1	1	3	2	3
x_4	1	1	2	3	4
x_5	2	1	3	2	5
x_6	1	2	3	2.5	6

C_l = the number of clauses containing l

μ_l = the average size of clauses containing l

$C_x < C_{\neg x}$ = sign flip

[1] ILLNER, Petr. New Compilation Languages Based on Restricted Weak Decomposability. In: Proceedings of the AAAI Conference on Artificial Intelligence. 2025. p. 14987-14996.

Cara caching scheme[1]

$\psi_1 = (\neg x_3 \lor \neg x_5) \land (x_4 \lor \neg x_6) \land (x_3 \lor x_5 \lor x_6) \land (\neg x_4 \lor x_5 \lor \neg x_6)$

$\#\psi_1 = ?$

$V_{\psi 1} = \{ 3, 4, 5, 6 \}$

	C_x	$C_{\neg x}$	μ_x	$\mu_{\neg x}$	ID	Order
x_3	1	1	3	2	3	**2**
x_4	1	1	2	3	4	**1**
x_5	2	1	3	2	5	**4**
x_6	1	2	3	2.5	6	**3**

C_l = the number of clauses containing l

μ_l = the average size of clauses containing l

$C_x < C_{\neg x}$ = sign flip

[1] ILLNER, Petr. New Compilation Languages Based on Restricted Weak Decomposability. In: Proceedings of the AAAI Conference on Artificial Intelligence. 2025. p. 14987-14996.

Cara caching scheme[1]

$\psi_1 = (\neg x_3 \lor \neg x_5) \land (x_4 \lor \neg x_6) \land (x_3 \lor x_5 \lor x_6) \land (\neg x_4 \lor x_5 \lor \neg x_6)$

$\#\psi_1 = ?$

$V_{\psi 1} = \{ 3, 4, 5, 6 \}$

	C_x	$C_{\neg x}$	μ_x	$\mu_{\neg x}$	ID	Order
x_3	1	1	3	2	3	**2**
x_4	1	1	2	3	4	**1**
x_5	2	1	3	2	5	**4**
x_6	1	2	3	2.5	6	**3**

C_l = the number of clauses containing l

μ_l = the average size of clauses containing l

$C_x < C_{\neg x}$ = sign flip

$M_{\psi 1} = \{ 3 \mapsto 2, 4 \mapsto 1, 5 \mapsto 4, 6 \mapsto 3 \}$

[1] ILLNER, Petr. New Compilation Languages Based on Restricted Weak Decomposability. In: Proceedings of the AAAI Conference on Artificial Intelligence. 2025. p. 14987-14996.

Cara caching scheme[1]

$\psi_1 = (\neg x_3 \lor \neg x_5) \land (x_4 \lor \neg x_6) \land (x_3 \lor x_5 \lor x_6) \land (\neg x_4 \lor x_5 \lor \neg x_6)$

$\#\psi_1 = ?$

$V_{\psi 1} = \{ 3, 4, 5, 6 \}$

	C_x		$C_{\neg x}$	μ_x	$\mu_{\neg x}$	**ID**	**Order**
x_3	1		1	3	2	3	**2**
x_4	1		1	2	3	4	**1**
x_5	2		1	3	2	5	**4**
x_6	1	<	2	3	2.5	6	**3**

C_l = the number of clauses containing l

μ_l = the average size of clauses containing l

$C_x < C_{\neg x}$ = sign flip

$M_{\psi 1} = \{ 3 \mapsto 2, 4 \mapsto 1, 5 \mapsto 4, 6 \mapsto \mathbf{-}3 \}$

[1] ILLNER, Petr. New Compilation Languages Based on Restricted Weak Decomposability. In: Proceedings of the AAAI Conference on Artificial Intelligence. 2025. p. 14987-14996.

Cara caching scheme[1]

$\psi_1 = (\neg x_3 \lor \neg x_5) \land (x_4 \lor \neg x_6) \land (x_3 \lor x_5 \lor x_6) \land (\neg x_4 \lor x_5 \lor \neg x_6)$

$\#\psi_1 = ?$

$V_{\psi 1} = \{ 3, 4, 5, 6 \}$

	C_x		$C_{\neg x}$	μ_x	$\mu_{\neg x}$	ID	Order
x_3	1		1	3	2	3	**2**
x_4	1		1	2	3	4	**1**
x_5	2		1	3	2	5	**4**
x_6	1	**<**	2	3	2.5	6	**3**

C_l = the number of clauses containing l

μ_l = the average size of clauses containing l

$C_x < C_{\neg x}$ = sign flip

$M_{\psi 1} = \{ 3 \mapsto 2, 4 \mapsto 1, 5 \mapsto 4, 6 \mapsto -3 \}$

$M_{\psi 1}(\psi_1) = (\neg x_2 \lor \neg x_4) \land (x_1 \lor x_3) \land (x_2 \lor x_4 \lor \neg x_3) \land (\neg x_1 \lor x_4 \lor x_3)$

[1] ILLNER, Petr. New Compilation Languages Based on Restricted Weak Decomposability. In: Proceedings of the AAAI Conference on Artificial Intelligence. 2025. p. 14987-14996.

MODEL COUNTING COMPETITION 2025

Cara caching scheme[1]

$\psi_1 = (\neg x_3 \vee \neg x_5) \wedge (x_4 \vee \neg x_6) \wedge (x_3 \vee x_5 \vee x_6) \wedge (\neg x_4 \vee x_5 \vee \neg x_6)$

$\#\psi_1 = ?$

$V_{\psi 1} = \{ 3, 4, 5, 6 \}$

	C_x		$C_{\neg x}$	μ_x	$\mu_{\neg x}$	ID	Order
x_3	1		1	3	2	3	**2**
x_4	1		1	2	3	4	**1**
x_5	2		1	3	2	5	**4**
x_6	1	<	2	3	2.5	6	**3**

C_l = the number of clauses containing l

μ_l = the average size of clauses containing l

$C_x < C_{\neg x}$ = sign flip

$M_{\psi 1} = \{ 3 \mapsto 2, 4 \mapsto 1, 5 \mapsto 4, 6 \mapsto -3 \}$

$M_{\psi 1}(\psi_1) = (\neg x_2 \vee \neg x_4) \wedge (x_1 \vee x_3) \wedge (x_2 \vee x_4 \vee \neg x_3) \wedge (\neg x_1 \vee x_4 \vee x_3)$

$r(\psi_1) = \underbrace{1, 3, 0, -2, -4, 0, -1, 3, 4, 0, 2, -3, 4, 0}_{\textbf{sorted residual clauses}}$

[1] ILLNER, Petr. New Compilation Languages Based on Restricted Weak Decomposability. In: Proceedings of the AAAI Conference on Artificial Intelligence. 2025. p. 14987-14996.

