Designing New Phase selection Heuristics

Arijit Shaw • Kuldeep S. Meel

Conflict Driven Clause Learning (CDCL)

CDCL(F)

```
A ← {}
while hasUnassignedVars(F)
  A ← A ∪ { DECIDE(F,A) }
  while BCP(F,A) = conflict
    ⟨b, c⟩ ← ANALYZECONFLICT()
    if b < 0 then return unsat
    else BACKTRACK(F,A, b)
    if clauseDeletionRequired()
      then reduceDB()
    if restartRequired()
      then restart()
return sat
```

Conflict Driven Clause Learning (CDCL)

CDCL(F)

```
A ← {}
while hasUnassignedVars(F)
  A ← A ∪ { DECIDE(F,A) }
  while BCP(F,A) = conflict
    ⟨b, c⟩ ← ANALYZECONFLICT()
    if b < 0 then return unsat
    else BACKTRACK(F,A, b)
    if clauseDeletionRequired()
      then reduceDB()
    if restartRequired()
      then restart()
return sat
```

Conflict Driven Clause Learning (CDCL)

CDCL(F)

```
A ← {}
while hasUnassignedVars(F)
  A ← A ∪ { DECIDE(F,A) }
  while BCP(F,A) = conflict
    ⟨b, c⟩ ← ANALYZECONFLICT()
    if b < 0 then return unsat
    else BACKTRACK(F,A, b)
    if clauseDeletionRequired()
      then reduceDB()
    if restartRequired()
      then restart()
return sat
```

Decision Heuristics

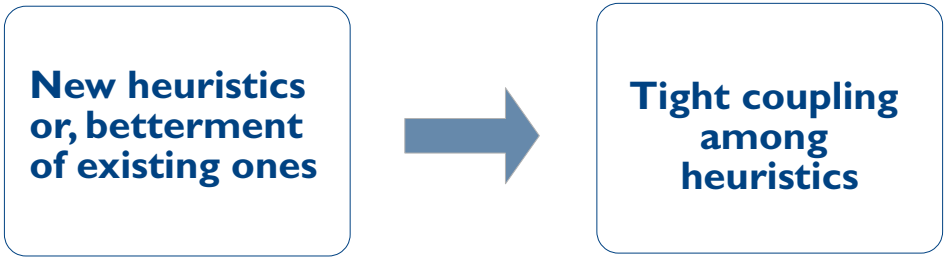
Clause Maintainance Heuristics

Restart Heuristics

Development of Solvers

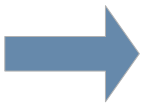
New heuristics or, betterment of existing ones

Development of Solvers



Development of Solvers

New heuristics or, betterment of existing ones



Tight coupling among heuristics

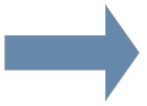
unclear whether VMTF only works in combination with Glucose restarts

Evaluating CDCL Variable Scoring Schemes (SAT'15)

Armin Biere and Andreas Fröhlich

Development of Solvers

New heuristics or, betterment of existing ones



Tight coupling among heuristics

unclear whether VMTF only works in combination with Glucose restarts

Evaluating CDCL Variable Scoring Schemes (SAT'15)

Armin Biere and Andreas Fröhlich

Between SAT and UNSAT: The Fundamental Difference in CDCL SAT (SAT'15)

Chanseok Oh

we decided to use the (VSIDS decay) factor of 0.999 in the no restart phase. For the Glucose (restart) phase, however, we retained the default value of 0.95

Heuristics under study

CDCL(F)

```
A ← {}
while hasUnassignedVars(F)
  A ← A ∪ { DECIDE(F,A) }
  while BCP(F,A) = conflict
    ⟨b, c⟩ ← ANALYZECONFLICT()
    if b < 0 then return unsat
    else BACKTRACK(F,A, b)
  if clauseDeletionRequired()
    then reduceDB()
  if restartRequired()
    then restart()
return sat
```

Heuristics under study

CDCL(F)

```
A ← {}
while hasUnassignedVars(F)
  A ← A ∪ { DECIDE(F,A) }
  while BCP(F,A) = conflict
    ⟨b, c⟩ ← ANALYZECONFLICT()
    if b < 0 then return unsat
    else BACKTRACK(F,A, b)
  if clauseDeletionRequired()
    then reduceDB()
  if restartRequired()
    then restart()
return sat
```

phase selection heuristic

backtracking strategy

Heuristics under study

CDCL(F)

```
A ← {}
while hasUnassignedVars(F)
  A ← A ∪ { DECIDE(F,A) }
  while BCP(F,A) = conflict
    ⟨b, c⟩ ← ANALYZECONFLICT()
    if b < 0 then return unsat
    else BACKTRACK(F,A, b)
  if clauseDeletionRequired()
    then reduceDB()
  if restartRequired()
    then restart()
return sat
```

phase selection heuristic

Phase Saving

backtracking strategy

Chronological Backtracking

Heuristics under study

CDCL(F)

```
A ← {}
while hasUnassignedVars(F)
  A ← A ∪ { DECIDE(F,A) }
  while BCP(F,A) = conflict
    ⟨b, c⟩ ← ANALYZECONFLICT()
    if b < 0 then return unsat
    else BACKTRACK(F,A, b)
  if clauseDeletionRequired()
    then reduceDB()
  if restartRequired()
    then restart()
return sat
```

phase selection heuristic

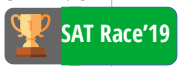
Phase Saving

backtracking strategy

Chronological Backtracking

Who uses?

1) Maple_LCM_Dist_ChronoBT
2) Maple_LCM_Dist_ChronoBTv3



3) CryptoMiniSat

Heuristics under study

CDCL(F)

```
A ← {}
while hasUnassignedVars(F)
  A ← A ∪ { DECIDE(F,A) }
  while BCP(F,A) = conflict
    ⟨b, c⟩ ← ANALYZECONFLICT()
    if b < 0 then return unsat
    else BACKTRACK(F,A, b)
  if clauseDeletionRequired()
    then reduceDB()
  if restartRequired()
    then restart()
return sat
```

phase selection heuristic

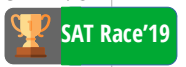
Phase Saving

backtracking strategy

Chronological Backtracking

Who uses?

1) Maple_LCM_Dist_ChronoBT
2) Maple_LCM_Dist_ChronoBTv3



3) CryptoMiniSat

Phase-Saving

a phase selection heuristic

Phase-Saving

a phase selection heuristic

A Lightweight Component Caching Scheme For Satisfiability Solvers (SAT'07)

Knot Pipatsrisawat and Adnan Darwiche

Phase-Saving

a phase selection heuristic

A Lightweight Component Caching Scheme For Satisfiability Solvers (SAT'07)

Knot Pipatsrisawat and Adnan Darwiche

x1	x2	x3	x4	x5
⊥	⊥	⊤	⊥	⊤

Data Strucure

Phase-Saving

a phase selection heuristic

A Lightweight Component Caching Scheme For Satisfiability Solvers (SAT'07)

Knot Pipatsrisawat and Adnan Darwiche

x1	x2	x3	x4	x5
⊥	⊥	⊤	⊥	⊤

SavedPhase(v) = assignment(v)

Data Strucure

Update during backtrack

Phase-Saving

a phase selection heuristic

A Lightweight Component Caching Scheme For Satisfiability Solvers (SAT'07)

Knot Pipatsrisawat and Adnan Darwiche

x1	x2	x3	x4	x5
⊥	⊥	⊤	⊥	⊤

Data Strucure

SavedPhase(v) = assignment(v)

Update during backtrack

return SavedPhase(v)

Phase Selection

Phase-Saving

a phase selection heuristic

A Lightweight Component Caching Scheme For Satisfiability Solvers (SAT'07)

Knot Pipatsrisawat and Adnan Darwiche

x1	x2	x3	x4	x5
⊥	⊥	⊤	⊥	⊤

Data Strucure

SavedPhase(v) = assignment(v)

Update during backtrack

return SavedPhase(v)

Phase Selection

on SAT '19 instances

solver	# solved	Avg. Runtime*
MLDC	237	4556
MLDC_random_phase	222	4785

*** MLDC :** Maple_LCM_Dist_ChronoBT_v3

Phase-Saving

a phase selection heuristic

A Lightweight Component Caching Scheme For Satisfiability Solvers (SAT'07)

Knot Pipatsrisawat and Adnan Darwiche

on SAT '19 instances

solver	# solved	Avg. Runtime*
MLDC	237	4556
MLDC_random_phase	222	4785

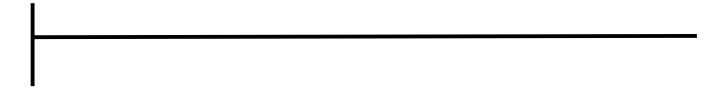
*** MLDC :** Maple_LCM_Dist_ChronoBT_v3

x1	x2	x3	x4	x5
⊥	⊥	⊤	⊥	⊤

Data Strucure

SavedPhase(v) = assignment(v)

Update during backtrack



return SavedPhase(v)

Phase Selection

Phase-Saving

a phase selection heuristic

A Lightweight Component Caching Scheme For Satisfiability Solvers (SAT'07)

Knot Pipatsrisawat and Adnan Darwiche

x1	x2	x3	x4	x5
⊥	⊥	⊤	⊥	⊤

Data Strucure

SavedPhase(v) = assignment(v)

