UpMax: User partitioning for MaxSAT

Pedro Orvalho [1], Vasco Manquinho [1] and Ruben Martins [2]

[1]INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal
[2]Carnegie Mellon University, PA, USA

SAT 2023, Alghero, Italy
July 8, 2023

TÉCNICO LISBOA

FCT
Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA EDUCAÇÃO E CIÊNCIA

inesc id lisboa

Carnegie Mellon University

Partitioning of MaxSAT Formulae

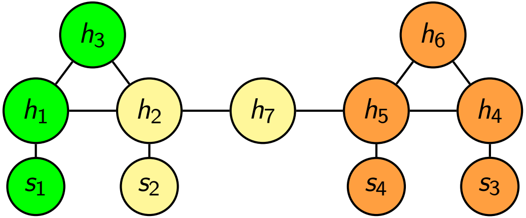
Hard:	$h_1 : (v_1 \vee v_2)$	$h_2 : (\neg v_2 \vee v_3)$	$h_3 : (\neg v_1 \vee \neg v_3)$	$h_4 : (v_4 \vee v_5)$
	$h_5 : (\neg v_5 \vee v_6)$	$h_6 : (\neg v_4 \vee \neg v_6)$	$h_7 : (\neg v_3 \vee \neg v_6)$	
Soft:	$s_1 : (\neg v_1)$	$s_2 : (\neg v_3)$	$s_3 : (\neg v_4)$	$s_4 : (\neg v_6)$

Partitioning of MaxSAT Formulae

Hard: $h_1 : (v_1 \lor v_2)$ $h_2 : (\neg v_2 \lor v_3)$ $h_3 : (\neg v_1 \lor \neg v_3)$ $h_4 : (v_4 \lor v_5)$
$h_5 : (\neg v_5 \lor v_6)$ $h_6 : (\neg v_4 \lor \neg v_6)$ $h_7 : (\neg v_3 \lor \neg v_6)$

Soft: $s_1 : (\neg v_1)$ $s_2 : (\neg v_3)$ $s_3 : (\neg v_4)$ $s_4 : (\neg v_6)$

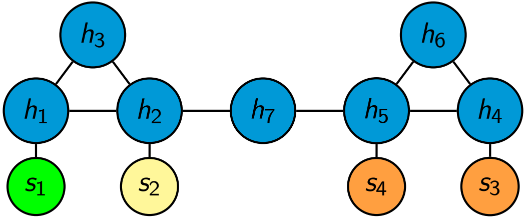
- Instead of dealing with the whole formula at once, some MaxSAT algorithms try to **split the formula into partitions** [Martins et al., 2012].



Partitioning of MaxSAT Formulae

Hard:	$h_1 : (v_1 \vee v_2)$	$h_2 : (\neg v_2 \vee v_3)$	$h_3 : (\neg v_1 \vee \neg v_3)$	$h_4 : (v_4 \vee v_5)$
	$h_5 : (\neg v_5 \vee v_6)$	$h_6 : (\neg v_4 \vee \neg v_6)$	$h_7 : (\neg v_3 \vee \neg v_6)$	
Soft:	$s_1 : (\neg v_1)$	$s_2 : (\neg v_3)$	$s_3 : (\neg v_4)$	$s_4 : (\neg v_6)$

- In particular, the partitioning focuses on **splitting the set of soft clauses into disjoint sets**.

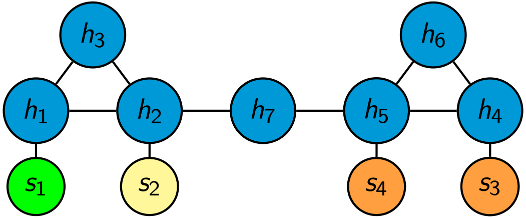


Partitioning of MaxSAT Formulae

Hard: $h_1 : (v_1 \vee v_2)$ $h_2 : (\neg v_2 \vee v_3)$ $h_3 : (\neg v_1 \vee \neg v_3)$ $h_4 : (v_4 \vee v_5)$
$h_5 : (\neg v_5 \vee v_6)$ $h_6 : (\neg v_4 \vee \neg v_6)$ $h_7 : (\neg v_3 \vee \neg v_6)$

Soft: $s_1 : (\neg v_1)$ $s_2 : (\neg v_3)$ $s_3 : (\neg v_4)$ $s_4 : (\neg v_6)$

- quickly identify a minimal cost;
- easier to solve;
- faster convergence to the optimum.



