On Dedicated CDCL Strategies for PB Solvers

Daniel Le Berre[1], Romain Wallon[2]

SAT 2021 – July 8th, 2021

[1] CRIL, Univ Artois & CNRS
[2] LIX (Laboratoire d'Informatique de l'X), Ecole Polytechnique, X-Uber Chair

SAT Solving and Limitations

Modern SAT solvers, based on the CDCL architecture (GRASP) and efficient heuristics and data structures (Chaff, MiniSat), are very efficient in practice

SAT Solving and Limitations

Modern SAT solvers, based on the CDCL architecture (GRASP) and efficient heuristics and data structures (Chaff, MiniSat), are very efficient in practice

However, some instances remain completely out of reach for these solvers, due to the weakness of the resolution proof system they use internally

SAT Solving and Limitations

Modern SAT solvers, based on the CDCL architecture (GRASP) and efficient heuristics and data structures (Chaff, MiniSat), are very efficient in practice

However, some instances remain completely out of reach for these solvers, due to the weakness of the resolution proof system they use internally

This is particularly true for instances requiring the ability to count, such as pigeonhole-principle formulae, stating that *"n pigeons do not fit in* $n-1$ *holes"*

SAT Solving and Limitations

Modern SAT solvers, based on the CDCL architecture (GRASP) and efficient heuristics and data structures (Chaff, MiniSat), are very efficient in practice

However, some instances remain completely out of reach for these solvers, due to the weakness of the resolution proof system they use internally

This is particularly true for instances requiring the ability to count, such as pigeonhole-principle formulae, stating that *"n pigeons do not fit in $n-1$ holes"*

While modern SAT solvers perform poorly on such instances for $n > 20$, PB solvers based on cutting-planes may solve them in linear time

Pseudo-Boolean (PB) Constraints

PB solvers generalize SAT solvers to take into account

- normalized PB constraints $\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta$
- cardinality constraints $\sum_{i=1}^{n} \ell_i \geq \delta$
- clauses $\sum_{i=1}^{n} \ell_i \geq 1 \equiv \bigvee_{i=1}^{n} \ell_i$

in which

- the coefficients α_i are non-negative integers
- ℓ_i are literals, i.e., a variable v or its negation $\bar{v} = 1 - v$
- the degree δ is a non-negative integer

CDCL in PB Solvers

Modern SAT solvers are very efficient in practice, especially because of the conflict-driven clause learning architecture

CDCL in PB Solvers

Modern SAT solvers are very efficient in practice, especially because of the conflict-driven clause learning architecture

Current PB solvers also implement conflict analysis, based on the cutting-planes proof system

CDCL in PB Solvers

Modern SAT solvers are very efficient in practice, especially because of the conflict-driven clause learning architecture

Current PB solvers also implement conflict analysis, based on the cutting-planes proof system

It is well known that, in addition to conflict analysis, several features are crucial for solving problems efficiently, such as:

- branching heuristic
- learned constraint deletion strategy
- restart policy

CDCL in PB Solvers

Modern SAT solvers are very efficient in practice, especially because of the conflict-driven clause learning architecture

Current PB solvers also implement conflict analysis, based on the cutting-planes proof system

It is well known that, in addition to conflict analysis, several features are crucial for solving problems efficiently, such as:

- branching heuristic
- learned constraint deletion strategy
- restart policy

These features are mostly reused as they are by current PB solvers, without taking into account the particular properties of PB constraints

Analyzing Conflicts: Generalized Resolution

The generalized resolution proof system [Hooker, 1988] is used in PB solvers as the counterpart of the resolution proof system:

$$\frac{\alpha\ell + \sum_{i=1}^{n} \alpha_i\ell_i \geq \delta_1 \qquad \beta\overline{\ell} + \sum_{i=1}^{n} \beta_i\ell_i \geq \delta_2}{\sum_{i=1}^{n}(\beta\alpha_i + \alpha\beta_i)\ell_i \geq \beta\delta_1 + \alpha\delta_2 - \alpha\beta} \text{ (cancellation)}$$

$$\frac{\sum_{i=1}^{n} \alpha_i\ell_i \geq \delta}{\sum_{i=1}^{n} \min(\alpha_i, \delta)\ell_i \geq \delta} \text{ (saturation)}$$