Cara caching scheme[1]

$\psi_1 = (\neg x_3 \lor \neg x_5) \land (x_4 \lor \neg x_6) \land (x_3 \lor x_5 \lor x_6) \land (\neg x_4 \lor x_5 \lor \neg x_6)$

$\#\psi_1 = \mathbf{5}$

$V_{\psi 1} = \{ 3, 4, 5, 6 \}$

	C_x		$C_{\neg x}$	μ_x	$\mu_{\neg x}$	ID	Order
x_3	1		1	3	2	3	**2**
x_4	1		1	2	3	4	**1**
x_5	2		1	3	2	5	**4**
x_6	1	<	2	3	2.5	6	**3**

C_l = the number of clauses containing l

μ_l = the average size of clauses containing l

$C_x < C_{\neg x}$ = sign flip

$M_{\psi 1} = \{ 3 \mapsto 2, 4 \mapsto 1, 5 \mapsto 4, 6 \mapsto -3 \}$

$M_{\psi 1}(\psi_1) = (\neg x_2 \lor \neg x_4) \land (x_1 \lor x_3) \land (x_2 \lor x_4 \lor \neg x_3) \land (\neg x_1 \lor x_4 \lor x_3)$

$r(\psi_1) = \underbrace{1, 3, 0, -2, -4, 0, -1, 3, 4, 0, 2, -3, 4, 0}_{\textbf{sorted residual clauses}}$

cache entry: key $= r(\psi_1)$,
value $= \#\psi_1 = 5$

[1] ILLNER, Petr. New Compilation Languages Based on Restricted Weak Decomposability. In: Proceedings of the AAAI Conference on Artificial Intelligence. 2025. p. 14987-14996.

Cara caching scheme[1] - key properties

1) Deterministic representations
2) Syntactically identical (up to ordering) CNF formulae yield the same representation and mapping function
=> everything detected by *i* is also detected by *Cara*
3) Additional moments can be used in tuples to enhance isomorphism detection

4) Construction is computationally expensive
5) Representations tend to be larger

[1] ILLNER, Petr. New Compilation Languages Based on Restricted Weak Decomposability. In: Proceedings of the AAAI Conference on Artificial Intelligence. 2025. p. 14987-14996.

Cara *caching scheme*

vs

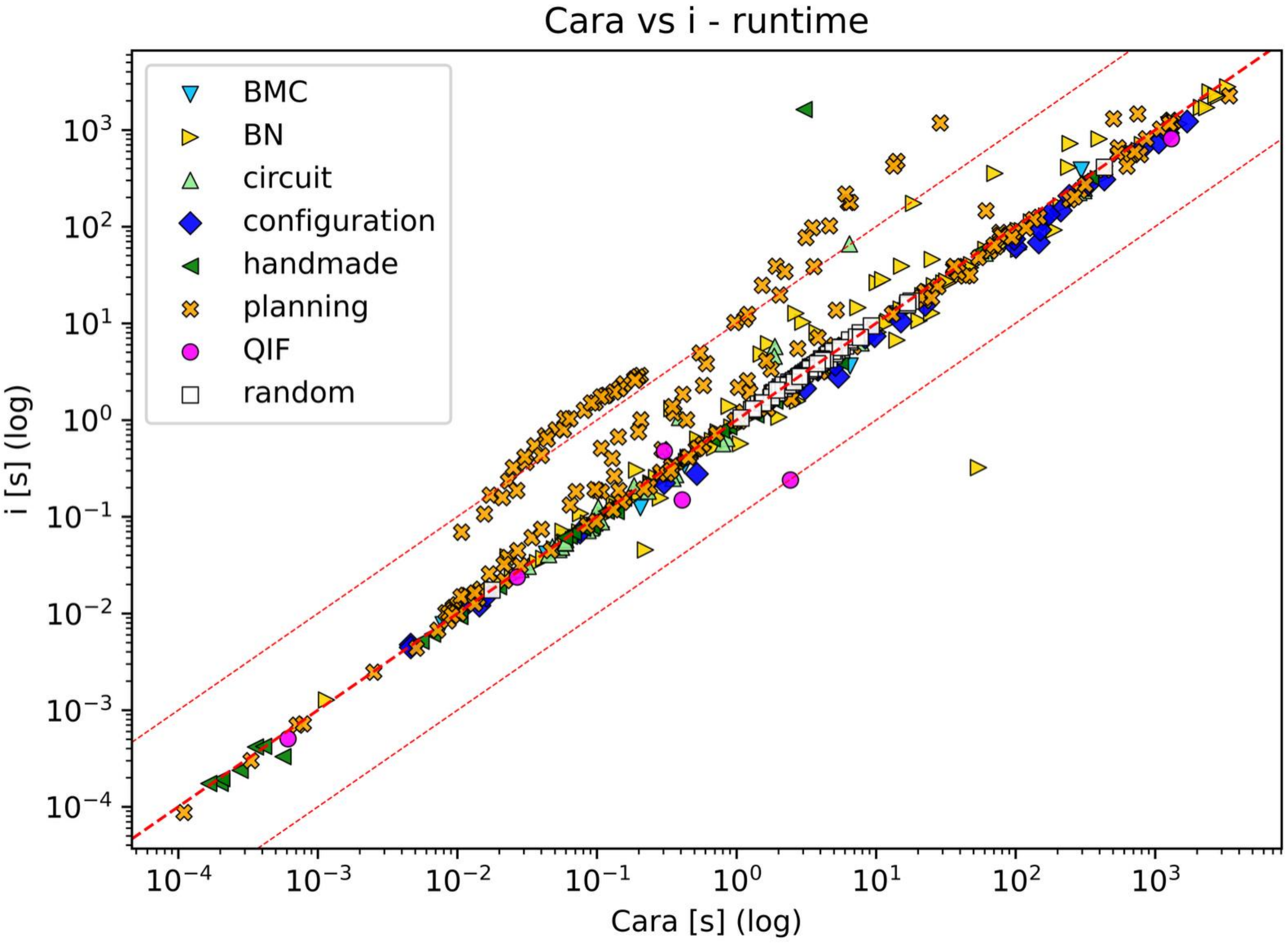
caching scheme ***i***

Experiments

The time-out (resp. memory-out) was set to **1 hour** (resp. **32 GB**).

Each instance was solved **3** times, and the given results are averages.