Partitioning of MaxSAT Formulae

There have been proposed several types of automatic partitioning methods, such as:

- Partitioning of soft clauses according to their weight [Ansotegui et al., 2012];

Partitioning of MaxSAT Formulae

There have been proposed several types of automatic partitioning methods, such as:

- Partitioning of soft clauses according to their weight [Ansotegui et al., 2012];
- Graph-based partitioning of partial MaxSAT formulae [Neves et al., 2015].

Partitioning of MaxSAT Formulae

Current Drawbacks

1. Partitioning methods are **interconnected to the MaxSAT algorithms**;

Partitioning of MaxSAT Formulae

Current Drawbacks

1. Partitioning methods are **interconnected to the MaxSAT algorithms**;
2. **Difficult to define and test new partitioning methods** with several MaxSAT algorithms;

Partitioning of MaxSAT Formulae

Current Drawbacks

1. Partitioning methods are **interconnected to the MaxSAT algorithms**;
2. **Difficult to define and test new partitioning methods** with several MaxSAT algorithms;
3. **Graph representations may become too large**;

Partitioning of MaxSAT Formulae

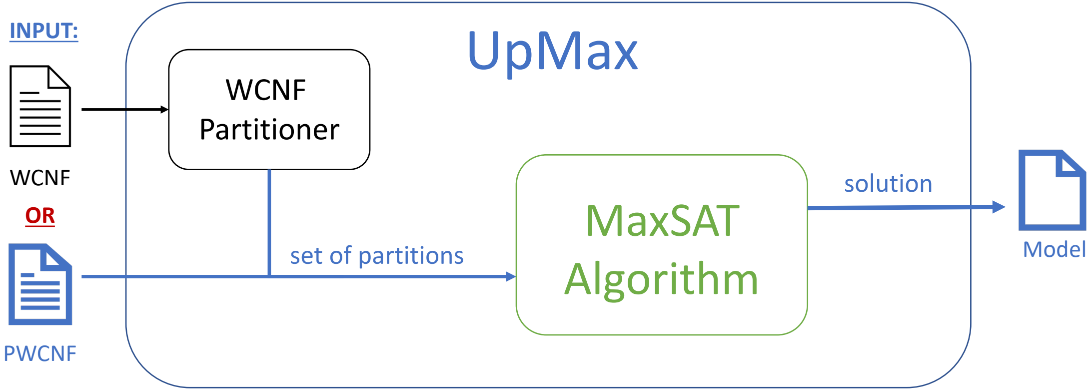
Current Drawbacks

1. Partitioning methods are **interconnected to the MaxSAT algorithms**;
2. **Difficult to define and test new partitioning methods** with several MaxSAT algorithms;
3. **Graph representations may become too large**;
4. **The partitions might not capture the problem structure** that is helpful for MaxSAT solving.

Partitioning of MaxSAT Formulae

Current Drawbacks

1. Partitioning methods are **interconnected to the MaxSAT algorithms**;
2. **Difficult to define and test new partitioning methods** with several MaxSAT algorithms;
3. **Graph representations may become too large**;
4. **The partitions might not capture the problem structure** that is helpful for MaxSAT solving.
5. The `wcnf` **format does not support the users to provide a partitioning scheme**.


INPUT:
WCNF
OR
PWCNF
UpMax
WCNF
Partitioner
set of partitions
MaxSAT
Algorithm
solution
Model

pwcnf format

- The pwcnf format starts with a header:

```
p pwcnf n_vars n_clauses topw n_part
```

`pwcnf` format

- The `pwcnf` format starts with a header:

```
p pwcnf n_vars n_clauses topw n_part
```

- and each line in the body is of the form:

```
[part] [weight] [literals*] 0
```

Use Cases

Use Case: Minimum Sum Coloring (MSC)

- MSC is a graph coloring problem.

Use Case: Minimum Sum Coloring (MSC)

- MSC is a graph coloring problem.
- Requirements:

Use Case: Minimum Sum Coloring (MSC)

- MSC is a graph coloring problem.
- Requirements:
  - Each vertex should be assigned exactly one color;

Use Case: Minimum Sum Coloring (MSC)

- MSC is a graph coloring problem.
- Requirements:
  - Each vertex should be assigned exactly one color;
  - Two adjacent vertices cannot be assigned the same color.