Update during backtrack

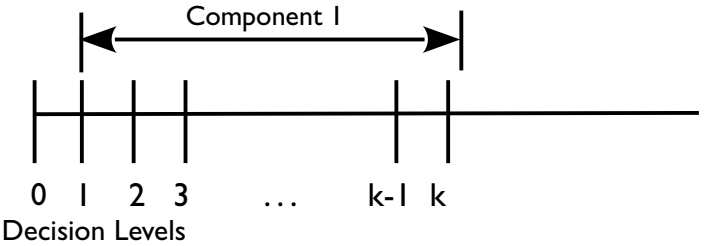
return SavedPhase(v)

Phase Selection

on SAT '19 instances

solver	# solved	Avg. Runtime*
MLDC	237	4556
MLDC_random_phase	222	4785

*** MLDC :** Maple_LCM_Dist_ChronoBT_v3



Phase-Saving

a phase selection heuristic

A Lightweight Component Caching Scheme For Satisfiability Solvers (SAT'07)

Knot Pipatsrisawat and Adnan Darwiche

x1	x2	x3	x4	x5
⊥	⊥	⊤	⊥	⊤

Data Strucure

SavedPhase(v) = assignment(v)

Update during backtrack

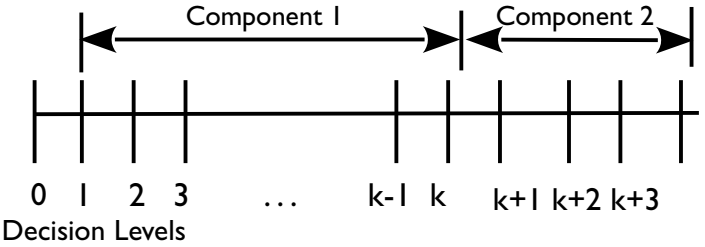
return SavedPhase(v)

Phase Selection

on SAT '19 instances

solver	# solved	Avg. Runtime*
MLDC	237	4556
MLDC_random_phase	222	4785

*** MLDC :** Maple_LCM_Dist_ChronoBT_v3



Phase-Saving

a phase selection heuristic

A Lightweight Component Caching Scheme For Satisfiability Solvers (SAT'07)

Knot Pipatsrisawat and Adnan Darwiche

x1	x2	x3	x4	x5
⊥	⊥	⊤	⊥	⊤

Data Strucure

SavedPhase(v) = assignment(v)

Update during backtrack

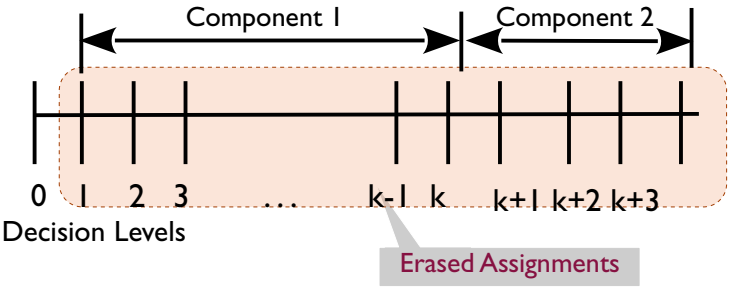
return SavedPhase(v)

Phase Selection

on SAT '19 instances

solver	# solved	Avg. Runtime*
MLDC	237	4556
MLDC_random_phase	222	4785

*** MLDC :** Maple_LCM_Dist_ChronoBT_v3



Phase-Saving

a phase selection heuristic

A Lightweight Component Caching Scheme For Satisfiability Solvers (SAT'07)

Knot Pipatsrisawat and Adnan Darwiche

x1	x2	x3	x4	x5
⊥	⊥	⊤	⊥	⊤

Data Strucure

SavedPhase(v) = assignment(v)

Update during backtrack

return SavedPhase(v)

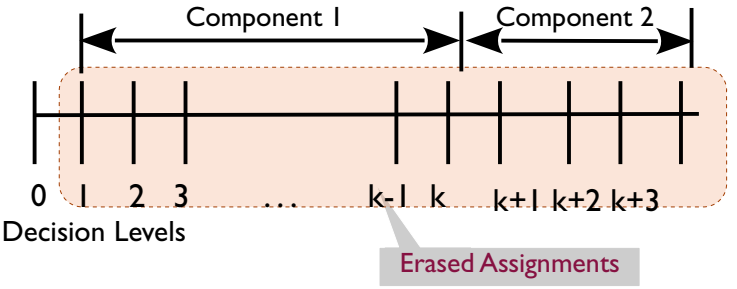
Phase Selection

on SAT '19 instances

solver	# solved	Avg. Runtime*
MLDC	237	4556
MLDC_random_phase	222	4785

* **MLDC :** Maple_LCM_Dist_ChronoBT_v3

SAT Race'19



Key Idea! : Do not **lose work** in long backtracks

Phase-Saving

a phase selection heuristic

A Lightweight Component Caching Scheme For Satisfiability Solvers (SAT'07)

Knot Pipatsrisawat and Adnan Darwiche

x1	x2	x3	x4	x5
⊥	⊥	⊤	⊥	⊤

Data Strucure

SavedPhase(v) = assignment(v)

Update during backtrack

return SavedPhase(v)

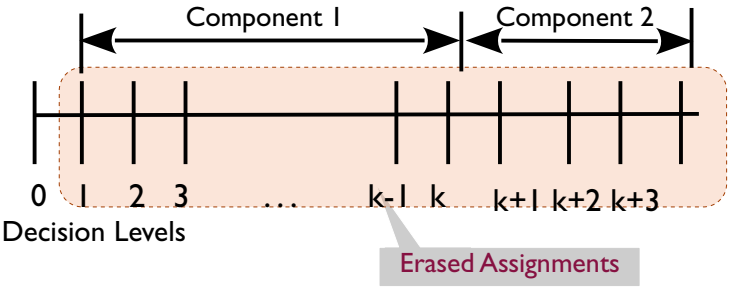
Phase Selection

on SAT '19 instances

solver	# solved	Avg. Runtime*
MLDC	237	4556
MLDC_random_phase	222	4785

* **MLDC :** Maple_LCM_Dist_ChronoBT_v3

SAT Race'19



Key Idea! : Do not **lose work** in long backtracks

Phase-Saving

a phase selection heuristic

A Lightweight Component Caching Scheme For Satisfiability Solvers (SAT'07)

Knot Pipatsrisawat and Adnan Darwiche

x1	x2	x3	x4	x5
⊥	⊥	⊤	⊥	⊤

Data Strucure

SavedPhase(v) = assignment(v)

Update during backtrack

return SavedPhase(v)

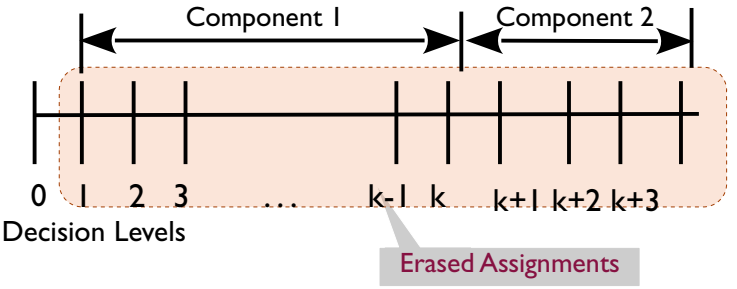
Phase Selection

on SAT '19 instances

solver	# solved	Avg. Runtime*
MLDC	237	4556
MLDC_random_phase	222	4785

* **MLDC :** Maple_LCM_Dist_ChronoBT_v3

SAT Race'19



Key Idea! : Do not **lose work** in long backtracks

Chronological Backtracking

When CDCL is all about Non- Chronological Backtracking

Chronological Backtracking

When CDCL is all about Non- Chronological Backtracking

Chronological Backtracking (SAT'18)

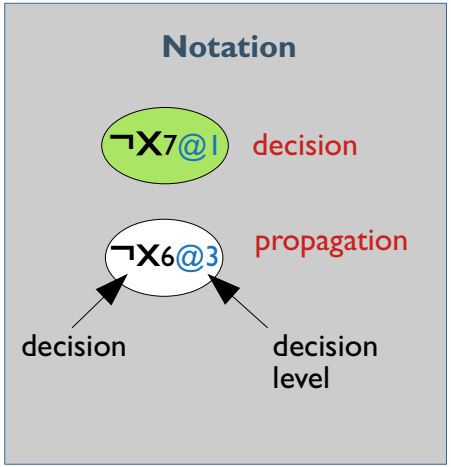
Alexander Nadel and Vadim Ryvchin

Chronological Backtracking

When CDCL is all about Non- Chronological Backtracking

Chronological Backtracking (SAT'18)

Alexander Nadel and Vadim Ryvchin

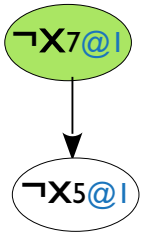


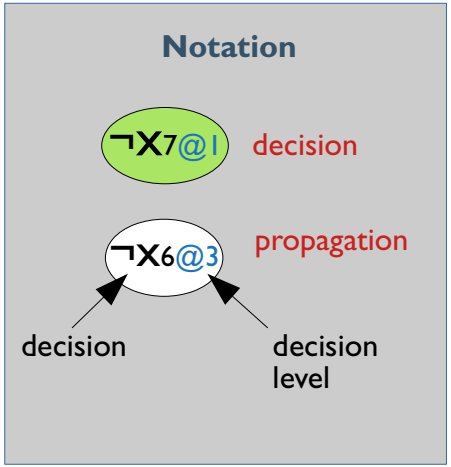
Chronological Backtracking

When CDCL is all about Non- Chronological Backtracking

Chronological Backtracking (SAT'18)