Analyzing Conflicts: Generalized Resolution

The generalized resolution proof system [Hooker, 1988] is used in PB solvers as the counterpart of the resolution proof system:

$$\frac{\alpha\ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta_1 \qquad \beta\overline{\ell} + \sum_{i=1}^{n} \beta_i \ell_i \geq \delta_2}{\sum_{i=1}^{n} (\beta\alpha_i + \alpha\beta_i)\ell_i \geq \beta\delta_1 + \alpha\delta_2 - \alpha\beta} \text{ (cancellation)}$$

$$\frac{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta}{\sum_{i=1}^{n} \min(\alpha_i, \delta)\ell_i \geq \delta} \text{ (saturation)}$$

Analyzing Conflicts: Generalized Resolution

The generalized resolution proof system [Hooker, 1988] is used in PB solvers as the counterpart of the resolution proof system:

$$\frac{\alpha\ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta_1 \qquad \beta\bar{\ell} + \sum_{i=1}^{n} \beta_i \ell_i \geq \delta_2}{\sum_{i=1}^{n} (\beta\alpha_i + \alpha\beta_i)\ell_i \geq \beta\delta_1 + \alpha\delta_2 - \alpha\beta} \text{ (cancellation)}$$

$$\frac{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta}{\sum_{i=1}^{n} \min(\alpha_i, \delta)\ell_i \geq \delta} \text{ (saturation)}$$

Analyzing Conflicts: Generalized Resolution

The generalized resolution proof system [Hooker, 1988] is used in PB solvers as the counterpart of the resolution proof system:

$$\frac{\alpha\ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta_1 \qquad \beta\overline{\ell} + \sum_{i=1}^{n} \beta_i \ell_i \geq \delta_2}{\sum_{i=1}^{n} (\beta\alpha_i + \alpha\beta_i)\ell_i \geq \beta\delta_1 + \alpha\delta_2 - \alpha\beta} \text{ (cancellation)}$$

$$\frac{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta}{\sum_{i=1}^{n} \min(\alpha_i, \delta)\ell_i \geq \delta} \text{ (saturation)}$$

Analyzing Conflicts: Generalized Resolution

The generalized resolution proof system [Hooker, 1988] is used in PB solvers as the counterpart of the resolution proof system:

$$\frac{\alpha\ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta_1 \qquad \beta\bar{\ell} + \sum_{i=1}^{n} \beta_i \ell_i \geq \delta_2}{\sum_{i=1}^{n} (\beta\alpha_i + \alpha\beta_i)\ell_i \geq \beta\delta_1 + \alpha\delta_2 - \alpha\beta} \text{ (cancellation)}$$

$$\frac{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta}{\sum_{i=1}^{n} \min(\alpha_i, \delta)\ell_i \geq \delta} \text{ (saturation)}$$

Analyzing Conflicts: Generalized Resolution

The generalized resolution proof system [Hooker, 1988] is used in PB solvers as the counterpart of the resolution proof system:

$$\frac{\alpha\ell + \sum_{i=1}^{n} \alpha_i \ell_i \geq \delta_1 \qquad \beta\bar{\ell} + \sum_{i=1}^{n} \beta_i \ell_i \geq \delta_2}{\sum_{i=1}^{n} (\beta\alpha_i + \alpha\beta_i)\ell_i \geq \beta\delta_1 + \alpha\delta_2 - \alpha\beta} \text{ (cancellation)}$$

$$\frac{\sum_{i=1}^{n} \alpha_i \ell_i \geq \delta}{\sum_{i=1}^{n} \min(\alpha_i, \delta)\ell_i \geq \delta} \text{ (saturation)}$$

These two rules are used during conflict analysis
to learn new constraints

Analyzing Conflicts: Example

Suppose that we have the following constraints:

$$3\bar{a}(?@?) + 3\bar{f}(?@?) + d(?@?) + e(?@?) + g(?@?) \geq 5$$

$$6a(?@?) + 3b(?@?) + 3c(?@?) + 3\bar{d}(?@?) + 3f(?@?) \geq 9$$

Analyzing Conflicts: Example

Suppose that we have the following constraints:

$$3\bar{a}(?@?) + 3\bar{f}(?@?) + d(?@?) + e(?@?) + g(?@?) \geq 5$$

$$6a(?@?) + 3b(1@1) + 3c(?@?) + 3\bar{d}(?@?) + 3f(?@?) \geq 9$$

Analyzing Conflicts: Example

Suppose that we have the following constraints:

