Dynamic Local Search for SAT: Design, Insights and Analysis

Dave Tompkins
Final PhD Oral Examination
Department of Computer Science, UBC
September 16, 2010

Supervisor: Holger Hoos
Supervisory Committee: Will Evans, Alan Hu
University Examiners: David Kirkpatrick, David Mitchell (SFU)
External Examiner: Steve Prestwich
Chair: Robin Turner (ECE)

Primary Goal

"to advance the state-of-the-art
for SLS algorithms for SAT"

Primary Goal

"to advance the state-of-the-art
for SLS algorithms for SAT"

- *Explicitly:* develop new SLS algorithms that can outperform existing algorithms

Primary Goal

"to advance the state-of-the-art
for SLS algorithms for SAT"

- *Explicitly:* develop new SLS algorithms that can outperform existing algorithms
- *Implicitly:* advance our understanding of current algorithms and introduce tools for developing new algorithms

Overview

- Introduction
  - The Propositional Satisfiability problem (SAT)
  - Stochastic Local Search (SLS) for SAT
  - Summary of key contributions
- Body of Work
- Conclusions
  - Review key contributions
  - Future work

Propositional Satisfiability

- Boolean variables are either (T)rue or (F)alse
  - x_1: Dave's PhD defence will have a positive outcome
  - x_2: Dave will celebrate tonight

$$(\neg x_1 \lor x_2) \land \underbrace{(x_1 \lor \neg x_2)}_{\text{clause}}$$

formula

negative literal

Propositional Satisfiability

- Boolean variables are either (T)rue or (F)alse
  - x_1: Dave's PhD defence will have a positive outcome
  - x_2: Dave will celebrate tonight

x_1	x_2
F	F
F	T
T	F
T	T

$(\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2)$

Propositional Satisfiability

- Boolean variables are either (T)rue or (F)alse
  - x_1: Dave's PhD defence will have a positive outcome
  - x_2: Dave will celebrate tonight

x_1	x_2	$(\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2)$
F	F	***T***
F	T	*F*
T	F	*F*
T	T	***T***

Propositional Satisfiability

- Boolean variables are either (T)rue or (F)alse
  - x_1: Dave's PhD defence will have a positive outcome
  - x_2: Dave will celebrate tonight

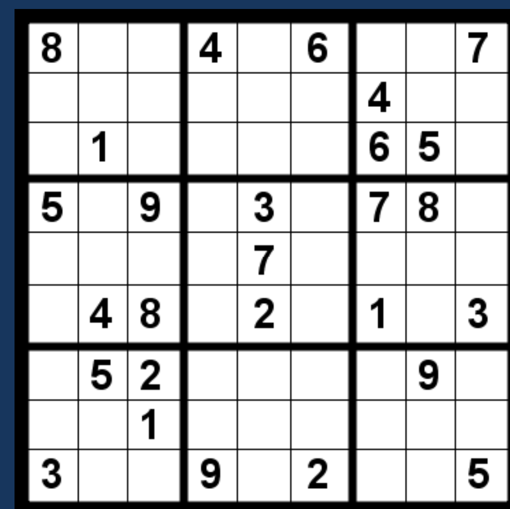
x_1	x_2	$(\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2)$
F	F	***T***
F	T	***F***
T	F	***F***
T	T	***T***

satisfying assignments

- Objective: Given a formula (SAT instance) find a satisfying assignment

Many "Real" SAT Applications

Software Verification



Sudoku

Exponential Search Space

x_1	x_2
F	F
F	T
T	F
T	T

Exponential Search Space

x_1	x_2
F	F
F	T
T	F
T	T

x_1	x_2	x_3
F	F	F
F	F	T
F	T	F
F	T	T
T	F	F
T	F	T
T	T	F
T	T	T

Exponential Search Space

x_1	x_2
F	F
F	T
T	F
T	T

x_1	x_2	x_3
F	F	F
F	F	T
F	T	F
F	T	T
T	F	F
T	F	T
T	T	F
T	T	T

x_1	x_2	x_3	x_4
F	F	F	F
F	F	F	T
F	F	T	F
F	F	T	T
F	T	F	F
F	T	F	T
F	T	T	F
F	T	T	T
T	F	F	F
T	F	F	T
T	F	T	F
T	F	T	T
T	T	F	F
T	T	F	T
T	T	T	F
T	T	T	T