Alexander Nadel and Vadim Ryvchin



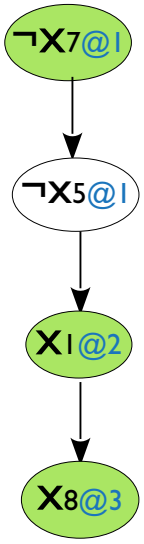


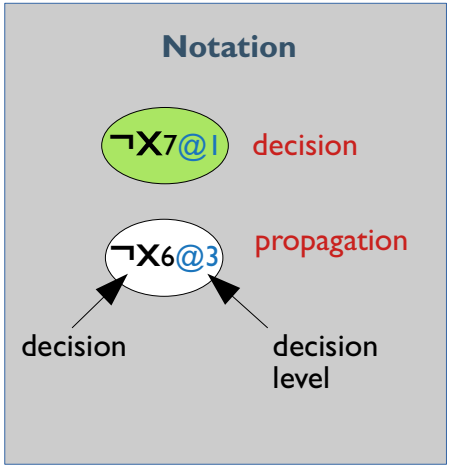
Chronological Backtracking

When CDCL is all about Non- Chronological Backtracking

Chronological Backtracking (SAT'18)

Alexander Nadel and Vadim Ryvchin



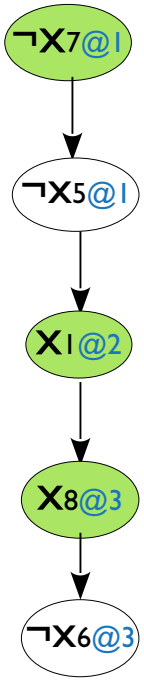


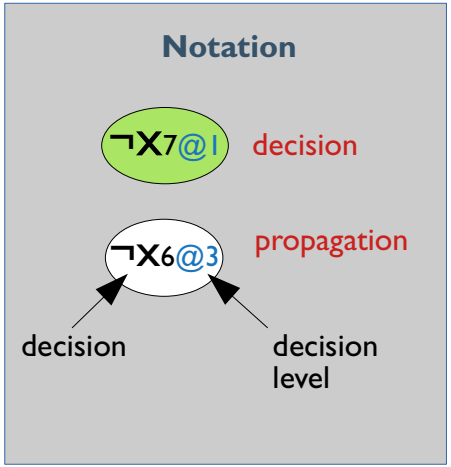
Chronological Backtracking

When CDCL is all about Non- Chronological Backtracking

Chronological Backtracking (SAT'18)

Alexander Nadel and Vadim Ryvchin



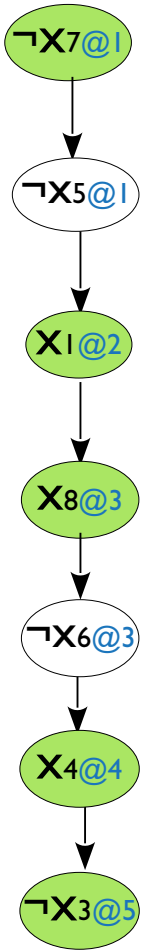


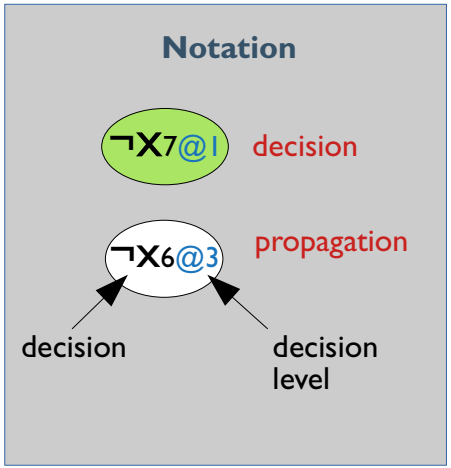
Chronological Backtracking

When CDCL is all about Non- Chronological Backtracking

Chronological Backtracking (SAT'18)

Alexander Nadel and Vadim Ryvchin



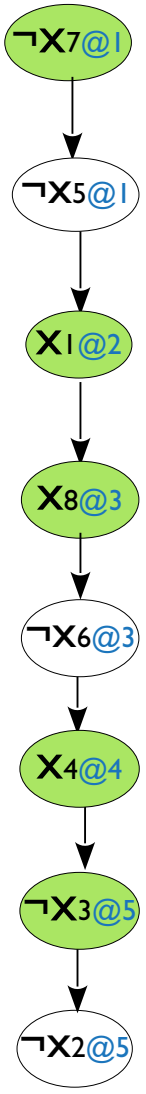


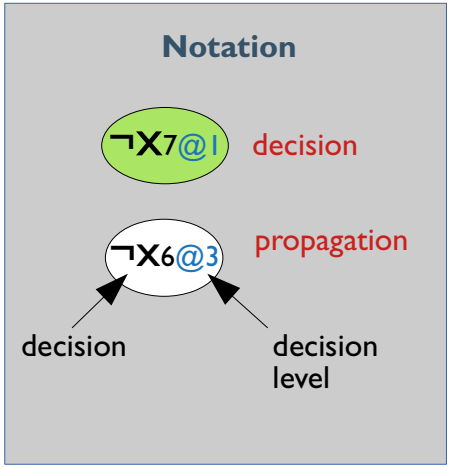
Chronological Backtracking

When CDCL is all about Non- Chronological Backtracking

Chronological Backtracking (SAT'18)

Alexander Nadel and Vadim Ryvchin



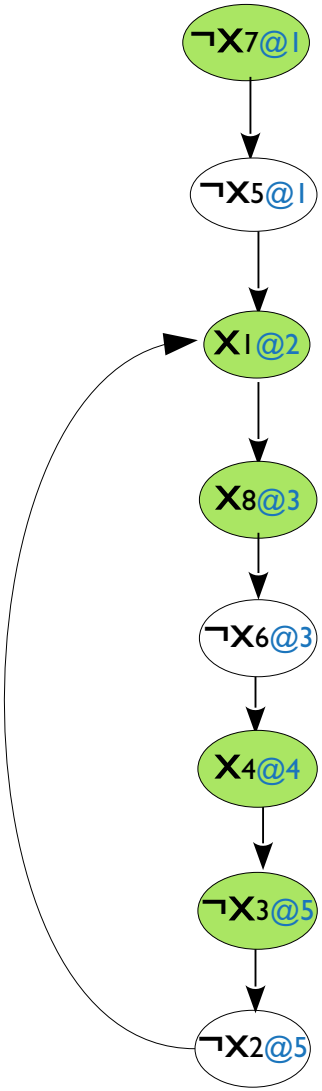


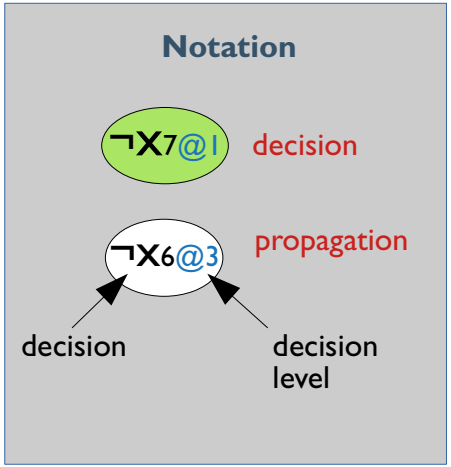
Chronological Backtracking

When CDCL is all about Non- Chronological Backtracking

Chronological Backtracking (SAT'18)

Alexander Nadel and Vadim Ryvchin



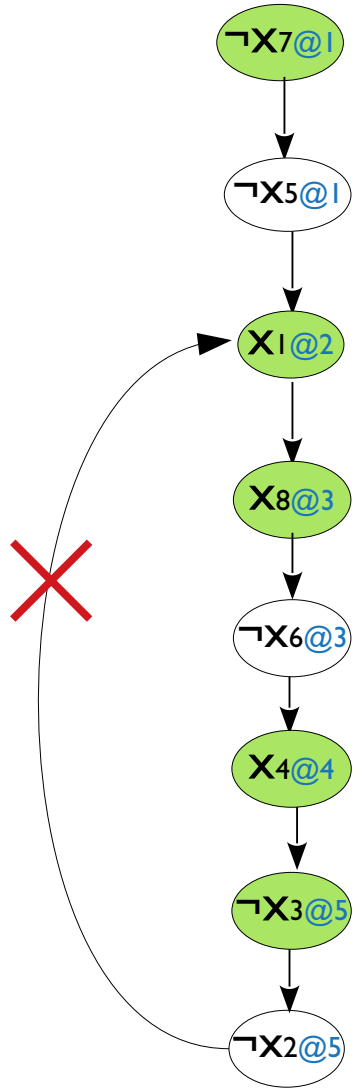


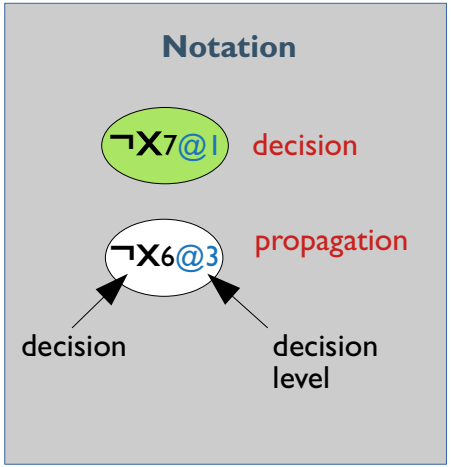
Chronological Backtracking

When CDCL is all about Non- Chronological Backtracking

Chronological Backtracking (SAT'18)