#solved runs	***Cara***	*i*
3/3	**593**	586
2/3	**1**	5
1/3	**0**	0
Total	**594**	591

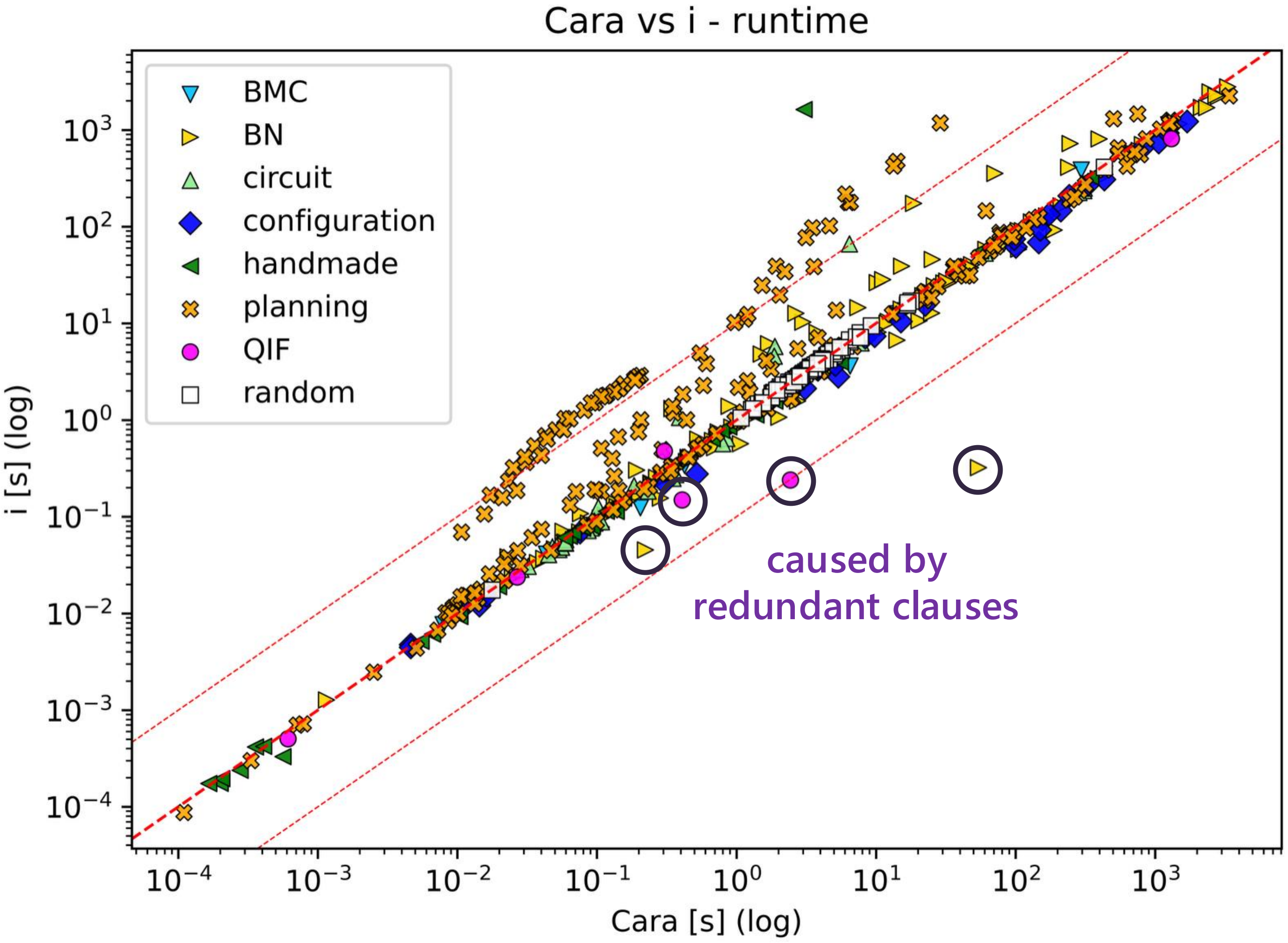


Experiments

The time-out (resp. memory-out) was set to **1 hour** (resp. **32 GB**).

Each instance was solved **3** times, and the given results are averages.

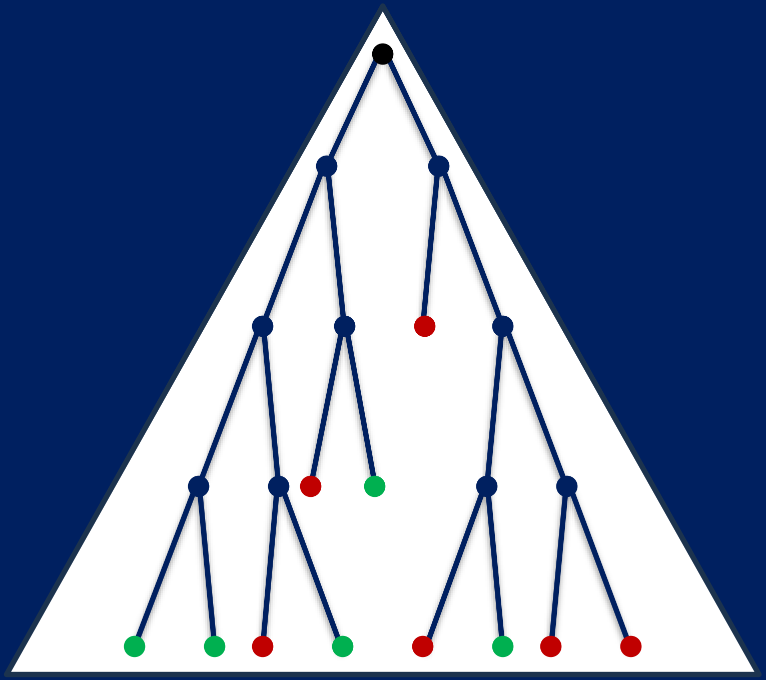
#solved runs	*Cara*	*i*
3/3	**593**	586
2/3	**1**	5
1/3	**0**	0
Total	**594**	591



Which caching scheme is better?