Use Case: Minimum Sum Coloring (MSC)

- MSC is a graph coloring problem.
- Requirements:
  - Each vertex should be assigned exactly one color;
  - Two adjacent vertices cannot be assigned the same color.
- **Goal:** minimize the number of different colors in the graph.

Use Case: Minimum Sum Coloring (MSC)

- X_c^v is true if color c is assigned to vertex v.

Use Case: Minimum Sum Coloring (MSC)

- X_c^v is true if color c is assigned to vertex v.
- **Goal (soft clauses):** Maximize $\neg X_c^v$ (weight $= c$).

Use Case: Minimum Sum Coloring (MSC)

Example

Given a graph G with 4 vertices, $v_1, \ldots, v_4$, and 4 different colors available $c_1, \ldots, c_4$.

Use Case: Minimum Sum Coloring (MSC)

Example

Given a graph G with 4 vertices, $v_1, \ldots, v_4$, and 4 different colors available $c_1, \ldots, c_4$.

When encoding the problem into `pwcnf` the user could provide either the following *VERTEX-based* or *COLOR-based* partition schemes:

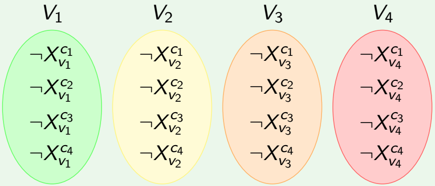
Use Case: Minimum Sum Coloring (MSC)

Example

Given a graph G with 4 vertices, $v_1, \ldots, v_4$, and 4 different colors available $c_1, \ldots, c_4$.

When encoding the problem into `pwcnf` the user could provide either the following *VERTEX-based* or *COLOR-based* partition schemes:

VERTEX-based



Use Case: Minimum Sum Coloring (MSC)

Example

Given a graph G with 4 vertices, $v_1, \ldots, v_4$, and 4 different colors available $c_1, \ldots, c_4$.

When encoding the problem into `pwcnf` the user could provide either the following *VERTEX-based* or *COLOR-based* partition schemes:

COLOR-based

C_1	C_2	C_3	C_4
$\neg X_{v_1}^{c_1}$	$\neg X_{v_1}^{c_2}$	$\neg X_{v_1}^{c_3}$	$\neg X_{v_1}^{c_4}$
$\neg X_{v_2}^{c_1}$	$\neg X_{v_2}^{c_2}$	$\neg X_{v_2}^{c_3}$	$\neg X_{v_2}^{c_4}$
$\neg X_{v_3}^{c_1}$	$\neg X_{v_3}^{c_2}$	$\neg X_{v_3}^{c_3}$	$\neg X_{v_3}^{c_4}$
$\neg X_{v_4}^{c_1}$	$\neg X_{v_4}^{c_2}$	$\neg X_{v_4}^{c_3}$	$\neg X_{v_4}^{c_4}$

Use Case: Seating Assignment (SA)

We want to seat persons at tables such that the following properties are met:

Use Case: Seating Assignment (SA)

We want to seat persons at tables such that the following properties are met:

- Each table has a minimum and a maximum number of persons;

Use Case: Seating Assignment (SA)

We want to seat persons at tables such that the following properties are met:

- Each table has a minimum and a maximum number of persons;
- Each person is seated at exactly one table;

Use Case: Seating Assignment (SA)

We want to seat persons at tables such that the following properties are met:

- Each table has a minimum and a maximum number of persons;
- Each person is seated at exactly one table;
- Each person has some tags that represent their interests.

Use Case: Seating Assignment (SA)

We want to seat persons at tables such that the following properties are met:

- Each table has a minimum and a maximum number of persons;
- Each person is seated at exactly one table;
- Each person has some tags that represent their interests.

Goal: minimize the number of different tags between all persons seated at the same table.

Use Case: Seating Assignment (SA)

- Y_t^g is true if there is *at least one person* p with a tag g that is seated at table t.

Use Case: Seating Assignment (SA)

- Y_t^g is true if there is *at least one person* p with a tag g that is seated at table t.
- **Goal (soft clauses):** maximize $\neg Y_t^g$.