Exponential Search Space

x_1	x_2
F	F
F	T
T	F
T	T

x_1	x_2	x_3
F	F	F
F	F	T
F	T	F
F	T	T
T	F	F
T	F	T
T	T	F
T	T	T

x_1	x_2	x_3	x_4
F	F	F	F
F	F	F	T
F	F	T	F
F	F	T	T
F	T	F	F
F	T	F	T
F	T	T	F
F	T	T	T
T	F	F	F
T	F	F	T
T	F	T	F
T	F	T	T
T	T	F	F
T	T	F	T
T	T	T	F
T	T	T	T

- n variables:
  2^n assignments
- 250 variables
  $\approx 10^{75}$ combinations
  $\approx$ # atoms in the universe

Stochastic Local Search (SLS) for SAT

```
randomly initialize all variables
while (formula not satisfied)
    select a variable and "flip" it
```

Stochastic Local Search (SLS) for SAT

randomly initialize all variables
while (formula not satisfied)
    select a variable and "flip" it

x_1	x_2	x_3	x_4	x_5

$(\neg x_1 \vee x_2 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_4 \vee \neg x_5) \wedge (\neg x_1 \vee x_2 \vee x_3 \vee \neg x_4)$

Stochastic Local Search (SLS) for SAT

randomly initialize all variables
while (formula not satisfied)
    select a variable and "flip" it

x_1	x_2	x_3	x_4	x_5
T	F	F	T	T

$(\neg x_1 \vee x_2 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_4 \vee \neg x_5) \wedge (\neg x_1 \vee x_2 \vee x_3 \vee \neg x_4)$
$(\neg x_1 \vee x_2 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_4 \vee \neg x_5) \wedge (\neg x_1 \vee x_2 \vee x_3 \vee \neg x_4)$

Stochastic Local Search (SLS) for SAT

randomly initialize all variables
while (formula not satisfied)
    select a variable and "flip" it

x_1	x_2	x_3	x_4	x_5
T	F	F	T	T

$(\neg x_1 \vee x_2 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_4 \vee \neg x_5) \wedge (\neg x_1 \vee x_2 \vee x_3 \vee \neg x_4)$
$(\neg x_1 \vee x_2 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_4 \vee \neg x_5) \wedge (\neg x_1 \vee x_2 \vee x_3 \vee \neg x_4)$

Stochastic Local Search (SLS) for SAT

randomly initialize all variables
while (formula not satisfied)
    select a variable and "flip" it

x_1	x_2	x_3	x_4	x_5
T	F	F	T	T
T	F	F	F	T

$(\neg x_1 \vee x_2 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_4 \vee \neg x_5) \wedge (\neg x_1 \vee x_2 \vee x_3 \vee \neg x_4)$
$(\neg x_1 \vee x_2 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_4 \vee \neg x_5) \wedge (\neg x_1 \vee x_2 \vee x_3 \vee \neg x_4)$
$(\neg x_1 \vee x_2 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_4 \vee \neg x_5) \wedge (\neg x_1 \vee x_2 \vee x_3 \vee \neg x_4)$

Stochastic Local Search (SLS) for SAT

randomly initialize all variables
while (formula not satisfied)
    select a variable and "flip" it

x_1	x_2	x_3	x_4	x_5
T	F	F	T	T
T	F	F	F	T
T	T	F	F	T

$(\neg x_1 \vee x_2 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_4 \vee \neg x_5) \wedge (\neg x_1 \vee x_2 \vee x_3 \vee \neg x_4)$
$(\neg x_1 \vee x_2 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_4 \vee \neg x_5) \wedge (\neg x_1 \vee x_2 \vee x_3 \vee \neg x_4)$
$(\neg x_1 \vee x_2 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_4 \vee \neg x_5) \wedge (\neg x_1 \vee x_2 \vee x_3 \vee \neg x_4)$
$(\neg x_1 \vee x_2 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_4 \vee \neg x_5) \wedge (\neg x_1 \vee x_2 \vee x_3 \vee \neg x_4)$

Stochastic Local Search (SLS) for SAT

randomly initialize all variables
while (formula not satisfied)
    select a variable and "flip" it

x_1	x_2	x_3	x_4	x_5
T	F	F	T	T
T	F	F	F	T
T	T	F	F	T
F	T	F	F	T

$(\neg x_1 \vee x_2 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_4 \vee \neg x_5) \wedge (\neg x_1 \vee x_2 \vee x_3 \vee \neg x_4)$
$(\neg x_1 \vee x_2 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_4 \vee \neg x_5) \wedge (\neg x_1 \vee x_2 \vee x_3 \vee \neg x_4)$
$(\neg x_1 \vee x_2 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_4 \vee \neg x_5) \wedge (\neg x_1 \vee x_2 \vee x_3 \vee \neg x_4)$
$(\neg x_1 \vee x_2 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_4 \vee \neg x_5) \wedge (\neg x_1 \vee x_2 \vee x_3 \vee \neg x_4)$
$(\neg x_1 \vee x_2 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_4 \vee \neg x_5) \wedge (\neg x_1 \vee x_2 \vee x_3 \vee \neg x_4)$