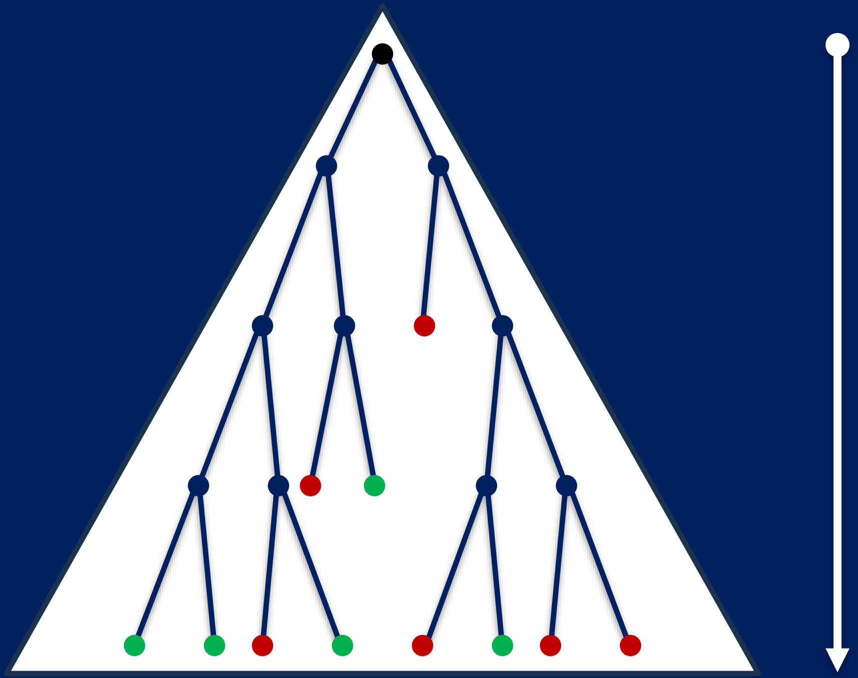
Balance of caching schemes – future research

search space



Balance of caching schemes – future research

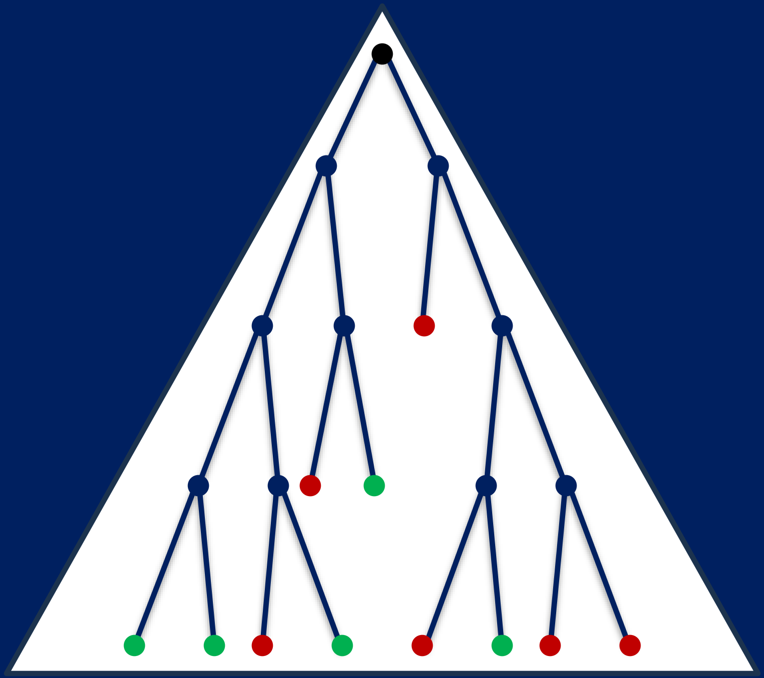
search space



larger residual formulae
lower isomorphism detection
more untouched clauses

Balance of caching schemes – future research

search space

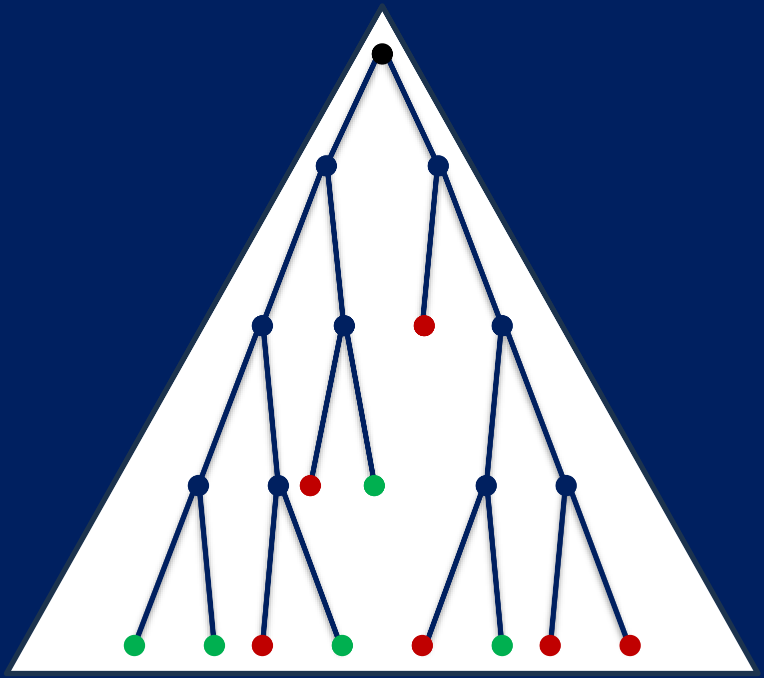


larger residual formulae
lower isomorphism detection
more untouched clauses

smaller residual formulae
higher isomorphism detection
fewer untouched clauses

Balance of caching schemes – future research

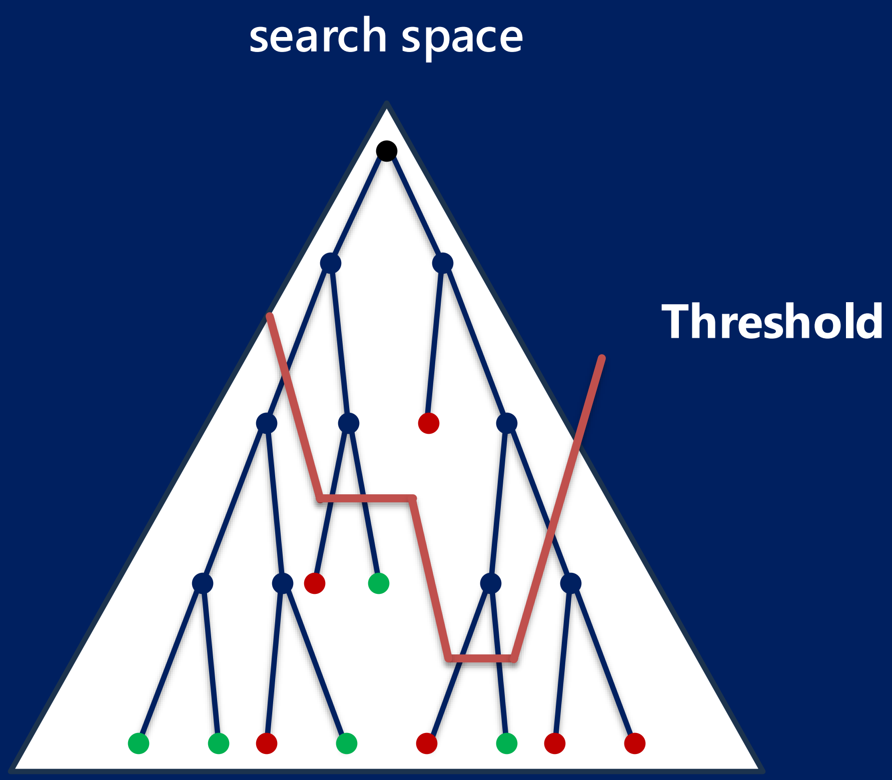
search space



caching scheme
(standard, basic, i, hybrid, PCC, …)