Alexander Nadel and Vadim Ryvchin



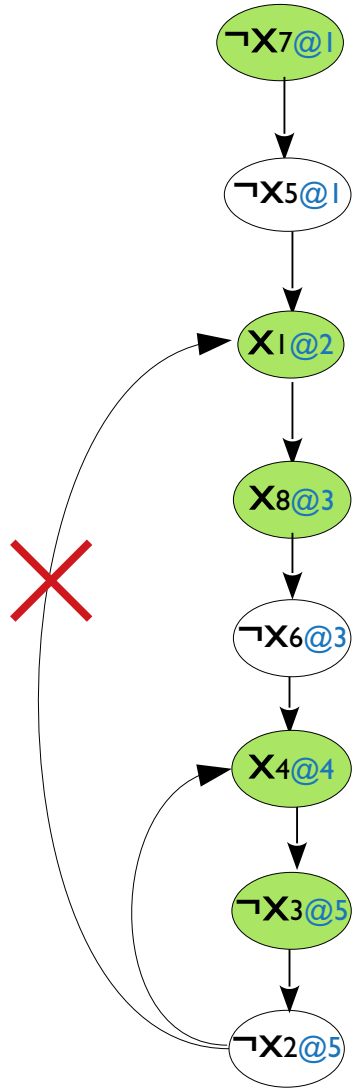


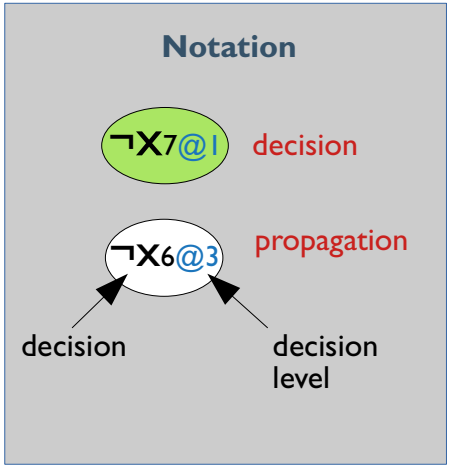
Chronological Backtracking

When CDCL is all about Non- Chronological Backtracking

Chronological Backtracking (SAT'18)

Alexander Nadel and Vadim Ryvchin





Chronological Backtracking

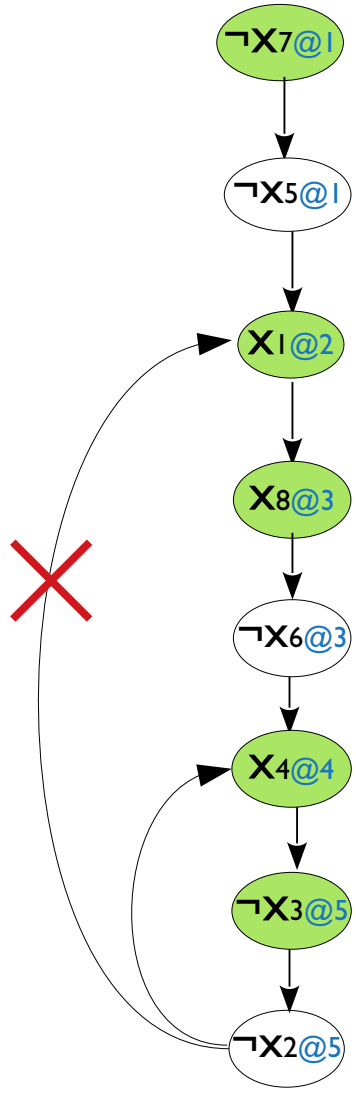
When CDCL is all about Non- Chronological Backtracking

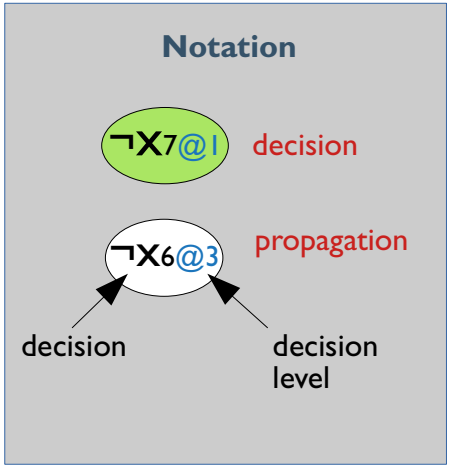
Chronological Backtracking (SAT'18)

Alexander Nadel and Vadim Ryvchin

Backing Backtracking (SAT'19)

Sibylle Möhle and Armin Biere





Chronological Backtracking

When CDCL is all about Non- Chronological Backtracking

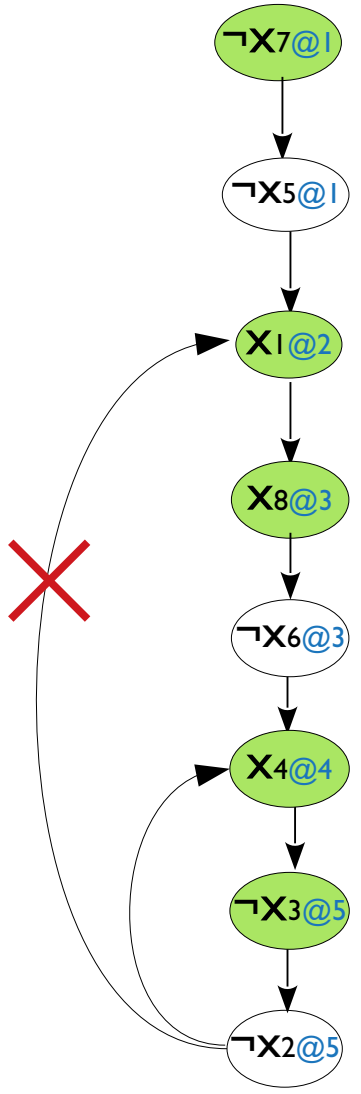
Chronological Backtracking (SAT'18)

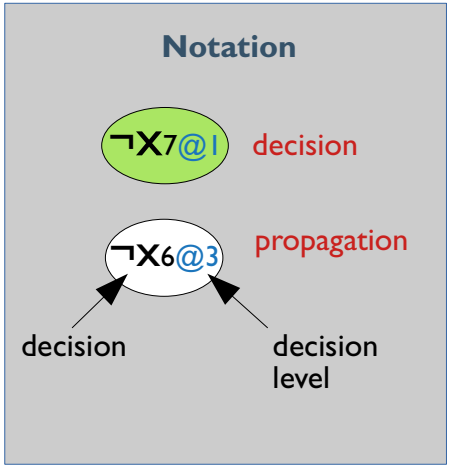
Alexander Nadel and Vadim Ryvchin

Backing Backtracking (SAT'19)

Sibylle Möhle and Armin Biere

- A combination of CB and NCB works the best.





Chronological Backtracking

When CDCL is all about Non- Chronological Backtracking

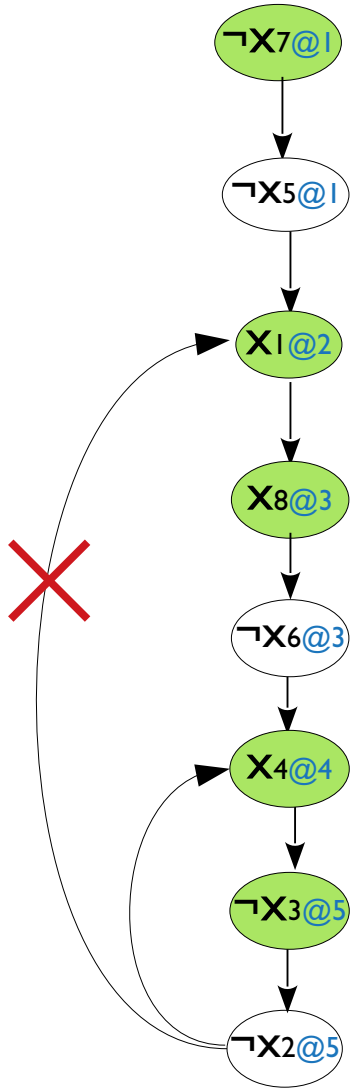
Chronological Backtracking (SAT'18)

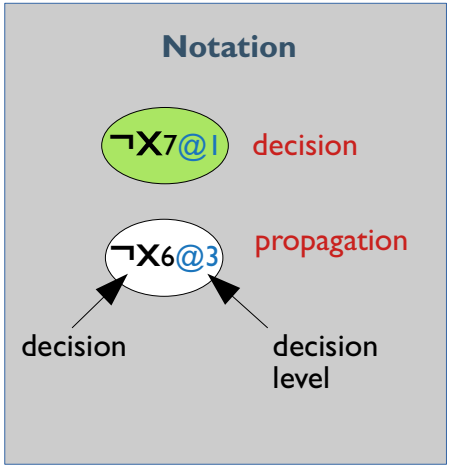
Alexander Nadel and Vadim Ryvchin

Backing Backtracking (SAT'19)

Sibylle Möhle and Armin Biere

- A combination of CB and NCB works the best.
- If the backtracking level is too high, do CB.