Stochastic Local Search (SLS) for SAT

randomly initialize all variables
while (formula not satisfied)
    select a variable and "flip" it

x_1	x_2	x_3	x_4	x_5
T	F	F	T	T
T	F	F	F	T

$(\neg x_1 \vee x_2 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_4 \vee \neg x_5) \wedge (\neg x_1 \vee x_2 \vee x_3 \vee \neg x_4)$
$(\neg x_1 \vee x_2 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_4 \vee \neg x_5) \wedge (\neg x_1 \vee x_2 \vee x_3 \vee \neg x_4)$
$(\neg x_1 \vee x_2 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_4 \vee \neg x_5) \wedge (\neg x_1 \vee x_2 \vee x_3 \vee \neg x_4)$

- Selecting a variable:

  make = # of clauses that become satisfied if we flip x

  break = ... unsatisfied ...

  score = make – break [GSAT: Selman, Levesque & Mitchell, 1992]

Key Contributions

Key Contributions

1. Developed UBCSAT
2. Created SAPS, a Clause Penalty (CP) algorithm
3. Analyzed CP algorithm behaviour
4. Analyzed random decisions in SLS algorithms
5. Introduced a new conceptual model for SLS algorithms with Variable Expressions (VEs)
   – Developed a new Design Architecture (DAVE)

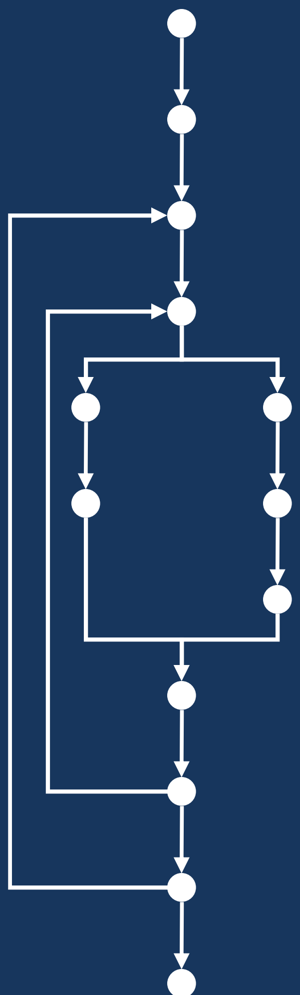
Key Contributions

1. Developed UBCSAT
2. Created SAPS, a Clause Penalty (CP) algorithm
3. Analyzed CP algorithm behaviour
4. Analyzed random decisions in SLS algorithms
5. Introduced a new conceptual model for SLS algorithms with Variable Expressions (VEs)
   – Developed a new Design Architecture (DAVE)