Use Case: Seating Assignment (SA)

Example

Consider that a user wants to seat 5 persons, $p_1, \ldots, p_5$, in two tables t_1, t_2. Moreover, the set of different tags is $\{$A, B, C$\}$.

Use Case: Seating Assignment (SA)

Example

Consider that a user wants to seat 5 persons, $p_1, \ldots, p_5$, in two tables t_1, t_2. Moreover, the set of different tags is $\{$A, B, C$\}$.

The user could provide either the following *TAGS-based* or *TABLES-based* partition scheme:

Use Case: Seating Assignment (SA)

Example

Consider that a user wants to seat 5 persons, $p_1, \ldots, p_5$, in two tables t_1, t_2. Moreover, the set of different tags is {A, B, C}.

The user could provide either the following *TAGS-based* or *TABLES-based* partition scheme:

TAGS-based

A	B	C
$\neg Y^{A}_{t_1}$	$\neg Y^{B}_{t_1}$	$\neg Y^{C}_{t_1}$
$\neg Y^{A}_{t_2}$	$\neg Y^{B}_{t_2}$	$\neg Y^{C}_{t_2}$

Use Case: Seating Assignment (SA)

Example

Consider that a user wants to seat 5 persons, $p_1, \ldots, p_5$, in two tables t_1, t_2. Moreover, the set of different tags is $\{$A, B, C$\}$.

The user could provide either the following *TAGS-based* or *TABLES-based* partition scheme:

TABLES-based

t_1	t_2
$\neg Y_{t_1}^{A}$	$\neg Y_{t_2}^{A}$
$\neg Y_{t_1}^{B}$	$\neg Y_{t_2}^{B}$
$\neg Y_{t_1}^{C}$	$\neg Y_{t_2}^{C}$

Implementation

UpMax

- UpMax is built on top of the open-source Open-WBO MaxSAT solver [Martins et al., 2014];

UpMax

- UpMax is built on top of the open-source Open-WBO MaxSAT solver [Martins et al., 2014];
- UpMax supports the new format `pwcnf` for user-based partitioning.

UpMax

- UpMax is built on top of the open-source Open-WBO MaxSAT solver [Martins et al., 2014];
- UpMax supports the new format `pwcnf` for user-based partitioning.
- It can also take as **input a `wcnf`** formula and **output a `pwcnf`** formula using an automatic partitioning strategy based on:
  - VIG;
  - CVIG;
  - RES;
  - randomly splitting the formula into k partitions.

UpMax

- UpMax currently supports **three UNSAT-based algorithms** (**WBO** [Manquinho et al., 2009], **OLL** [Morgado et al., 2014], and **MSU3** [Martins et al., 2014 (b)]) for both unweighted and weighted problems that take advantage of the partitions;

UpMax

- UpMax currently supports **three UNSAT-based algorithms** (**WBO** [Manquinho et al., 2009], **OLL** [Morgado et al., 2014], and **MSU3** [Martins et al., 2014 (b)]) for both unweighted and weighted problems that take advantage of the partitions;
- We have also extended **RC2** [Ignatiev et al., 2019] and **Hitman** [Moreno-Centeno et al., 2013], available in PySAT [Ignatiev et al., 2018], to use our `pwcnf` formulae.

Experimental Results

Experimental Results

- We randomly generate **1,000 instances for both use cases** by varying the different parameters of each problem.

Experimental Results

- We randomly generate **1,000 instances for both use cases** by varying the different parameters of each problem.

- Partitioning strategies used:
  - Graph-based partitions (VIG, CVIG, RES);
  - Random partitioning strategy ($k = 16$);
  - User-based partitions (UP):
    - *VERTEX/COLOR-based* partitions (MSC);
    - *TAGS/TABLES-based* partitions (SA);
  - No partitions (`wcnf`).

Experimental Results

- We randomly generate **1,000 instances for both use cases** by varying the different parameters of each problem.

- Partitioning strategies used:
  - Graph-based partitions (VIG, CVIG, RES);
  - Random partitioning strategy ($k = 16$);
  - User-based partitions (UP):
    - *VERTEX/COLOR-based* partitions (MSC);
    - *TAGS/TABLES-based* partitions (SA);
  - No partitions (`wcnf`).

- All of the experiments were run on StarExec [Stump et al., 2014], with a timeout of 1800 seconds and a memory limit of 32 GB.