$$3\bar{a}(?@?) + 3\bar{f}(?@?) + d(?@?) + e(?@?) + g(?@?) \geq 5$$

$$6a(?@?) + 3b(1@1) + 3c(0@2) + 3\bar{d}(?@?) + 3f(?@?) \geq 9$$

Analyzing Conflicts: Example

Suppose that we have the following constraints:

$$3\bar{a}(?@?) + 3\bar{f}(?@?) + d(?@?) + e(?@?) + g(0@3) \geq 5$$

$$6a(?@?) + 3b(1@1) + 3c(0@2) + 3\bar{d}(?@?) + 3f(?@?) \geq 9$$

Analyzing Conflicts: Example

Suppose that we have the following constraints:

$$3\bar{a}(?@?) + 3\bar{f}(?@?) + d(0@4) + e(?@?) + g(0@3) \geq 5$$

$$6a(?@?) + 3b(1@1) + 3c(0@2) + 3\bar{d}(1@4) + 3f(?@?) \geq 9$$

Analyzing Conflicts: Example

Suppose that we have the following constraints:

$$3\bar{a}(1@4) + 3\bar{f}(1@4) + d(0@4) + e(?@?) + g(0@3) \geq 5$$

$$6a(0@4) + 3b(1@1) + 3c(0@2) + 3\bar{d}(1@4) + 3f(0@4) \geq 9$$

Analyzing Conflicts: Example

Suppose that we have the following constraints:

$$3\bar{a}(1@4) + 3\bar{f}(1@4) + d(0@4) + e(?@?) + g(0@3) \geq 5$$

$$6a(0@4) + 3b(1@1) + 3c(0@2) + 3\bar{d}(1@4) + 3f(0@4) \geq 9$$

We now apply the cancellation rule between these two constraints:

$$\frac{3\bar{a} + 3\bar{f} + d + e + g \geq 5 \qquad 6a + 3b + 3c + 3\bar{d} + 3f \geq 9}{3a(0@4) + 3b(1@1) + 3c(0@2) + 2\bar{d}(1@4) + e(?@?) + g(0@3) \geq 7}$$

Analyzing Conflicts: Example

Suppose that we have the following constraints:

$$3\bar{a}(1@4) + 3\bar{f}(1@4) + d(0@4) + e(?@?) + g(0@3) \geq 5$$

$$6a(0@4) + 3b(1@1) + 3c(0@2) + 3\bar{d}(1@4) + 3f(0@4) \geq 9$$

We now apply the cancellation rule between these two constraints:

$$\frac{3\bar{a} + 3\bar{f} + d + e + g \geq 5 \qquad 6a + 3b + 3c + 3\bar{d} + 3f \geq 9}{3a(?@?) + 3b(1@1) + 3c(0@2) + 2\bar{d}(?@?) + e(?@?) + g(0@3) \geq 7}$$

Analyzing Conflicts: Example

Suppose that we have the following constraints:

$$3\bar{a}(1@4) + 3\bar{f}(1@4) + d(0@4) + e(?@?) + g(0@3) \geq 5$$

$$6a(0@4) + 3b(1@1) + 3c(0@2) + 3\bar{d}(1@4) + 3f(0@4) \geq 9$$

We now apply the cancellation rule between these two constraints:

$$\frac{3\bar{a} + 3\bar{f} + d + e + g \geq 5 \qquad 6a + 3b + 3c + 3\bar{d} + 3f \geq 9}{3a(?@?) + 3b(1@1) + 3c(0@2) + 2\bar{d}(?@?) + e(?@?) + g(0@3) \geq 7}$$

The PB constraints involved in this conflict analysis have very different properties compared to clauses!

Branching Heuristics: Classical Implementation

SAT and PB solvers use the EVSIDS heuristic for choosing the next variable to branch on

Branching Heuristics: Classical Implementation

SAT and PB solvers use the EVSIDS heuristic for choosing the next variable to branch on

In this heuristic, all variables encountered during conflict analysis are bumped

Branching Heuristics: Classical Implementation

SAT and PB solvers use the EVSIDS heuristic for choosing the next variable to branch on

In this heuristic, all variables encountered during conflict analysis are bumped

This is the case for all the variables appearing in the previous reason:

$$3\bar{a} + 3\bar{f} + d + e + g \geq 5$$

Branching Heuristics: Classical Implementation

SAT and PB solvers use the EVSIDS heuristic for choosing the next variable to branch on