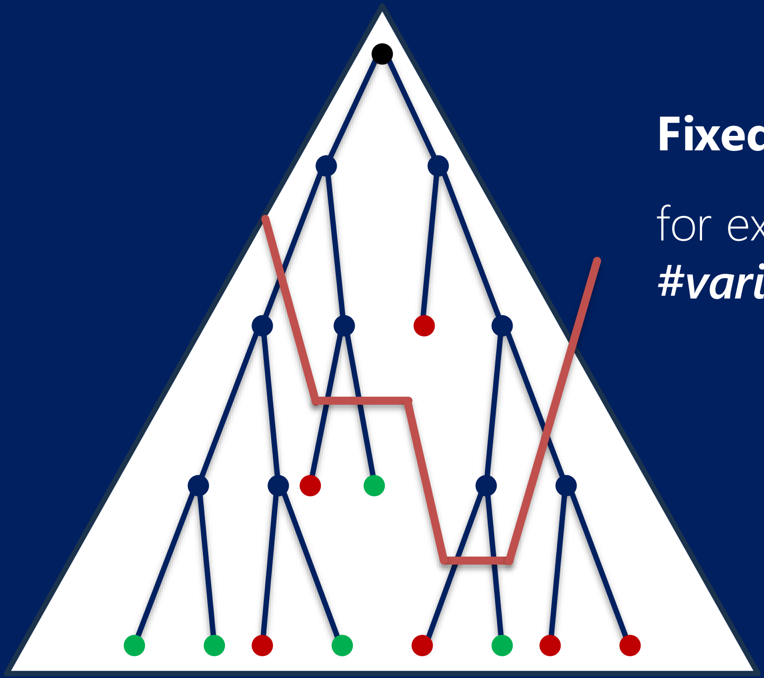
isomorphism-aware caching scheme
(Cara, …)

Balance of caching schemes – future research



Balance of caching schemes – future research

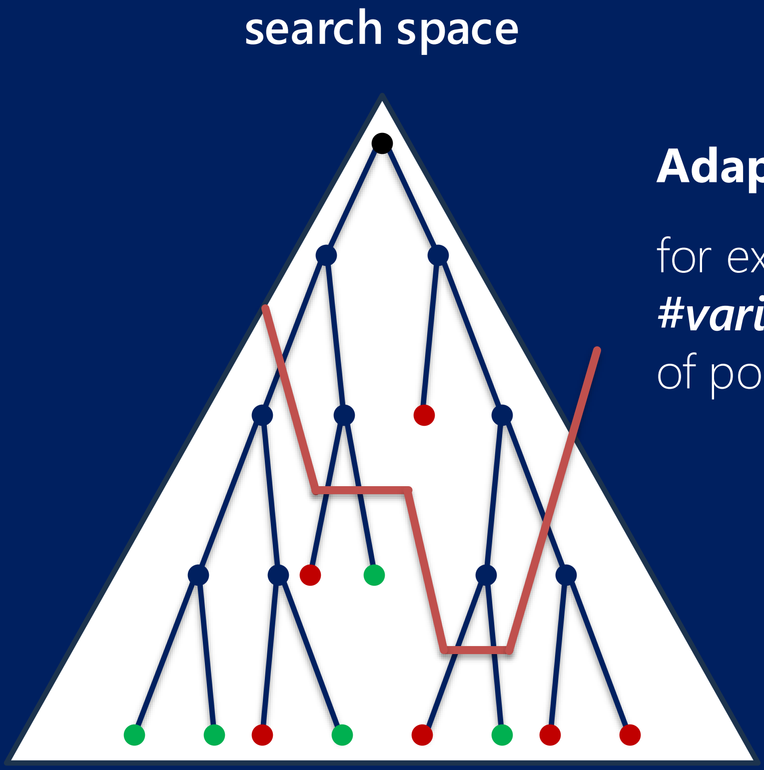
search space



Fixed threshold:

for example,
#variables / *#clauses* / *formula size <= "X"*

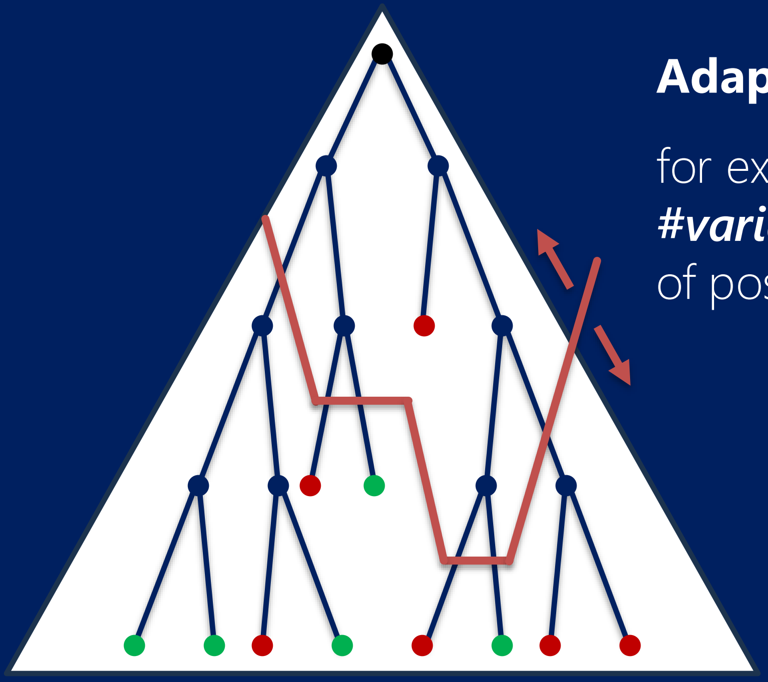
Balance of caching schemes – future research



Adaptive threshold:

for example, the **average** / maximum ***#variables*** / *#clauses* / *formula size* of positively cached residual formulae

Balance of caching schemes – future research



Adaptive threshold:

for example, the **average** / maximum ***#variables*** / *#clauses* / *formula size* of positively cached residual formulae

Note: Invalid cache entries can be easily detected and removed when the next cache cleaning strategy is triggered.