Use Case: Minimum Sum Coloring

Table: Number of solved instances for the Minimum Sum Coloring (MSC) problem.

		User Part.		Graph Part.			
Solver	**No Part.**	**Vertex**	**Color**	**VIG**	**CVIG**	**RES**	**Random**
MSU3	245	758	770	774	770	775	**776**
OLL	796	863	594	945	944	**947**	756
WBO	483	622	314	745	750	**755**	493
Hitman	610	613	471	605	**614**	609	592
RC2	796	866	528	943	939	**944**	687

Use Case: Minimum Sum Coloring

Table: Number of solved instances for the Minimum Sum Coloring (MSC) problem.

		User Part.		Graph Part.			
Solver	**No Part.**	**Vertex**	**Color**	**VIG**	**CVIG**	**RES**	**Random**
MSU3	245	758	770	774	770	775	**776**
OLL	796	863	594	945	944	**947**	756
WBO	483	622	314	745	750	**755**	493
Hitman	610	613	471	605	**614**	609	592
RC2	796	866	528	943	939	**944**	687

MaxSAT Eval 2022:

- EvalMaxSAT: 729; MaxHS: 873; CASHWMaxSAT: 993;
- UWrMaxSat: 994; **MaxCDCL: 995**.

Use Case: Seating Assignment

Table: Number of solved instances for the Seating Assignment problem.

		User Part.		Graph Part.			
Solver	**No Part.**	**Table**	**Tag**	**VIG**	**CVIG**	**RES**	**Random**
MSU3	558	**671**	639	659	641	640	565
OLL	526	**634**	624	627	599	608	528
WBO	306	400	**536**	400	385	386	360
Hitman	420	403	**510**	406	425	420	440
RC2	530	620	**624**	618	600	597	541

Use Case: Seating Assignment

Table: Number of solved instances for the Seating Assignment problem.

		User Part.		Graph Part.			
Solver	**No Part.**	**Table**	**Tag**	**VIG**	**CVIG**	**RES**	**Random**
MSU3	558	**671**	639	659	641	640	565
OLL	526	**634**	624	627	599	608	528
WBO	306	400	**536**	400	385	386	360
Hitman	420	403	**510**	406	425	420	440
RC2	530	620	**624**	618	600	597	541

MaxSAT Eval 2022:

- UWrMaxSat: 580; CASHWMaxSAT: 585, MaxCDCL: 593;
- MaxHS: 643, **EvalMaxSAT: 653**;

To conclude

- In this paper, we propose UpMax, a new framework that decouples the partition generation from the MaxSAT solving.

To conclude

- In this paper, we propose UpMax, a new framework that decouples the partition generation from the MaxSAT solving.

- UpMax allows users to specify how to **partition MaxSAT formulae based on their domain knowledge** with the `pwcnf` format.

To conclude

- In this paper, we propose UpMax, a new framework that decouples the partition generation from the MaxSAT solving.
- UpMax allows users to specify how to **partition MaxSAT formulae based on their domain knowledge** with the `pwcnf` format.
- Experimental results with two use cases with 5 algorithms (MSU3, WBO, OLL, RC2, Hitman), show that **partitioning can improve the performance of MaxSAT algorithms**.

To conclude

- In this paper, we propose UpMax, a new framework that decouples the partition generation from the MaxSAT solving.
- UpMax allows users to specify how to **partition MaxSAT formulae based on their domain knowledge** with the `pwcnf` format.
- Experimental results with two use cases with 5 algorithms (MSU3, WBO, OLL, RC2, Hitman), show that **partitioning can improve the performance of MaxSAT algorithms**.
- Check **Alloy$^{\text{Max}}$ [Zhang et al., 2021]** paper for UpMax's results on other application domains.

UpMax

GitHub

arXiv

UpMax

GitHub

arXiv

Thank you!

References

Ruben Martins and Vasco Manquinho and Inês Lynce (2012)
On Partitioning for Maximum Satisfiability.
ECAI 12.

Carlos Ansótegui and Maria Luisa Bonet and Joel Gabàs and Jordi Levy (2012)
Improving SAT-Based Weighted MaxSAT Solvers.
CP 12.

Ruben Martins and Vasco Manquinho and Inês Lynce (2014)
Open-WBO: a Modular MaxSAT Solver.
SAT 14.