In this heuristic, all variables encountered during conflict analysis are bumped

This is the case for all the variables appearing in the previous reason:

$$3\bar{a} + 3\bar{f} + d + e + g \geq 5$$

This means that the scores of the variables a, f, d, e and g are incremented

Branching Heuristics: Coefficients

An obvious difference between clauses and PB constraints is the presence of coefficients in the constraint

$$3\bar{a} + 3\bar{f} + d + e + g \geq 5$$

Branching Heuristics: Coefficients

An obvious difference between clauses and PB constraints is the presence of coefficients in the constraint

$$3\bar{a} + 3\bar{f} + d + e + g \geq 5$$

Because of these coefficients, literals are not symmetrical in the constraint

Branching Heuristics: Coefficients

An obvious difference between clauses and PB constraints is the presence of coefficients in the constraint

$$3\bar{a} + 3\bar{f} + d + e + g \geq 5$$

Because of these coefficients, literals are not symmetrical in the constraint

A possible way to adapt VSIDS is to increment the score of the variables proportionately w.r.t. these coefficients

Branching Heuristics: Assignment

Observe that some literals are unassigned in the reason for $\overline{f}$

$$3\overline{a} + 3\overline{f} + d + e + g \geq 5$$

Branching Heuristics: Assignment

Observe that some literals are unassigned in the reason for $\bar{f}$

$$3\bar{a} + 3\bar{f} + d + e + g \geq 5$$

In an assertive clause, all literals are assigned, and all but one are falsified: these latter literals are those involved in the propagation

Branching Heuristics: Assignment

Observe that some literals are unassigned in the reason for $\overline{f}$

$$3\overline{a} + 3\overline{f} + d + e + g \geq 5$$

In an assertive clause, all literals are assigned, and all but one are falsified: these latter literals are those involved in the propagation

If we weaken away (i.e., assign to 1 and simplify) the literal e, the constraint still propagates $\overline{f}$ $\quad\rightsquigarrow\quad$ $3\overline{a} + 3\overline{f} + d + g \geq 4$

Branching Heuristics: Assignment

Observe that some literals are unassigned in the reason for $\overline{f}$

$$3\overline{a} + 3\overline{f} + d + e + g \geq 5$$

> *In an assertive clause, all literals are assigned, and all but one are falsified: these latter literals are those involved in the propagation*

If we weaken away (i.e., assign to 1 and simplify) the literal e, the constraint still propagates $\overline{f}$ $\rightsquigarrow$ $3\overline{a} + 3\overline{f} + d + g \geq 4$

This is also true if we also weaken away the literal a $\rightsquigarrow$ $3\overline{f} + d + g \geq 1$

Branching Heuristics: Assignment

Observe that some literals are unassigned in the reason for $\bar{f}$

$$3\bar{a} + 3\bar{f} + d + e + g \geq 5$$

In an assertive clause, all literals are assigned, and all but one are falsified: these latter literals are those involved in the propagation

If we weaken away (i.e., assign to 1 and simplify) the literal e, the constraint still propagates $\bar{f}$ $\rightsquigarrow$ $3\bar{a} + 3\bar{f} + d + g \geq 4$

This is also true if we also weaken away the literal a $\rightsquigarrow$ $3\bar{f} + d + g \geq 1$

Actually, the literals that should be bumped are those of this constraint!

Branching Heuristics: Experiments in Sat4j[1]

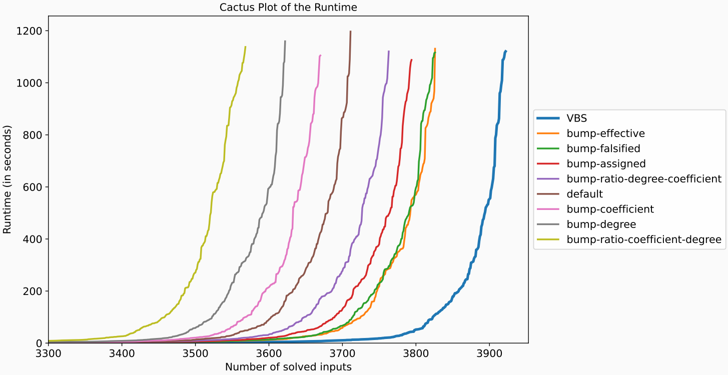

Figure 1: Performance of different bumping strategies on decision problems

[1] More at `https://gitlab.com/pb-cdcl-strategies/experiments`

Quality of Learned Constraints: Classical Implementations

In SAT solvers, evaluating the quality of learned constraints is used to choose which constraints should be deleted and to decide when a restart should be triggered