Cara[1] + *Arjun*[*2]

VS

Ganak[3]

[*] ensuring identical preprocessing

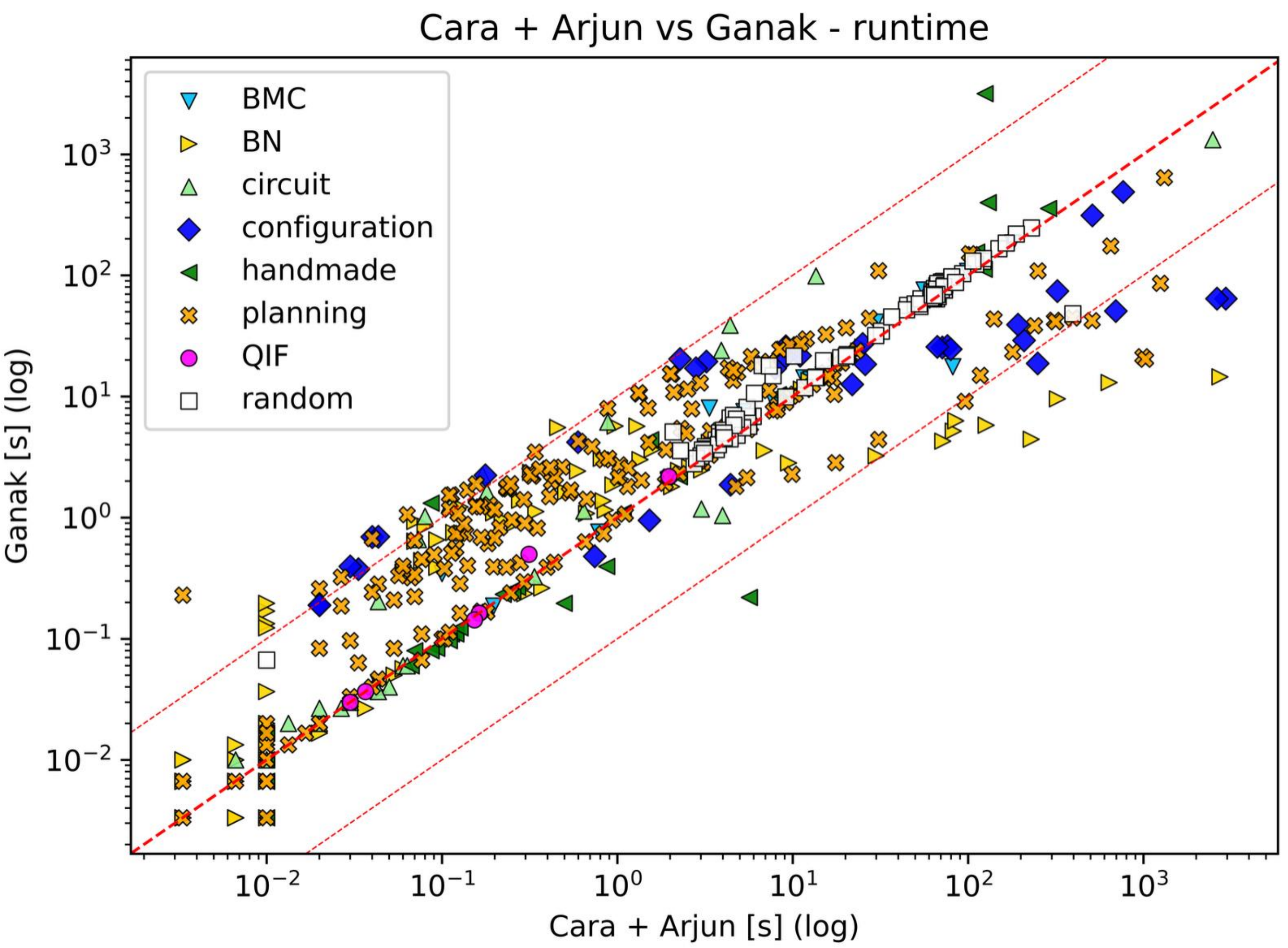
[1] https://github.com/Illner/carasolver
[2] https://github.com/meelgroup/arjun
[3] https://github.com/meelgroup/ganak

Experiments

The time-out (resp. memory-out) was set to **1 hour** (resp. **32 GB**).

Each instance was solved 3 times, and the given results are averages.