Chronological Backtracking

When CDCL is all about Non- Chronological Backtracking

Chronological Backtracking (SAT'18)

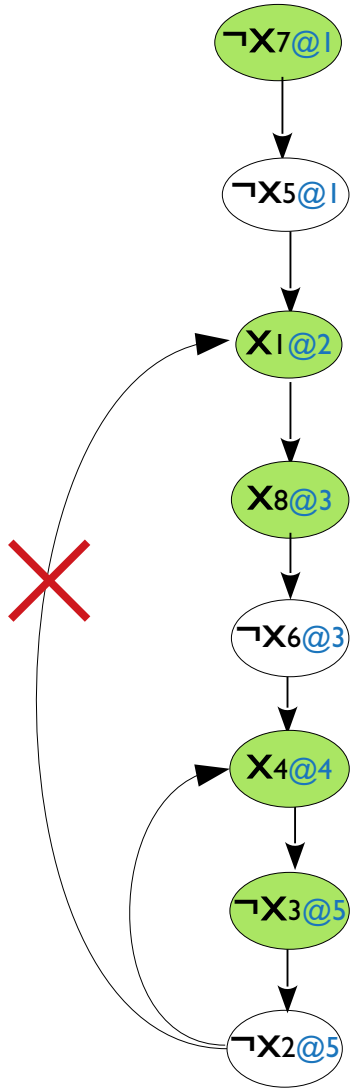
Alexander Nadel and Vadim Ryvchin

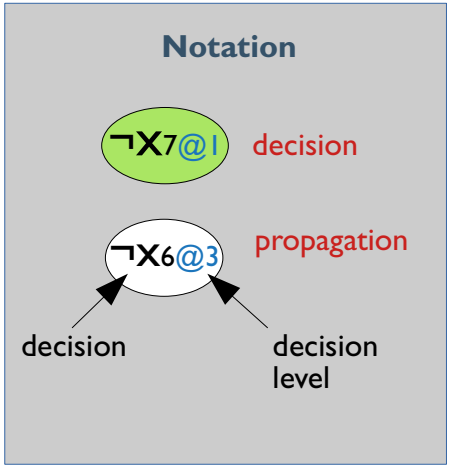
Backing Backtracking (SAT'19)

Sibylle Möhle and Armin Biere

- A combination of CB and NCB works the best.
- If the backtracking level is too high, do CB.

Question : Is phase saving still useful, if the solver backtracks chronologically?





How well are they together?

Testing coupling of Phase Saving and Chronological Backtracking (CB)

How well are they together?

Testing coupling of Phase Saving and Chronological Backtracking (CB)

Introduce CB-state and NCB-state

How well are they together?

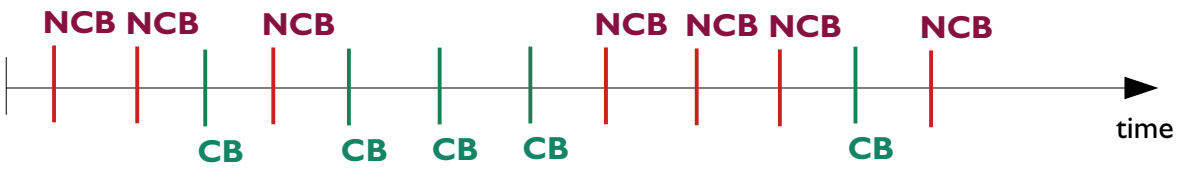
Testing coupling of Phase Saving and Chronological Backtracking (CB)

Introduce CB-state and NCB-state

How well are they together?

Testing coupling of Phase Saving and Chronological Backtracking (CB)

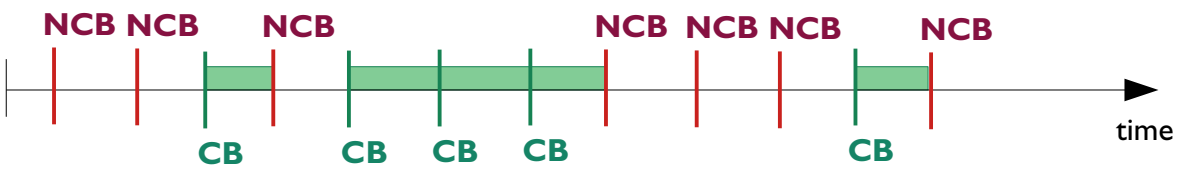
Introduce CB-state and NCB-state



How well are they together?

Testing coupling of Phase Saving and Chronological Backtracking (CB)

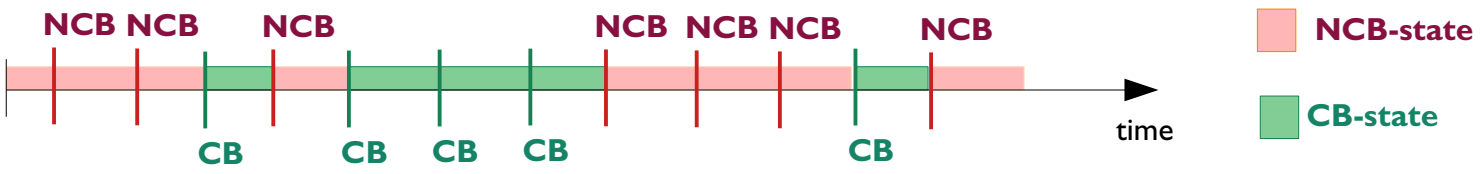
Introduce CB-state and NCB-state



How well are they together?

Testing coupling of Phase Saving and Chronological Backtracking (CB)

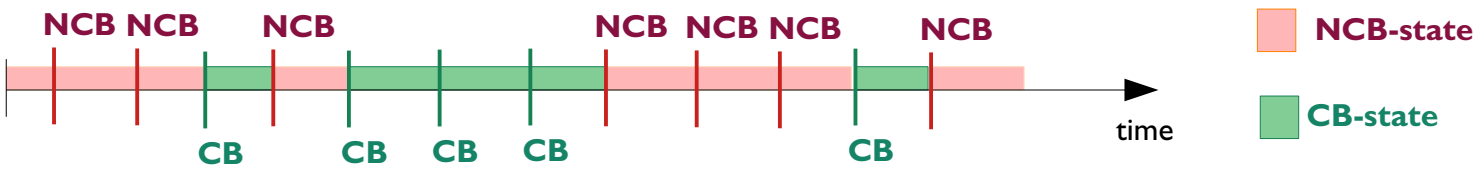
Introduce CB-state and NCB-state



How well are they together?

Testing coupling of Phase Saving and Chronological Backtracking (CB)

Introduce CB-state and NCB-state

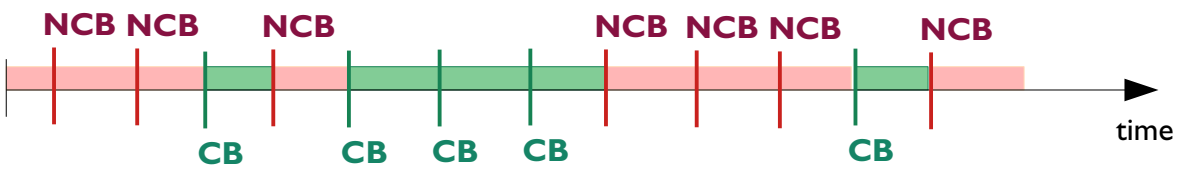


Vary phase selection heuristics in CB-state

How well are they together?

Testing coupling of Phase Saving and Chronological Backtracking (CB)

Introduce CB-state and NCB-state



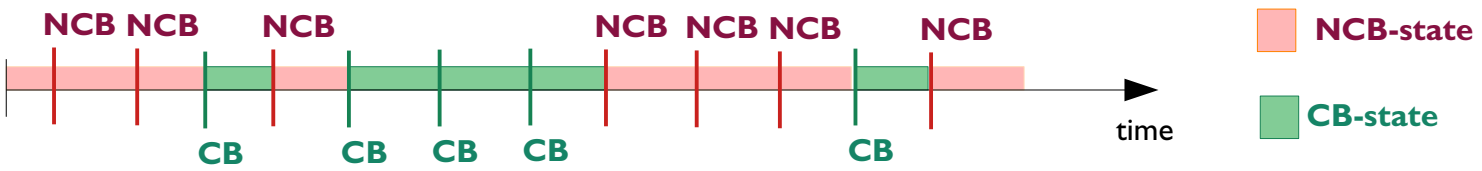
Vary phase selection heuristics in CB-state

Phase Selection Heuristic used		Instances solved	Avg. Runtime*
NCB-state	CB-state		
Phase Saving	Phase Saving		
Phase Saving	Random		

How well are they together?