UBCSAT Architecture

while (formula not satisfied)
   select a variable and "flip" it

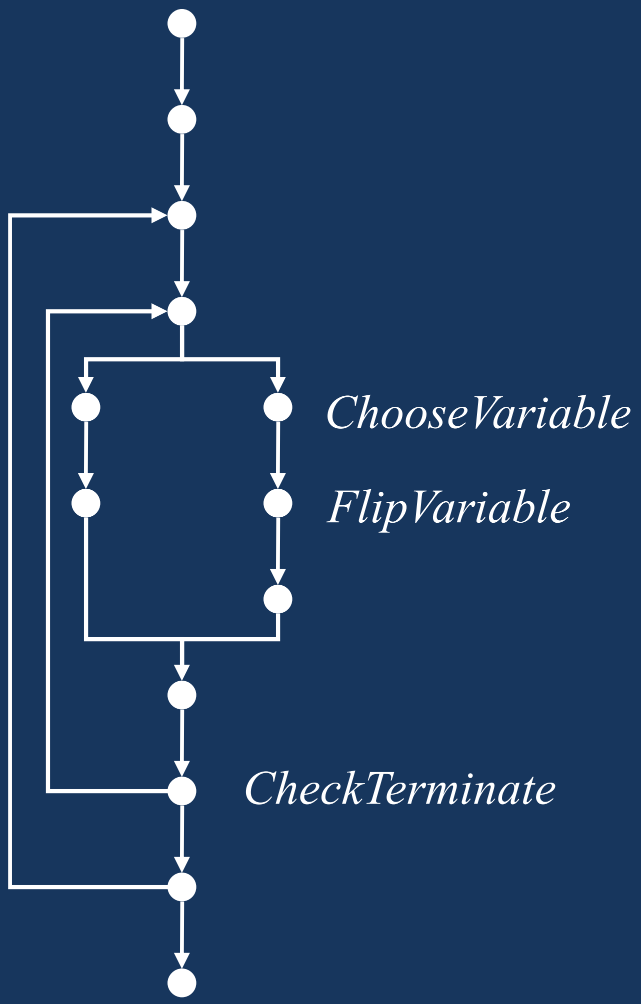
UBCSAT Architecture



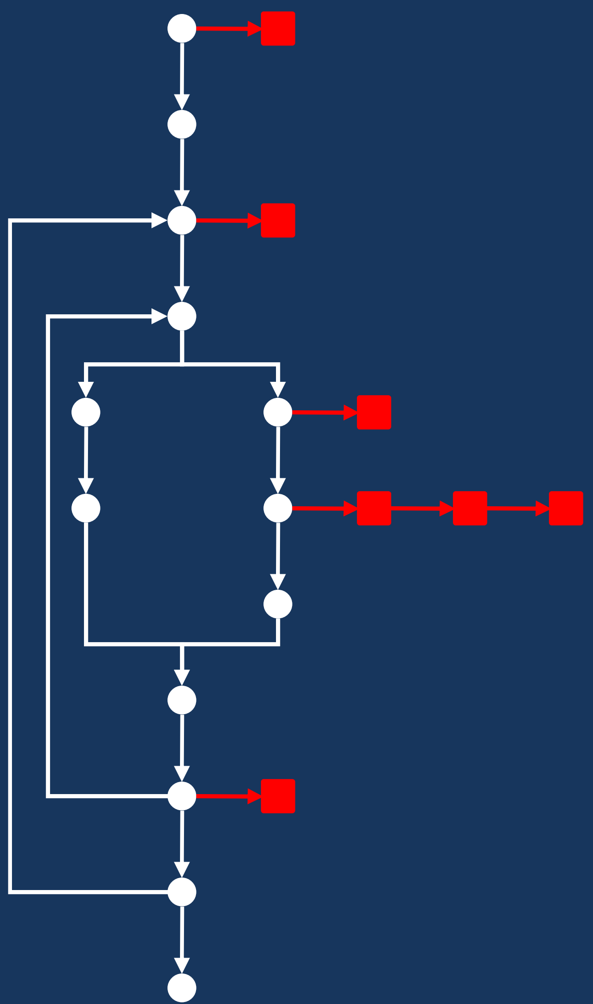
while (formula not satisfied)