Miguel Neves and Ruben Martins and Mikolás Janota and Inês Lynce and Vasco Manquinho (2015)
Exploiting Resolution-Based Representations for MaxSAT Solving.
SAT 15.

References (2)

Vasco Manquinho and Joao Marques-Silva and Jordi Planes (2009)
Algorithms for Weighted Boolean Optimization.
SAT 09.

Ruben Martins and Saurabh Joshi and Vasco Manquinho and Inês Lynce (2014)
Incremental Cardinality Constraints for MaxSAT.
CP 14.

António Morgado and Carmine Dodaro and João Marques-Silva (2014)
Core-Guided MaxSAT with Soft Cardinality Constraints.
CP 14.

Alexey Ignatiev and António Morgado and João Marques-Silva (2019)
RC2: an Efficient MaxSAT Solver.
SAT 19.

References (3)

Erick Moreno-Centeno and Richard M. Karp (2013)
The Implicit Hitting Set Approach to Solve Combinatorial Optimization Problems with an Application to Multigenome Alignment.
Oper. Res. 13.

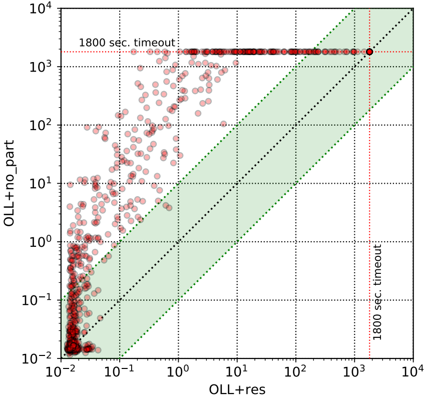
Alexey Ignatiev and António Morgado and João Marques-Silva (2018)
PySAT: A Python Toolkit for Prototyping with SAT Oracles.
SAT 18.

Aaron Stump and Geoff Sutcliffe and Cesare Tinelli (2014)
StarExec: A Cross-Community Infrastructure for Logic Solving.
IJCAR 14.

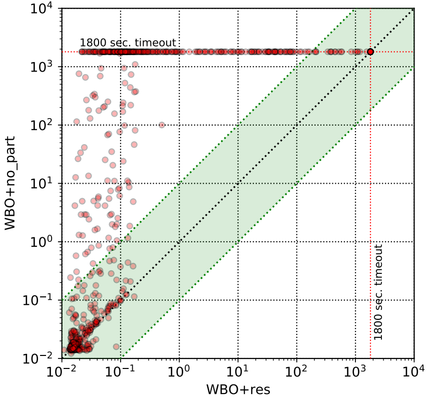
Changjian Zhang and Ryan Wagner and Pedro Orvalho and David Garlan and Vasco Manquinho and Ruben Martins and Eunsuk Kang (2021)
AlloyMax: bringing maximum satisfaction to relational specifications.
ESEC/FSE 21.

Appendix

MSC: Scatter plots



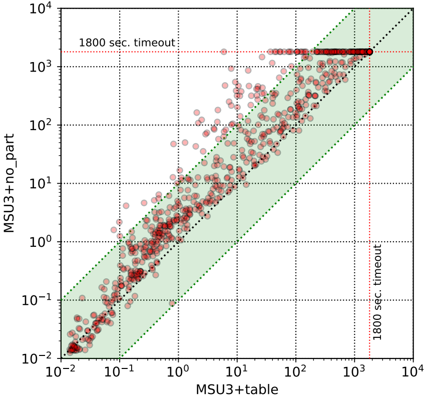
(a) MSC - OLL RES VS No Part.



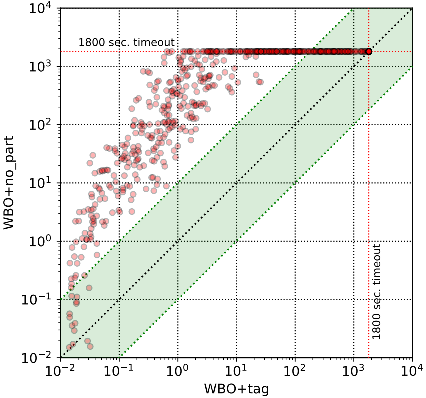
(b) MSC - WBO RES VS No Part.