Testing coupling of Phase Saving and Chronological Backtracking (CB)

Introduce CB-state and NCB-state



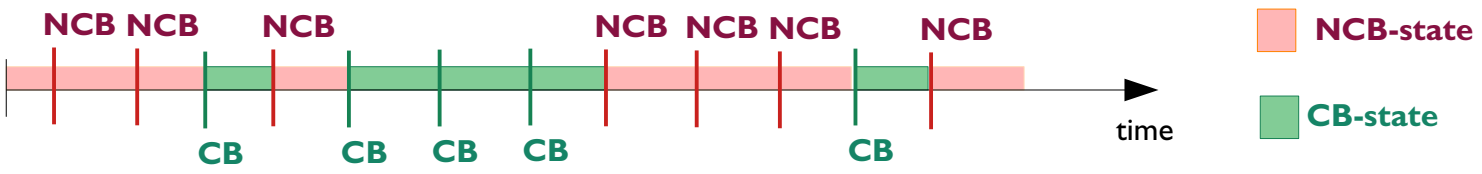
Vary phase selection heuristics in CB-state

Phase Selection Heuristic used		Instances solved	Avg. Runtime*
NCB-state	CB-state		
Phase Saving	Phase Saving	237	4607
Phase Saving	Random	239	4537

How well are they together?

Testing coupling of Phase Saving and Chronological Backtracking (CB)

Introduce CB-state and NCB-state



Vary phase selection heuristics in CB-state

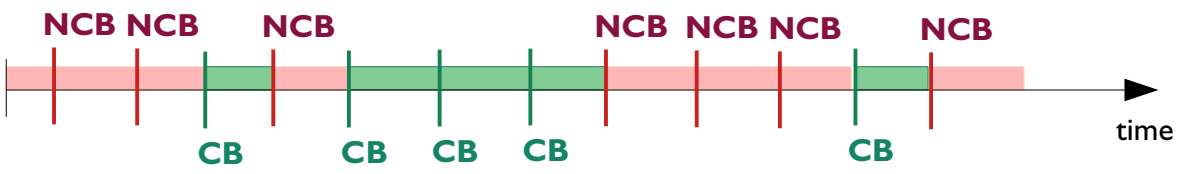
Phase Selection Heuristic used		Instances solved	Avg. Runtime*
NCB-state	**CB-state**		
Phase Saving	Phase Saving	237	4607
Phase Saving	Random	239	4537
Phase Saving	Always False	235	4679
Phase Saving	Opp. Phase Saving	237	4785

* **Base Solver :** Maple_LCM_Dist_ChronoBT_v3
Benchmarks : SAT Race '19

How well are they together?

Testing coupling of Phase Saving and Chronological Backtracking (CB)

Introduce CB-state and NCB-state



Vary phase selection heuristics in CB-state

Phase Selection Heuristic used		Instances solved	Avg. Runtime*
NCB-state	**CB-state**		
Phase Saving	Phase Saving	237	4607
Phase Saving	Random	239	4537
Phase Saving	Always False	235	4679
Phase Saving	Opp. Phase Saving	237	4785
Random	Random	222	5040

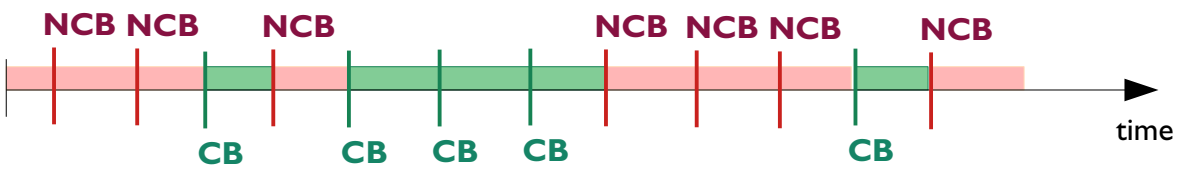
* **Base Solver :** Maple_LCM_Dist_ChronoBT_v3
Benchmarks : SAT Race '19

How well are they together?

Testing coupling of Phase Saving and Chronological Backtracking (CB)

Introduce CB-state and NCB-state

Vary phase selection heuristics in CB-state



Conclusion : Phase Saving's usefulness is not valid for CB.

Phase Selection Heuristic used		Instances solved	Avg. Runtime*
NCB-state	CB-state		
Phase Saving	Phase Saving	237	4607
Phase Saving	Random	239	4537
Phase Saving	Always False	235	4679
Phase Saving	Opp. Phase Saving	237	4785
Random	Random	222	5040

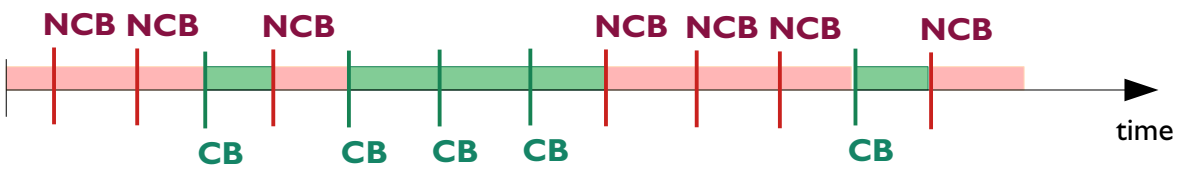
* **Base Solver :** Maple_LCM_Dist_ChronoBT_v3
Benchmarks : SAT Race '19

How well are they together?

Testing coupling of Phase Saving and Chronological Backtracking (CB)

Introduce CB-state and NCB-state

Vary phase selection heuristics in CB-state



Conclusion : Phase Saving's usefulness is not valid for CB.

Phase Selection Heuristic used		Instances solved	Avg. Runtime*
NCB-state	**CB-state**		
Phase Saving	Phase Saving	237	4607
Phase Saving	Random	239	4537
Phase Saving	Always False	235	4679
Phase Saving	Opp. Phase Saving	237	4785
Random	Random	222	5040

* **Base Solver :** Maple_LCM_Dist_ChronoBT_v3
Benchmarks : SAT Race '19

Decaying Phase Score (DPS)

Idea! : Capture the "trend" of phase for the variables

Decaying Phase Score (DPS)

Idea! : Capture the "trend" of phase for the variables

Phase Saving

x1	x2	x3	x4	x5
⊥	⊥	⊤	⊥	⊤

SavedPhase(v) = assignment(v)

return SavedPhase(v)

Data Strucure

Update during backtrack

Phase Selection

Decaying Phase Score (DPS)

Idea! : Capture the "trend" of phase for the variables

Phase Saving

x1	x2	x3	x4	x5
⊥	⊥	⊤	⊥	⊤

SavedPhase(v) = assignment(v)

return SavedPhase(v)

Data Strucure	Update during backtrack	Phase Selection

DPS

x1	x2	x3	x4	x5
2.50	- 0.5	- 9.3	7.9	0.25

Decaying Phase Score (DPS)

Idea! : Capture the "trend" of phase for the variables

	Phase Saving	DPS
Data Strucure	x1 x2 x3 x4 x5 ⊥ ⊥ ⊤ ⊥ ⊤	x1 x2 x3 x4 x5 2.50 - 0.5 - 9.3 7.9 0.25
Update during backtrack	**SavedPhase**(v) = assignment(v)	**DPS**(v) = λ·DPS(v) + polarity(v) 0.5 < λ < 1.0 polarity(⊤) = +1 polarity(⊥) = - 1
Phase Selection	**return** SavedPhase(v)	

Decaying Phase Score (DPS)

Idea! : Capture the "trend" of phase for the variables

	Phase Saving	DPS
Data Strucure	x1 x2 x3 x4 x5 ⊥ ⊥ ⊤ ⊥ ⊤	x1 x2 x3 x4 x5 2.50 - 0.5 - 9.3 7.9 0.25
Update during backtrack	**SavedPhase**(v) = assignment(v)	**DPS**(v) = λ·DPS(v) + polarity(v) 0.5 < λ < 1.0 | polarity(⊤) = +1 polarity(⊥) = - 1
Phase Selection	**return** SavedPhase(v)	

Decaying Phase Score (DPS)

Idea! : Capture the "trend" of phase for the variables

	Phase Saving	DPS
Data Strucure	x1 x2 x3 x4 x5 ⊥ ⊥ ⊤ ⊥ ⊤	x1 x2 x3 x4 x5 2.50 - 0.5 - 9.3 7.9 0.25
Update during backtrack	**SavedPhase**(v) = assignment(v)	**DPS**(v) = λ·DPS(v) + polarity(v) $0.5 < \lambda < 1.0$ | polarity(⊤) = +1 polarity(⊥) = - 1
Phase Selection	**return** SavedPhase(v)	**if** DPS(v) > 0 **then return** *true* **else return** *false*

Decaying Phase Score (DPS)

Idea! : Capture the "trend" of phase for the variables

	Phase Saving	DPS
Data Strucure	x1 x2 x3 x4 x5 ⊥ ⊥ ⊤ ⊥ ⊤	x1 x2 x3 x4 x5 2.50 - 0.5 - 9.3 7.9 0.25
Update during backtrack	**SavedPhase**(v) = assignment(v)	**DPS**(v) = λ·DPS(v) + polarity(v) 0.5 < λ < 1.0 | polarity(⊤) = +1 polarity(⊥) = - 1
Phase Selection	**return** SavedPhase(v)	**if** DPS(v) > 0 **then return** *true* **else return** *false*

on SAT '19 instances

solver	# solved	Avg. Runtime*
MLDC	237	4556
MLDC_DPS	239	4585

* **MLDC :** Maple_LCM_Dist_ChronoBT_v3

LSIDS : Literal State Independent Decaying Sum

Key Idea! : Prioritize the phase which occur **more** and **recently**
in learnt clauses.

LSIDS : Literal State Independent Decaying Sum

Key Idea! : Prioritize the phase which occur **more** and **recently**
in learnt clauses.

Score for **each** literal, updates according the following rules:

LSIDS : Literal State Independent Decaying Sum

Key Idea! : Prioritize the phase which occur **more** and **recently**
in learnt clauses.