select a variable and "flip" it

UBCSAT Architecture

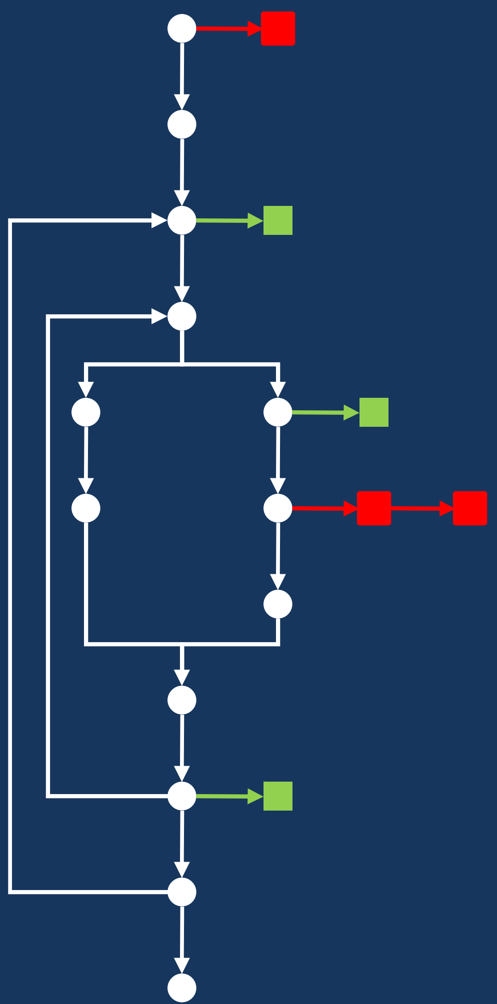


UBCSAT Algorithms



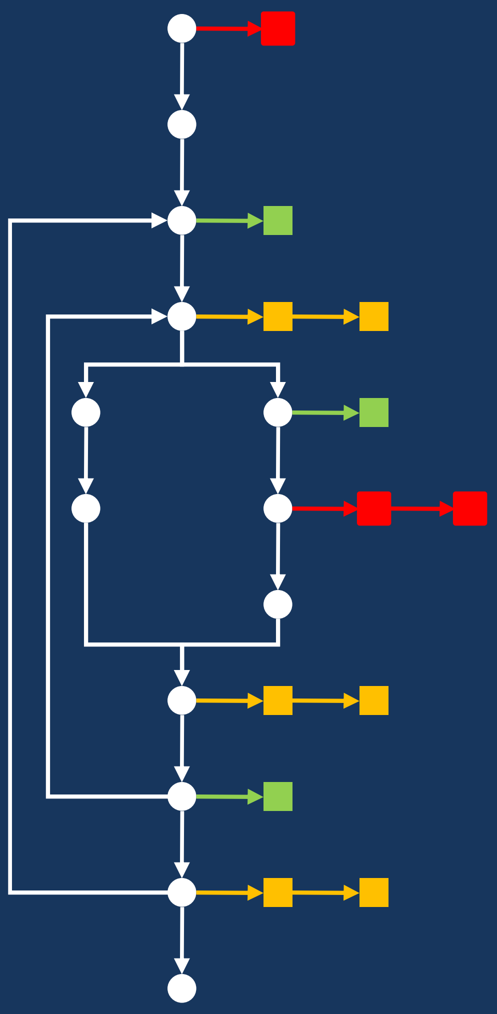
- All (typical) SLS algorithms can be seen as a series of procedures that happen at "event points"
- When you select the algorithm, the appropriate procedures are "triggered"

UBCSAT Algorithms



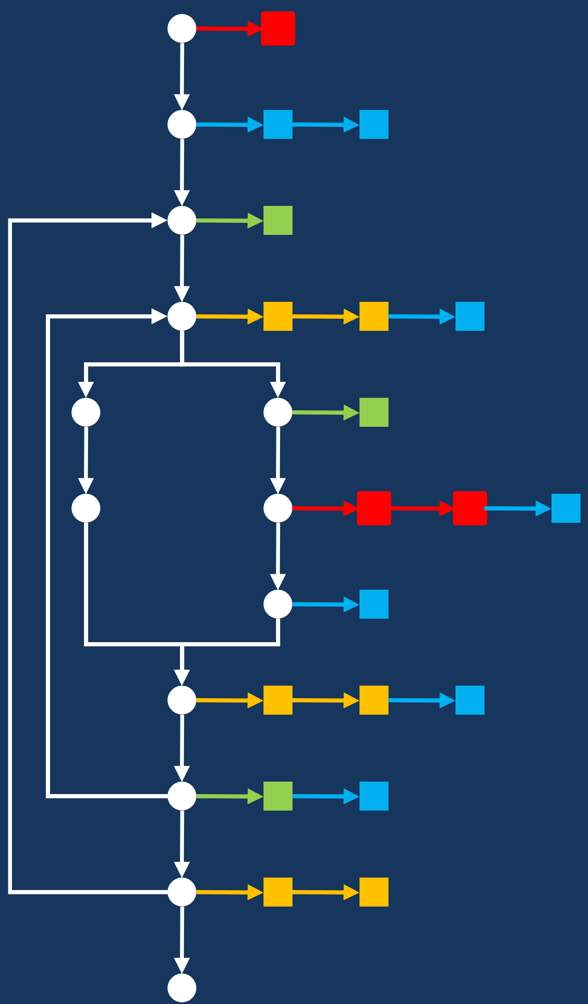
- Similar algorithms can re-use existing triggers

UBCSAT Reports



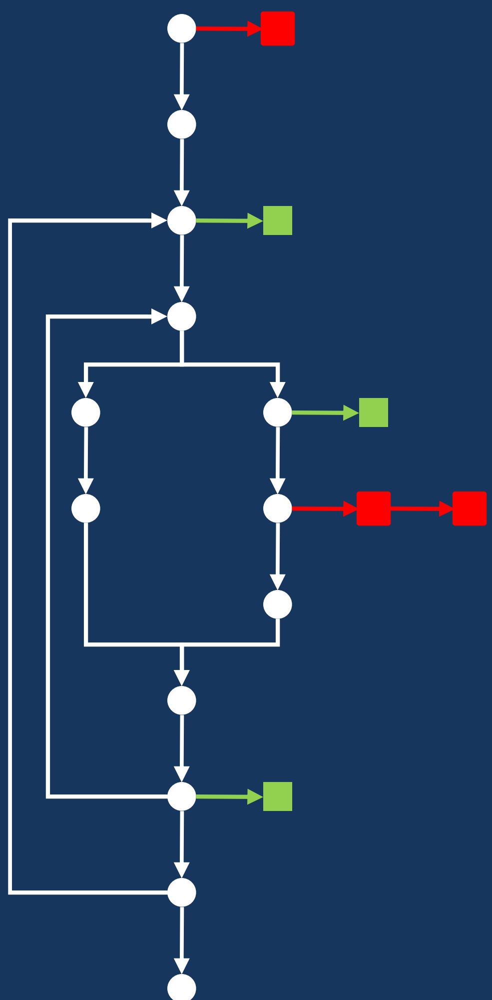
- Additional Reports and Statistics can be "activated" when needed