#solved runs	***Cara***	*Ganak*
3/3	623	632
2/3	0	0
1/3	0	0
Total	623	**632**

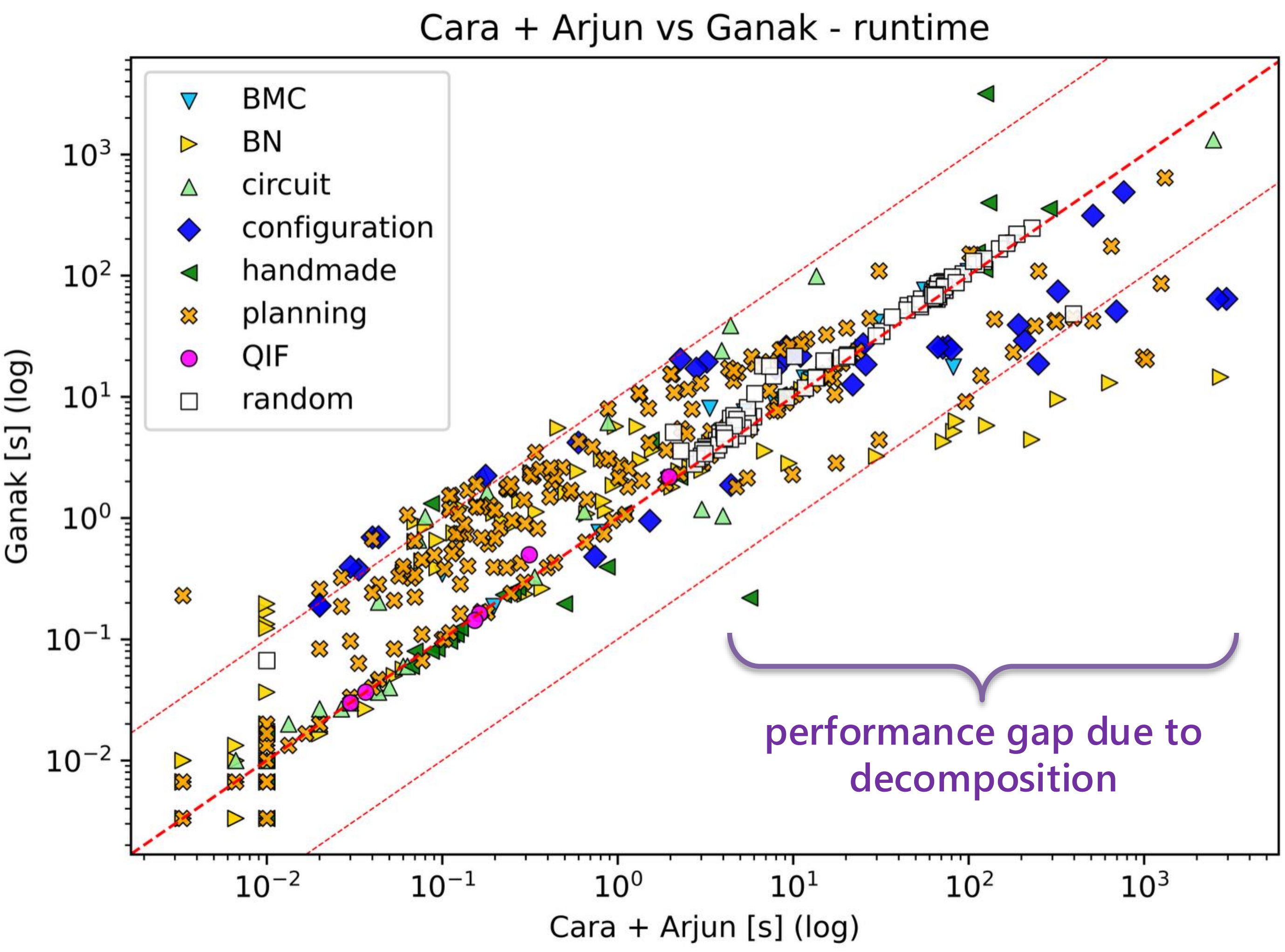


Experiments

The time-out (resp. memory-out) was set to **1 hour** (resp. **32 GB**).

Each instance was solved 3 times, and the given results are averages.

#solved runs	***Cara***	*Ganak*
3/3	623	632
2/3	0	0
1/3	0	0
Total	623	**632**



Cara[1]

vs

SymGanak*[2]

* no Arjun

[1] https://github.com/Illner/carasolver
[2] https://github.com/meelgroup/ganak

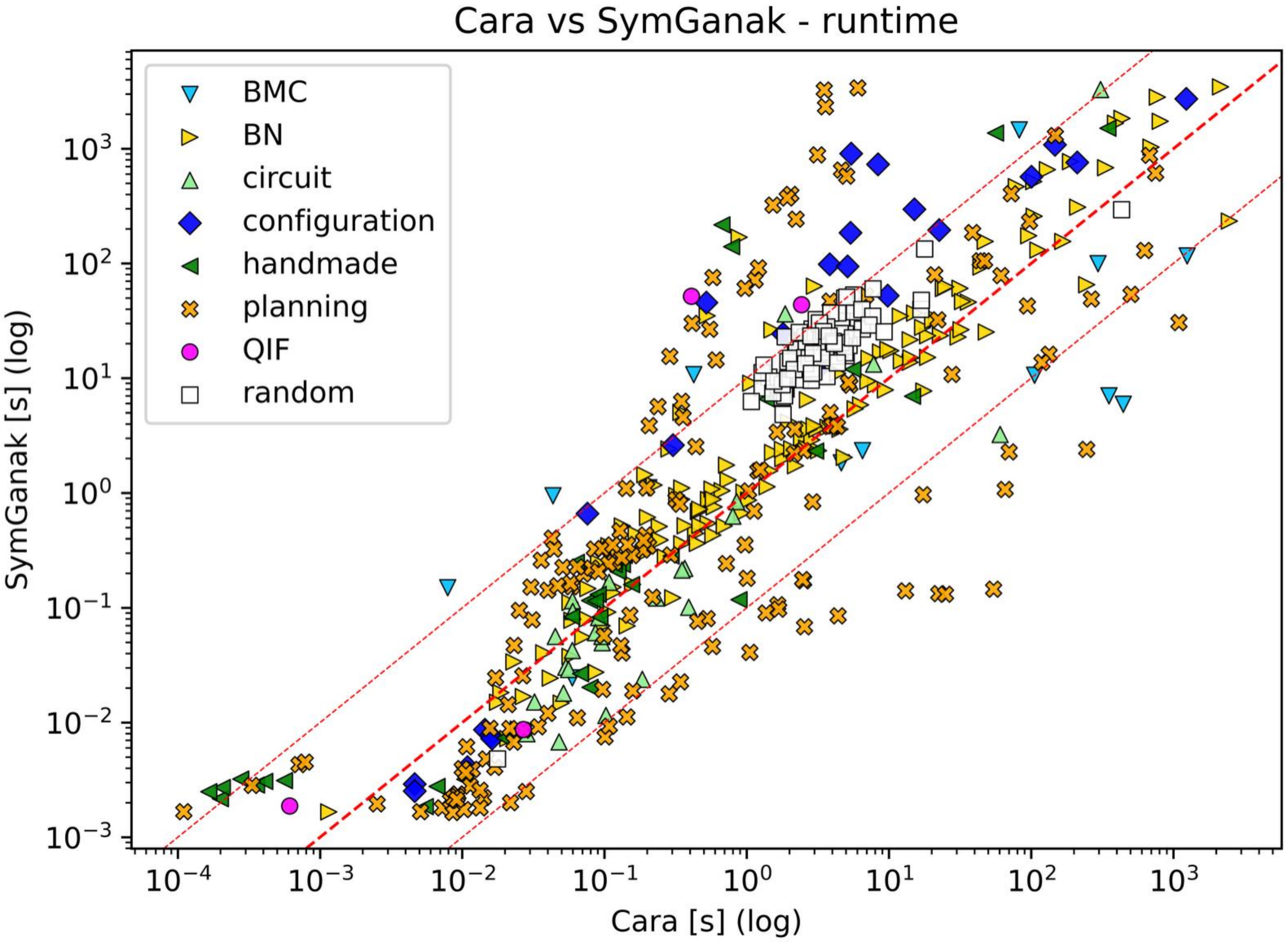
Experiments

The time-out (resp. memory-out) was set to **1 hour** (resp. **32 GB**).

Each instance was solved 3 times, and the given results are averages.