Score for **each** literal, updates according the following rules:

Bump

- learnt clause is $(a \vee b \vee \neg c)$:
  bump score for literals
  a, b, $\neg c$.

LSIDS : Literal State Independent Decaying Sum

Key Idea! : Prioritize the phase which occur **more** and **recently**
in learnt clauses.

Score for **each** literal, updates according the following rules:

Bump

- learnt clause is $(a \vee b \vee \neg c)$:
  bump score for literals
  a, b, $\neg c$.

Decay

- Multiply each score by
  $f = 0.8$ at each conflict.

LSIDS : Literal State Independent Decaying Sum

Key Idea! : Prioritize the phase which occur **more** and **recently**
in learnt clauses.

Score for **each** literal, updates according the following rules:

Bump

- learnt clause is $(a \vee b \vee \neg c)$:
  bump score for literals
  a, b, $\neg c$.

Decay

- Multiply each score by
  $f = 0.8$ at each conflict.

Prioritize the phase which occur **more**

LSIDS : Literal State Independent Decaying Sum

Key Idea! : Prioritize the phase which occur **more** and **recently**
in learnt clauses.

Score for **each** literal, updates according the following rules:

Bump

- learnt clause is $(a \vee b \vee \neg c)$:
  bump score for literals
  a, b, $\neg c$.

Prioritize the phase which occur **more**

Decay

- Multiply each score by
  $f = 0.8$ at each conflict.

Prioritize the phase which occur **recently**

LSIDS : Literal State Independent Decaying Sum

Key Idea! : Prioritize the phase which occur **more** and **recently**
in learnt clauses.

Score for **each** literal, updates according the following rules:

Bump

- learnt clause is $(a \vee b \vee \neg c)$:
  bump score for literals
  a, b, $\neg c$.

Prioritize the phase which occur **more**

Decay

- Multiply each score by
  $f = 0.8$ at each conflict.

Prioritize the phase which occur **recently**

LSIDS : Literal State Independent Decaying Sum

Key Idea! : Prioritize the phase which occur **more** and **recently**
in learnt clauses.

Score for **each** literal, updates according the following rules:

Bump

- learnt clause is $(a \vee b \vee \neg c)$:
  bump score for literals
  a, b, $\neg c$.

Prioritize the phase which occur **more**

Decay

- Multiply each score by
  $f = 0.8$ at each conflict.

Prioritize the phase which occur **recently**

Backtrack Bump

- Bump score for literal a if assignment a is cancelled during backtrack.

LSIDS : Literal State Independent Decaying Sum

Key Idea! : Prioritize the phase which occur **more** and **recently**
in learnt clauses.

Score for **each** literal, updates according the following rules:

Maintain the essence of **phase saving**

Bump

- learnt clause is $(a \vee b \vee \neg c)$:
  bump score for literals
  a, b, $\neg c$.

Prioritize the phase which occur **more**

Decay

- Multiply each score by
  $f = 0.8$ at each conflict.

Prioritize the phase which occur **recently**

Backtrack Bump

- Bump score for literal a if assignment a is cancelled during backtrack.

LSIDS : a scoring scheme for literals

Literal State Independent Decaying Sum

Example!

Learnt Clauses :

c1 : ¬x1 ∨ x2 ∨ ¬x4

c2 : ¬x1 ∨ ¬x2 ∨ x3

c3 : x3 ∨ ¬x4

x1	¬x1	x2	¬x2	x3	¬x3	x4	¬x4
0	0	0	0	0	0	0	0

LSIDS : a scoring scheme for literals

Literal State Independent Decaying Sum

Example!

Learnt Clauses :

c1 : ¬x1 ∨ x2 ∨ ¬x4

c2 : ¬x1 ∨ ¬x2 ∨ x3

c3 : x3 ∨ ¬x4

x1	¬x1	x2	¬x2	x3	¬x3	x4	¬x4
0	0	0	0	0	0	0	0

LSIDS : a scoring scheme for literals

Literal State Independent Decaying Sum

Example!

Learnt Clauses :

c1 : ¬x1 ∨ x2 ∨ ¬x4

c2 : ¬x1 ∨ ¬x2 ∨ x3

c3 : x3 ∨ ¬x4

	x1	¬x1	x2	¬x2	x3	¬x3	x4	¬x4
	0	0	0	0	0	0	0	0
Bump	0	1	1	0	0	0	0	1

LSIDS : a scoring scheme for literals

Literal State Independent Decaying Sum

Example!

Learnt Clauses :

c1 : ¬x1 ∨ x2 ∨ ¬x4

c2 : ¬x1 ∨ ¬x2 ∨ x3

c3 : x3 ∨ ¬x4

	x1	¬x1	x2	¬x2	x3	¬x3	x4	¬x4
	0	0	0	0	0	0	0	0
Bump	0	1	1	0	0	0	0	1

LSIDS : a scoring scheme for literals

Literal State Independent Decaying Sum

Example!

Learnt Clauses :

c1 : ¬x1 ∨ x2 ∨ ¬x4

c2 : ¬x1 ∨ ¬x2 ∨ x3

c3 : x3 ∨ ¬x4

	x1	¬x1	x2	¬x2	x3	¬x3	x4	¬x4
	0	0	0	0	0	0	0	0
Bump	0	1	1	0	0	0	0	1
Decay	0	0.8	0.8	0	0	0	0	0.8

LSIDS : a scoring scheme for literals

Literal State Independent Decaying Sum

Example!

Learnt Clauses :

c1 : ¬x1 ∨ x2 ∨ ¬x4

c2 : ¬x1 ∨ ¬x2 ∨ x3

c3 : x3 ∨ ¬x4

	x1	¬x1	x2	¬x2	x3	¬x3	x4	¬x4
	0	0	0	0	0	0	0	0
Bump	0	1	1	0	0	0	0	1
Decay	0	0.8	0.8	0	0	0	0	0.8
Bump	0	1.8	0.8	1	1	0	0	0.8

LSIDS : a scoring scheme for literals

Literal State Independent Decaying Sum

Example!

Learnt Clauses :

c1 : ¬x1 ∨ x2 ∨ ¬x4

c2 : ¬x1 ∨ ¬x2 ∨ x3

c3 : x3 ∨ ¬x4

	x1	¬x1	x2	¬x2	x3	¬x3	x4	¬x4
	0	0	0	0	0	0	0	0
Bump	0	1	1	0	0	0	0	1
Decay	0	0.8	0.8	0	0	0	0	0.8
Bump	0	1.8	0.8	1	1	0	0	0.8

LSIDS : a scoring scheme for literals

Literal State Independent Decaying Sum

Example!

Learnt Clauses :

c1 : ¬x1 ∨ x2 ∨ ¬x4

c2 : ¬x1 ∨ ¬x2 ∨ x3

c3 : x3 ∨ ¬x4

	x1	¬x1	x2	¬x2	x3	¬x3	x4	¬x4
	0	0	0	0	0	0	0	0
Bump	0	1	1	0	0	0	0	1
Decay	0	0.8	0.8	0	0	0	0	0.8
Bump	0	1.8	0.8	1	1	0	0	0.8
Decay	0	1.44	0.64	0.8	0.8	0	0	0.64

LSIDS : a scoring scheme for literals

Literal State Independent Decaying Sum

Example!

Learnt Clauses :

c1 : ¬x1 ∨ x2 ∨ ¬x4

c2 : ¬x1 ∨ ¬x2 ∨ x3

c3 : x3 ∨ ¬x4

	x1	¬x1	x2	¬x2	x3	¬x3	x4	¬x4
	0	0	0	0	0	0	0	0
Bump	0	1	1	0	0	0	0	1
Decay	0	0.8	0.8	0	0	0	0	0.8
Bump	0	1.8	0.8	1	1	0	0	0.8
Decay	0	1.44	0.64	0.8	0.8	0	0	0.64
Bump	0	1.44	0.64	0.8	1.8	0	0	1.64

LSIDS : a scoring scheme for literals

Literal State Independent Decaying Sum

Example!

Learnt Clauses :

c1 : ¬x1 ∨ x2 ∨ ¬x4

c2 : ¬x1 ∨ ¬x2 ∨ x3

c3 : x3 ∨ ¬x4

Backtrack :

	x1	¬x1	x2	¬x2	x3	¬x3	x4	¬x4
	0	0	0	0	0	0	0	0
Bump	0	1	1	0	0	0	0	1
Decay	0	0.8	0.8	0	0	0	0	0.8
Bump	0	1.8	0.8	1	1	0	0	0.8
Decay	0	1.44	0.64	0.8	0.8	0	0	0.64
Bump	0	1.44	0.64	0.8	1.8	0	0	1.64

LSIDS : a scoring scheme for literals

Literal State Independent Decaying Sum

Example!

Learnt Clauses :

c1 : ¬x1 ∨ x2 ∨ ¬x4

c2 : ¬x1 ∨ ¬x2 ∨ x3

c3 : x3 ∨ ¬x4

Backtrack :

¬x2

	x1	¬x1	x2	¬x2	x3	¬x3	x4	¬x4
	0	0	0	0	0	0	0	0
Bump	0	1	1	0	0	0	0	1
Decay	0	0.8	0.8	0	0	0	0	0.8
Bump	0	1.8	0.8	1	1	0	0	0.8
Decay	0	1.44	0.64	0.8	0.8	0	0	0.64
Bump	0	1.44	0.64	0.8	1.8	0	0	1.64

LSIDS : a scoring scheme for literals

Literal State Independent Decaying Sum

Example!