Figure: Scatter plots comparing different MaxSAT algorithms and respective partitioning schemes for the Minimum Sum Coloring (MSC) problem.

SA: Scatter plots



(a) SA - MSU3 Table VS No Part.



(b) SA - WBO Tag VS No Part.

Figure: Scatter plots comparing different MaxSAT algorithms and respective partitioning schemes for the Seating Assignment (SA) problem.

MSC: Cactus plot

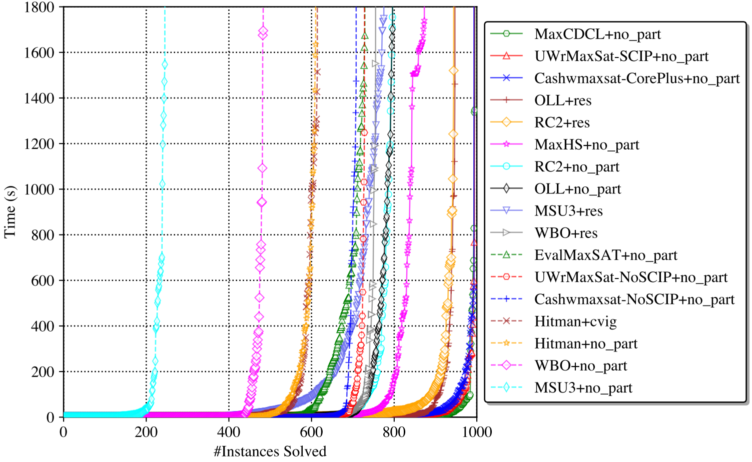


Figure: Cactus plot - The number of instances solved for the MSC problem for each MaxSAT algorithm and partitioning scheme.

SA: Cactus plot

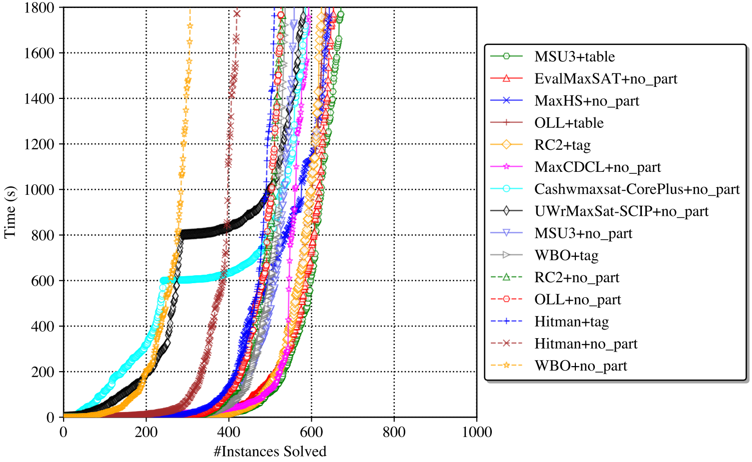
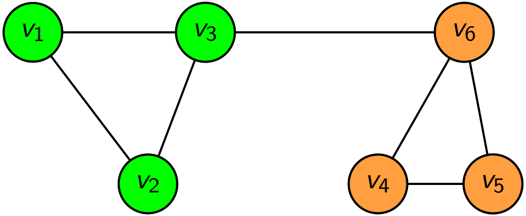


Figure: Cactus plot - The number of instances solved for the Seating Assignment problem for each MaxSAT algorithm and partitioning scheme.

Variable Incidence Graph (VIG)

Hard:	$h_1 : (v_1 \lor v_2)$	$h_2 : (\neg v_2 \lor v_3)$	$h_3 : (\neg v_1 \lor \neg v_3)$	$h_4 : (v_4 \lor v_5)$
	$h_5 : (\neg v_5 \lor v_6)$	$h_6 : (\neg v_4 \lor \neg v_6)$	$h_7 : (\neg v_3 \lor \neg v_6)$	
Soft:	$s_1 : (\neg v_1)$	$s_2 : (\neg v_3)$	$s_3 : (\neg v_4)$	$s_4 : (\neg v_6)$