Quality of Learned Constraints: Classical Implementations

In SAT solvers, evaluating the quality of learned constraints is used to choose which constraints should be deleted and to decide when a restart should be triggered

The quality measures used by SAT solvers do not take into account the properties of PB constraints

Quality of Learned Constraints: Classical Implementations

In SAT solvers, evaluating the quality of learned constraints is used to choose which constraints should be deleted and to decide when a restart should be triggered

The quality measures used by SAT solvers do not take into account the properties of PB constraints

Adapting quality measures to PB constraints may be used to design learned constraint deletion strategies and restart policies dedicated to PB problems

Quality of Learned Constraints: Size and Coefficients

In SAT solvers, the size of a clause may be used as a measure of its quality: the longer the clause, the lower its strength

Quality of Learned Constraints: Size and Coefficients

In SAT solvers, the size of a clause may be used as a measure of its quality: the longer the clause, the lower its strength

In the PB case, the length of a constraint does not reflect its strength

Quality of Learned Constraints: Size and Coefficients

In SAT solvers, the size of a clause may be used as a measure of its quality: the longer the clause, the lower its strength

In the PB case, the length of a constraint does not reflect its strength

The size of a PB constraint also takes into account its coefficients

$$3a(?@?) + 3b(1@1) + 3c(0@2) + 2\bar{d}(?@?) + e(?@?) + g(0@3) \geq 7$$

Quality of Learned Constraints: Size and Coefficients

In SAT solvers, the size of a clause may be used as a measure of its quality: the longer the clause, the lower its strength

In the PB case, the length of a constraint does not reflect its strength

The size of a PB constraint also takes into account its coefficients

$$3a(?@?) + 3b(1@1) + 3c(0@2) + 2\bar{d}(?@?) + e(?@?) + g(0@3) \geq 7$$

In practice, the coefficients may become very big, which requires the use of arbitrary precision encodings and slows down arithmetic operations

Quality of Learned Constraints: Size and Coefficients

In SAT solvers, the size of a clause may be used as a measure of its quality: the longer the clause, the lower its strength

In the PB case, the length of a constraint does not reflect its strength

The size of a PB constraint also takes into account its coefficients

$$3a(?@?) + 3b(1@1) + 3c(0@2) + 2\bar{d}(?@?) + e(?@?) + g(0@3) \geq 7$$

In practice, the coefficients may become very big, which requires the use of arbitrary precision encodings and slows down arithmetic operations

We consider a quality measure based on the degree of the constraints: the lower the degree, the better the constraint

Quality of Learned Constraint: Assignment (LBD)

In SAT solvers, the Literal Block Distance (LBD) measures the quality of clauses by the number of decision levels appearing in this clause

$$3a(?@?) + 3b(1@1) + 3c(0@2) + 2\bar{d}(?@?) + e(?@?) + g(0@3) \geq 7$$

Quality of Learned Constraint: Assignment (LBD)

In SAT solvers, the Literal Block Distance (LBD) measures the quality of clauses by the number of decision levels appearing in this clause

$$3a(?@?) + 3b(1@1) + 3c(0@2) + 2\bar{d}(?@?) + e(?@?) + g(0@3) \geq 7$$

There are satisfied and unassigned literals in this constraint!

Quality of Learned Constraint: Assignment (LBD)

In SAT solvers, the Literal Block Distance (LBD) measures the quality of clauses by the number of decision levels appearing in this clause

$$3a(?@?) + 3b(1@1) + 3c(0@2) + 2\bar{d}(?@?) + e(?@?) + g(0@3) \geq 7$$

There are satisfied and unassigned literals in this constraint!