UBCSAT Reports



- Facilitating empirical analysis is an important component of UBCSAT

UBCSAT Efficiency



- UBCSAT is very efficient with little overhead

algorithm	UBCSAT Speedup
WalkSAT/SKC	1.5x – 2.2x
Novelty	1.3x – 2.0x
GSAT	1.7x – 7.6x
GWSAT	2.5x – 7.4x

UBCSAT

A software framework for SLS algorithms

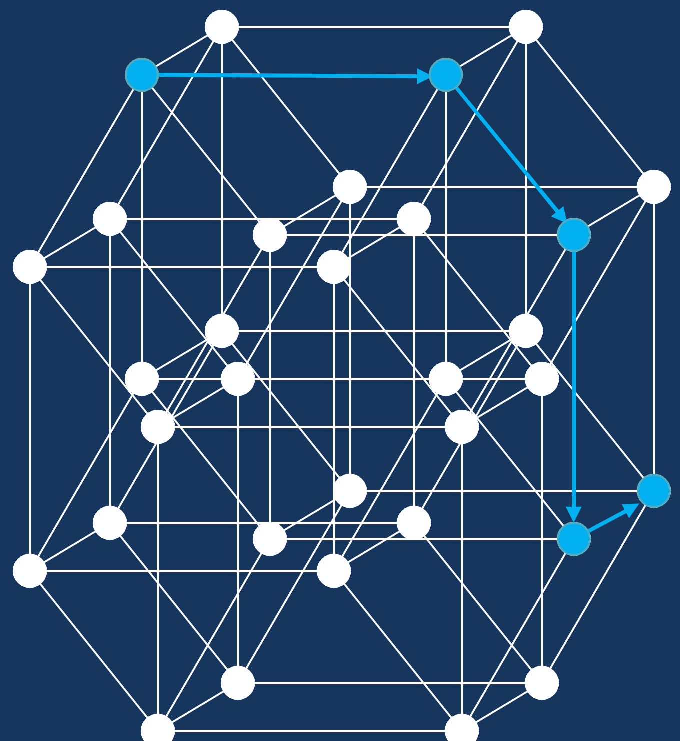
- Incorporates existing SLS algorithms
  - highly efficient, accurate implementations
- Facilitate development of new SLS algorithms
- Advanced empirical analysis of algorithms
- Open-source
- Cornerstone of the dissertation

Key Contributions

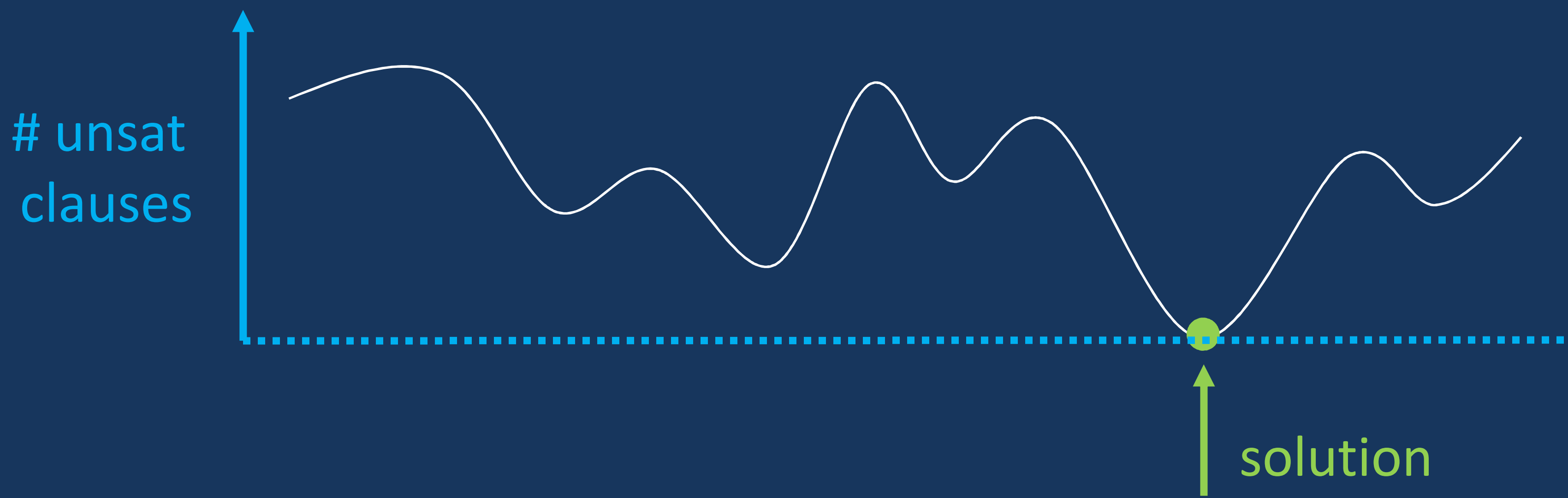
1. Developed UBCSAT
2. Created SAPS, a Clause Penalty (CP) algorithm
3. Analyzed CP algorithm behaviour
4. Analyzed random decisions in SLS algorithms
5. Introduced a new conceptual model for SLS algorithms with Variable Expressions (VEs)
   – Developed a new Design Architecture (DAVE)