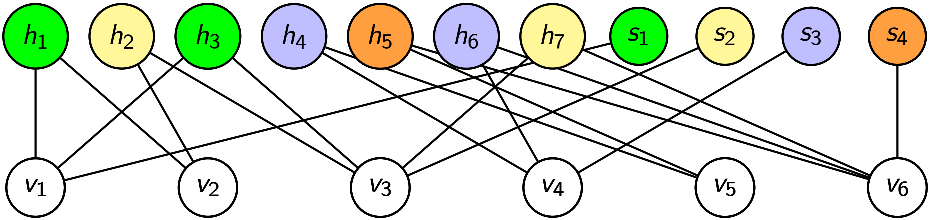


Clause-Variable Incidence Graph (CVIG)

Hard: $h_1 : (v_1 \lor v_2)$ $h_2 : (\neg v_2 \lor v_3)$ $h_3 : (\neg v_1 \lor \neg v_3)$ $h_4 : (v_4 \lor v_5)$

$h_5 : (\neg v_5 \lor v_6)$ $h_6 : (\neg v_4 \lor \neg v_6)$ $h_7 : (\neg v_3 \lor \neg v_6)$

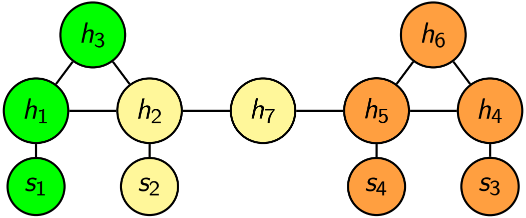
Soft: $s_1 : (\neg v_1)$ $s_2 : (\neg v_3)$ $s_3 : (\neg v_4)$ $s_4 : (\neg v_6)$



Resolution-based Graphs (RES)

Hard: $h_1 : (v_1 \vee v_2)$ $h_2 : (\neg v_2 \vee v_3)$ $h_3 : (\neg v_1 \vee \neg v_3)$ $h_4 : (v_4 \vee v_5)$
$h_5 : (\neg v_5 \vee v_6)$ $h_6 : (\neg v_4 \vee \neg v_6)$ $h_7 : (\neg v_3 \vee \neg v_6)$

Soft: $s_1 : (\neg v_1)$ $s_2 : (\neg v_3)$ $s_3 : (\neg v_4)$ $s_4 : (\neg v_6)$



The Boolean Satisfiability Problem

Given a CNF formula ϕ, the Satisfiability (SAT) problem corresponds to decide if there is an assignment such that ϕ is satisfied or prove that no such assignment exists.

$$\phi: \quad c_1 : (v_1 \vee v_2) \qquad c_2 : (\neg v_2 \vee v_3) \qquad c_3 : (\neg v_1 \vee \neg v_3)$$

The Boolean Satisfiability Problem

Given a CNF formula ϕ, the Satisfiability (SAT) problem corresponds to decide if there is an assignment such that ϕ is satisfied or prove that no such assignment exists.

ϕ :	$c_1 : (v_1 \vee v_2)$	$c_2 : (\neg v_2 \vee v_3)$	$c_3 : (\neg v_1 \vee \neg v_3)$
Solution :	v_1	$\neg v_2$	$\neg v_3$

The Maximum Satisfiability Problem

- The Maximum Satisfiability (MaxSAT) is an optimization version of the SAT problem.

The Maximum Satisfiability Problem

- The Maximum Satisfiability (MaxSAT) is an optimization version of the SAT problem.
- In partial MaxSAT, clauses in ϕ are split in hard and soft clauses.

Hard:	$h_1 : (v_1 \vee v_2)$	$h_2 : (\neg v_2 \vee v_3)$	$h_3 : (\neg v_1 \vee \neg v_3)$
Soft:	$s_1 : (\neg v_1)$	$s_2 : (\neg v_3)$	

The Maximum Satisfiability Problem

- The Maximum Satisfiability (MaxSAT) is an optimization version of the SAT problem.
- In partial MaxSAT, clauses in ϕ are split in hard and soft clauses.
- Given a formula ϕ, the goal is to find an assignment that satisfies all hard clauses while minimizing the number of unsatisfied soft clauses.

Hard:	$h_1 : (v_1 \vee v_2)$	$h_2 : (\neg v_2 \vee v_3)$	$h_3 : (\neg v_1 \vee \neg v_3)$
Soft:	$s_1 : (\neg v_1)$	$s_2 : (\neg v_3)$	
Solution :	v_1	$\neg v_2$	$\neg v_3$

$s_1 \vee s_2 \longrightarrow \text{Cost} = 1$