#solved runs	***Cara****[*]	*SymGanak*
3/3	**593**	512
2/3	**1**	0
1/3	**0**	0
Total	**594**	512

[*] strict dominance

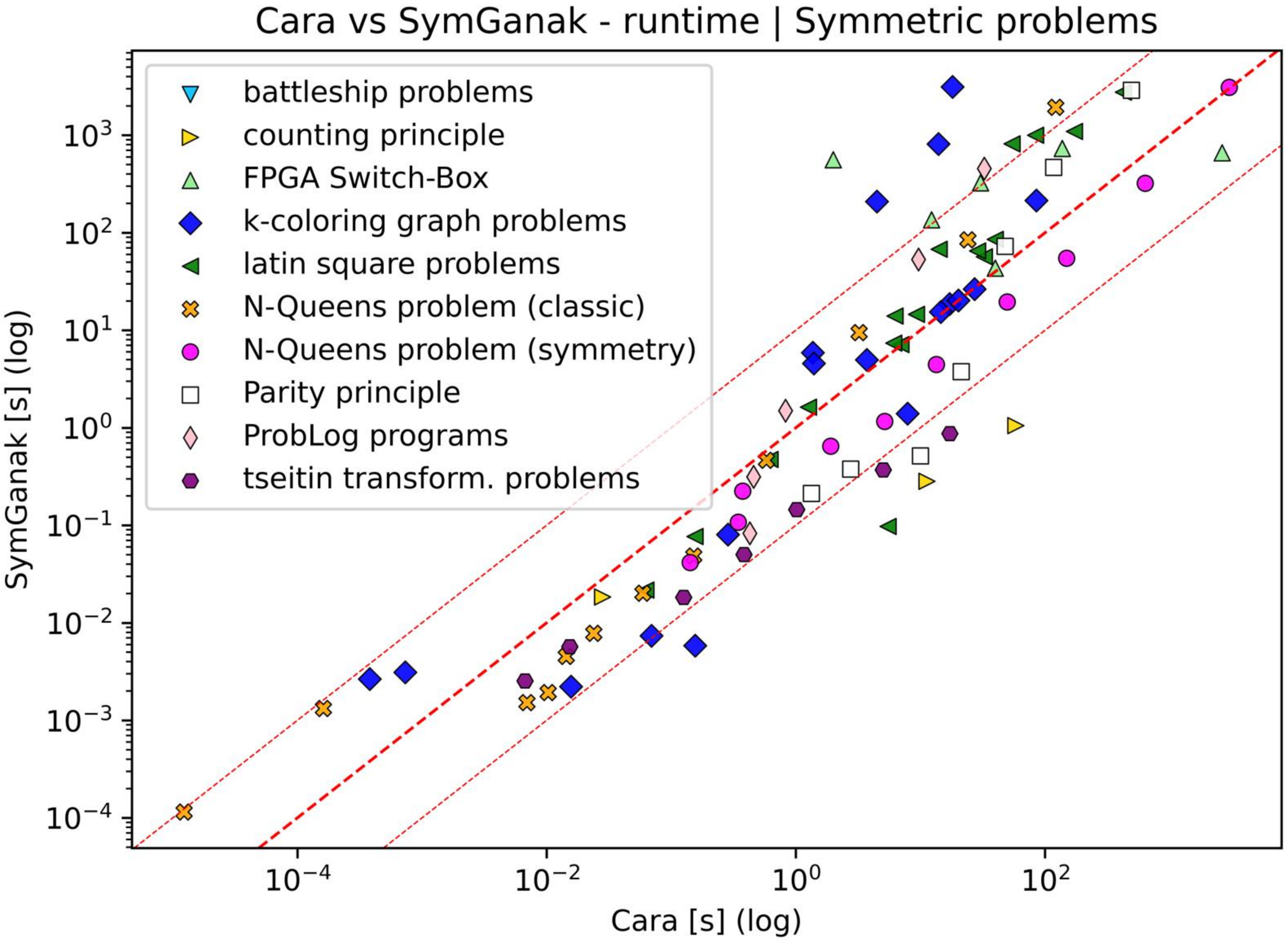


Experiments

The time-out (resp. memory-out) was set to **1 hour** (resp. **32 GB**).

Each instance was solved 3 times, and the given results are averages.

#solved runs	***Cara***	*SymGanak*
3/3	**98**	93
2/3	1	1
1/3	1	0
Total	**100**	94



Symmetric problems

Problem[1]	#instances	#successfully solved instances	
		Cara	SymGanak
Battleship problems	12	**1**	0
ProbLog programs	6	**6**	5
N-Queens problem (symmetry)	11	**10**	**10**
N-Queens problem (classic)	17	**13**	12
Latin square problems	30	**20**	17
Counting principle	9	3	**6**
k-coloring graph problems	23	**21**	18
Parity principle	21	**8**	7
Tseitin transformation problems	16	7	**13**
FPGA Switch-Box	20	**11**	6

[1] https://github.com/VincentDerk/Paper-SymGANAK-benchmark

MODEL COUNTING COMPETITION 2025

Functional pigeonhole principle (FPHP)

Instance[1]	#pigeons	#holes
fphp-010-020	10	20
fphp-015-020	15	20

[1] https://www.cril.univ-artois.fr/kc/benchmarks.html

Functional pigeonhole principle (FPHP)

Instance[1]	#pigeons	#holes
fphp-010-020	10	20
fphp-015-020	15	20

Instance[1]	Runtime [s]					
	***Cara*[2]**	***Cara*[2] + *Arjun*[3]**	*SymGanak*[4]	*Ganak*[4]	*SharpSAT-TD*[5]	*D4*[6]
fphp-010-020	**3.04**	124.81	**2.37**	3 176.17*	—	—
fphp-015-020	**14.61**	314.69	**7.03**	—	—	—

* invalid number of models

[1] https://www.cril.univ-artois.fr/kc/benchmarks.html
[2] https://github.com/lllner/carasolver
[3] https://github.com/meelgroup/arjun
[4] https://github.com/meelgroup/ganak
[5] https://github.com/Laakeri/sharpsat-td
[6] https://github.com/crillab/d4v2

MODEL COUNTING COMPETITION 2025

N-Queens problem

N	Runtime [s]	
	Cara[1]	*SymGanak*[2]
10	0.17	**0.05**
11	0.62	**0.47**
12	**3.37**	9.81
13	**25.89**	92.09
14	**128.39**	1858.32
15	**1264.51**	—

[1] https://github.com/Illner/carasolver
[2] https://github.com/meelgroup/ganak

Future research

1) Adaptive balance of caching schemes
2) Replace the obsolete *MiniSat*[1], which is used for satisfiability checks and implied literals at each inner node, with *CaDiCaL*[2].
3) **Integrate the decomposability approach used by *Ganak* and *SharpSAT-TD* into the adaptiveness framework.**

[1] EÉN, Niklas; SÖRENSSON, Niklas. An extensible SAT-solver. In: International conference on theory and applications of satisfiability testing. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003. p. 502-518.
[2] BIERE, Armin, et al. CaDiCaL 2.0. In: International Conference on Computer Aided Verification. Cham: Springer Nature Switzerland, 2024. p. 133-152.