SAT Search Space

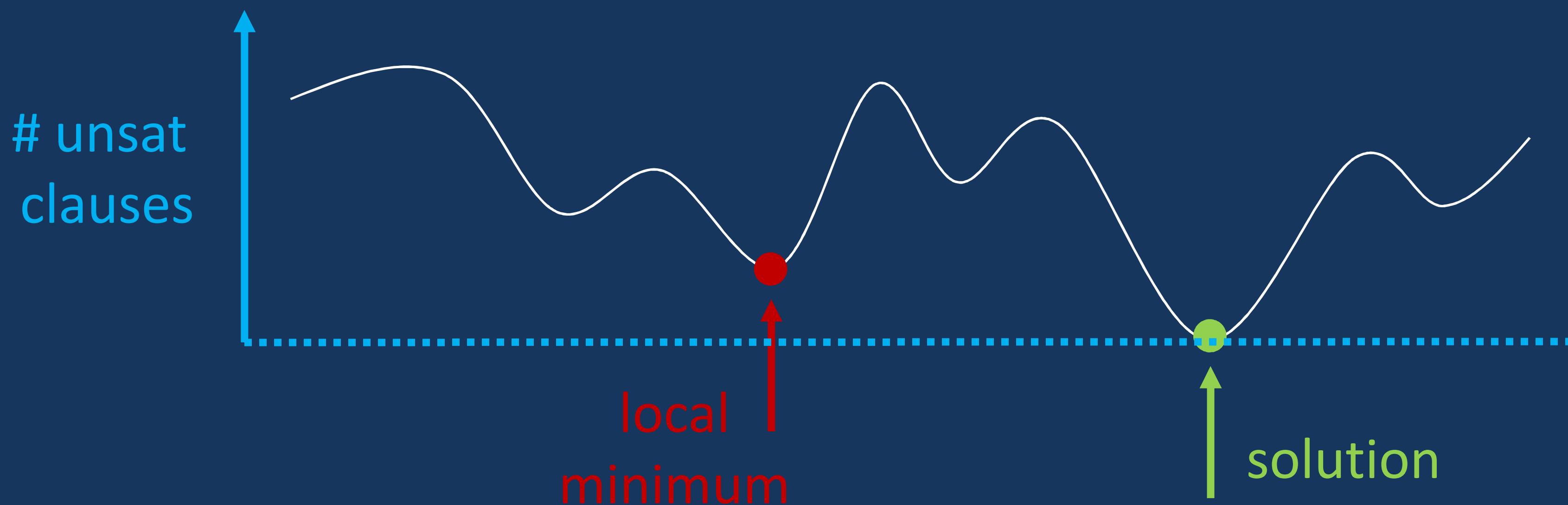
- n-dimensional hypercube



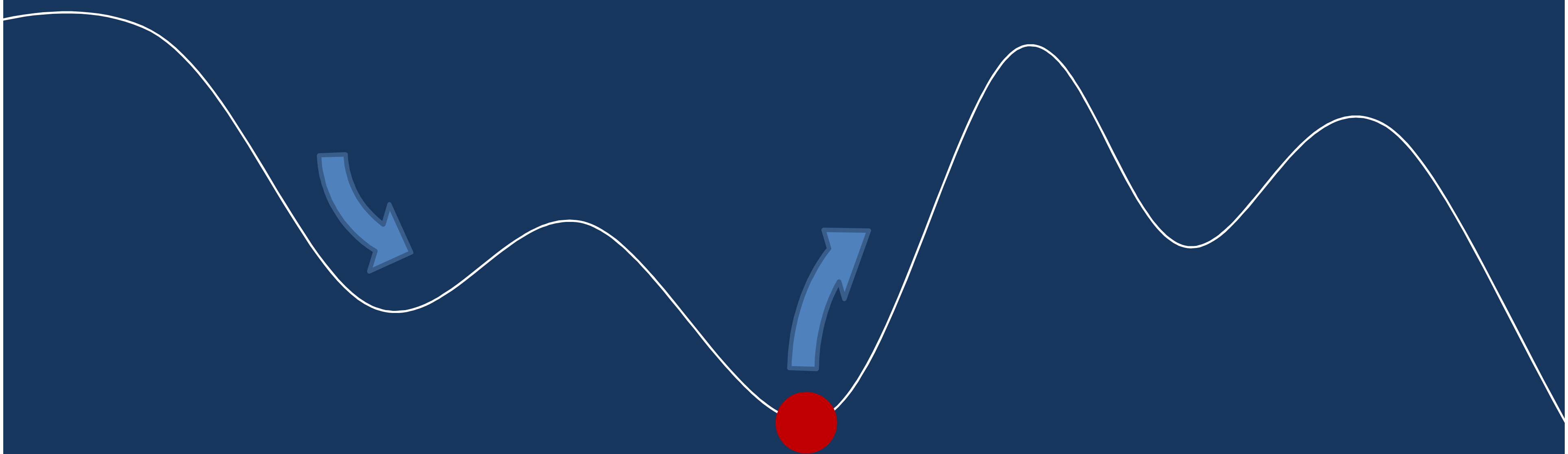
"2D" Search Landscape



"2D" Search Landscape



Intensification & Diversification



Clause Penalties

- Each clause is assigned a penalty value

- Score is no longer just make – break

  $\text{score} = \Sigma_{\text{make}}\ \text{c.penalty} - \Sigma_{\text{break}}\ \text{c.penalty}$

Original Idea:

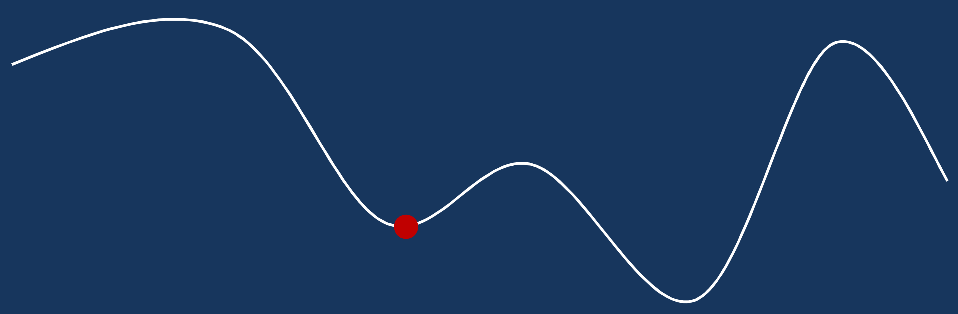
- Breakout Method [Morris, 1993]