As for bumping strategies, the computation of the LBD should take into account these literals to be more accurate

Quality of Learned Constraint: Experiments in Sat4j[2]

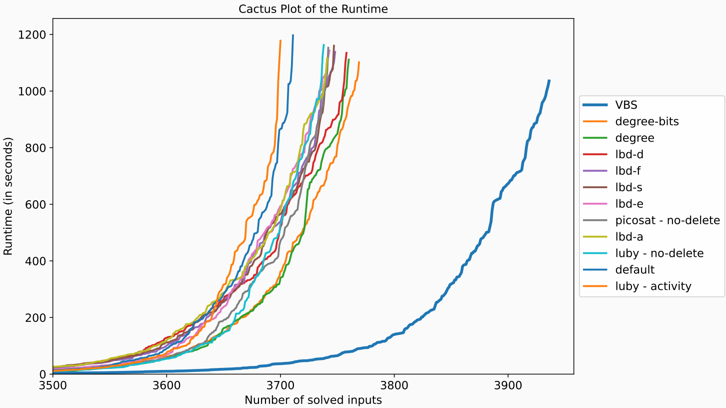

Figure 2: Performance of different learned constraint deletion and restart strategies on decision problems

[2] More at `https://gitlab.com/pb-cdcl-strategies/experiments`

Experiments: Description

Experimental Setup

- Intel XEON X5550 (2.66 GHz, 8 MB cache)
- Time limited to 1200 seconds
- Memory limited to 32 GB

Instances

- Decision problems of all PB competitions (small integers)
- Optimization problems of all PB competitions (small integers)

Solvers

- Different configurations of *Sat4j*
  - *Sat4j-GeneralizedResolution*
  - *Sat4j-RoundingSat*
  - *Sat4j-PartialRoundingSat*
- *RoundingSat*

Experiments: Decision Problems[3]

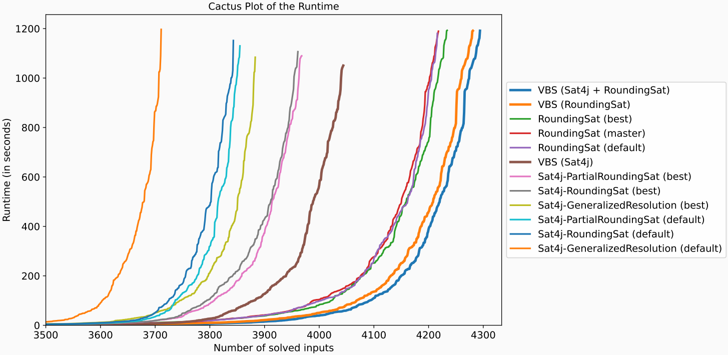

Figure 3: Performance of different PB Solvers on decision problems

[3]More at `https://gitlab.com/pb-cdcl-strategies/experiments`

Experiments: Optimization Problems[4]

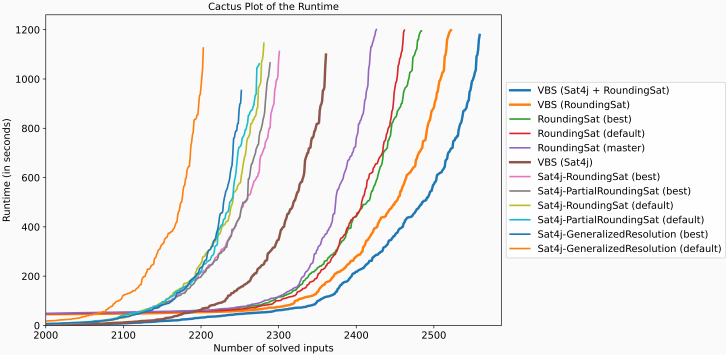

Figure 4: Performance of different PB solvers on optimization problems

[4]More at `https://gitlab.com/pb-cdcl-strategies/experiments`

Conclusion and Perspectives

Conclusion

- PB solvers implement CDCL strategies mostly “as they are” from their original definition in SAT solvers
- However, PB solvers should also take into account the particular form of PB constraints
- Considering coefficients and assignments improves the performance of PB solvers (in our case, *Sat4j* and *RoundingSat*)

Conclusion and Perspectives

Conclusion

- PB solvers implement CDCL strategies mostly "as they are" from their original definition in SAT solvers
- However, PB solvers should also take into account the particular form of PB constraints
- Considering coefficients and assignments improves the performance of PB solvers (in our case, *Sat4j* and *RoundingSat*)

Perspectives

- Find new ways to adapt CDCL strategies
- Find better combinations of the proposed extensions
- Dynamically configure the best strategies for a given instance

On Dedicated CDCL Strategies for PB Solvers

Daniel Le Berre[1], Romain Wallon[2]

SAT 2021 – July 8th, 2021

[1] CRIL, Univ Artois & CNRS
[2] LIX (Laboratoire d'Informatique de l'X), Ecole Polytechnique, X-Uber Chair