Learnt Clauses :

c1 : ¬x1 ∨ x2 ∨ ¬x4

c2 : ¬x1 ∨ ¬x2 ∨ x3

c3 : x3 ∨ ¬x4

Backtrack :

¬x2

	x1	¬x1	x2	¬x2	x3	¬x3	x4	¬x4
	0	0	0	0	0	0	0	0
Bump	0	1	1	0	0	0	0	1
Decay	0	0.8	0.8	0	0	0	0	0.8
Bump	0	1.8	0.8	1	1	0	0	0.8
Decay	0	1.44	0.64	0.8	0.8	0	0	0.64
Bump	0	1.44	0.64	0.8	1.8	0	0	1.64
Bump	0	1.44	0.64	2.8	1.8	0	0	1.64

LSIDS : a scoring scheme for literals

Literal State Independent Decaying Sum

Example!

Learnt Clauses :

c1 : ¬x1 ∨ x2 ∨ ¬x4

c2 : ¬x1 ∨ ¬x2 ∨ x3

c3 : x3 ∨ ¬x4

Backtrack :

¬x2

	x1	¬x1	x2	¬x2	x3	¬x3	x4	¬x4
	0	0	0	0	0	0	0	0
Bump	0	1	1	0	0	0	0	1
Decay	0	0.8	0.8	0	0	0	0	0.8
Bump	0	1.8	0.8	1	1	0	0	0.8
Decay	0	1.44	0.64	0.8	0.8	0	0	0.64
Bump	0	1.44	0.64	0.8	1.8	0	0	1.64
Bump	0	1.44	0.64	2.8	1.8	0	0	1.64

Phase Selection

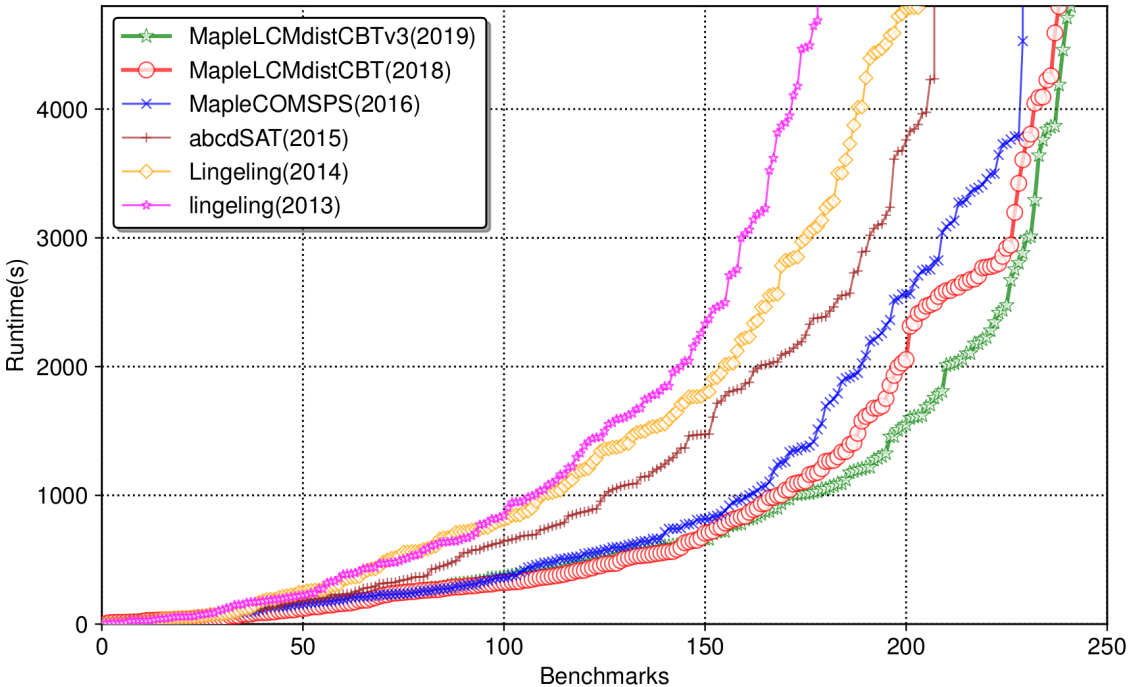
- Compare score of two literals of the variable.
- Choose the one with higher score.

SAT Revolution

over years 2013 - 19

SAT Revolution

over years 2013 - 19



on SAT '19 benchmarks
5000s timeout

solver	year	# solved
lingeling	2013	179
lingeling	2014	188
abcdSAT	2015	202
MapleCOMSPS	2016	224
MapleLCMDistCBT	2018	233
MapleLCMDistCBTv3	2019	237

Results

on 400 SAT '19 instances

solver	# solved	Avg. Runtime*
MLDC		
MLDC_LSIDS		

*** MLDC :** Maple_LCM_Dist_ChronoBT_v3

***Avg. Runtime :** PAR-2 scores

Results

on 400 SAT '19 instances

solver	# solved	Avg. Runtime*
MLDC	237	4556
MLDC_LSIDS	**243**	**4398**

*** MLDC :** Maple_LCM_Dist_ChronoBT_v3

***Avg. Runtime :** PAR-2 scores

Results

on 400 SAT '19 instances

solver	# solved	Avg. Runtime*
MLDC	237	4556
MLDC_LSIDS	**243**	**4398**

*** MLDC :** Maple_LCM_Dist_ChronoBT_v3

***Avg. Runtime :** PAR-2 scores

Room for being skeptical

- Is LSIDS complete noise?

Results

on 400 SAT '19 instances

solver	# solved	Avg. Runtime*
MLDC	237	4556
MLDC_LSIDS	**243**	**4398**

*** MLDC :** Maple_LCM_Dist_ChronoBT_v3

***Avg. Runtime :** PAR-2 scores

Room for being skeptical

- Is LSIDS complete noise?

on 500 cryptographic instances

solver	# solved	Avg. Runtime*
MLDC		
MLDC_LSIDS		

Results

on 400 SAT '19 instances

solver	# solved	Avg. Runtime*
MLDC	237	4556
MLDC_LSIDS	**243**	**4398**

*** MLDC :** Maple_LCM_Dist_ChronoBT_v3

***Avg. Runtime :** PAR-2 scores

Room for being skeptical

- Is LSIDS complete noise?

on 500 cryptographic instances

solver	# solved	Avg. Runtime*
MLDC	291	9939
MLDC_LSIDS	**299**	**9710**

Results

on 400 SAT '19 instances

solver	# solved	Avg. Runtime*
MLDC	237	4556
MLDC_LSIDS	**243**	**4398**

*** MLDC :** Maple_LCM_Dist_ChronoBT_v3

***Avg. Runtime :** PAR-2 scores

Room for being skeptical

- Is LSIDS complete noise?

on 500 cryptographic instances

solver	# solved	Avg. Runtime*
MLDC	291	9939
MLDC_LSIDS	**299**	**9710**

T : CB if (decision level - backtracking level) > T
C : NCB for first C conflicts

Results

on 400 SAT '19 instances

solver	# solved	Avg. Runtime*
MLDC	237	4556
MLDC_LSIDS	**243**	**4398**

Room for being skeptical

- Is LSIDS complete noise?

on 500 cryptographic instances

solver	# solved	Avg. Runtime*
MLDC	291	9939
MLDC_LSIDS	**299**	**9710**

*** MLDC :** Maple_LCM_Dist_ChronoBT_v3

*Avg. Runtime : PAR-2 scores

T : CB if (decision level - backtracking level) > T
C : NCB for first C conflicts

		T = 100				C = 4000			
		C = 2000	C = 3000	C = 4000	C = 5000	T = 25	T = 90	T = 150	T = 200
# solved	MLDC	235	237	235	234	237	233	229	235
	MLDC -LSIDS	**242**	**240**	**243**	**239**	**241**	**238**	**238**	**239**
Avg. Run-time	MLDC	4663	4588	4556	4674	4609	4706	4773	4641
	MLDC -LSIDS	**4506**	**4558**	**4398**	**4575**	**4555**	**4556**	**4622**	**4583**

Key Contribution

Key Contribution

- Discovery : Phase Saving is **not** efficient with CB.
  - "Issues serious warrant to the community."

Key Contribution

- Discovery : Phase Saving is **not** efficient with CB.
  - "Issues serious warrant to the community."
- Designing a phase selection which is efficient with CB.

Key Contribution

- Discovery : Phase Saving is **not** efficient with CB.
  - "Issues serious warrant to the community."
- Designing a phase selection which is efficient with CB.

Thanks!

Special Thanks to Mate Soos and Norbert Manthey
for pointing out discripency between code and paper.

Key Contribution

- Discovery : Phase Saving is **not** efficient with CB.
  - "Issues serious warrant to the community."
- Designing a phase selection which is efficient with CB.

github.com/meelgroup/duriansat

Thanks!

Special Thanks to Mate Soos and Norbert Manthey
for pointing out discripency between code and paper.

Key Contribution

- Discovery : Phase Saving is **not** efficient with CB.
  - "Issues serious warrant to the community."
- Designing a phase selection which is efficient with CB.

github.com/meelgroup/duriansat

Thanks!

Special Thanks to Mate Soos and Norbert Manthey
for pointing out discripency between code and paper.