- GSAT+CW [Selman & Kautz, 1993]

"Breakout" Approach

- When a local minimum occurs:

  Σ_{make} c.penalty $\leq$ Σ_{break} c.penalty

  increment the penalty for unsatisfied clauses



"Breakout" Approach

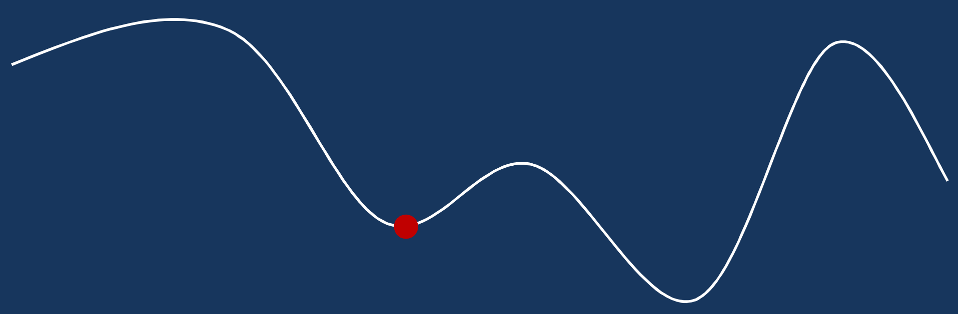
- When a local minimum occurs:

  Σ_{make} c.penalty $\leq$ Σ_{break} c.penalty

  increment the penalty for unsatisfied clauses

- Eventually, will no longer be in a local minimum

  score = Σ_{make} c.penalty – Σ_{break} c.penalty



"Breakout" Approach

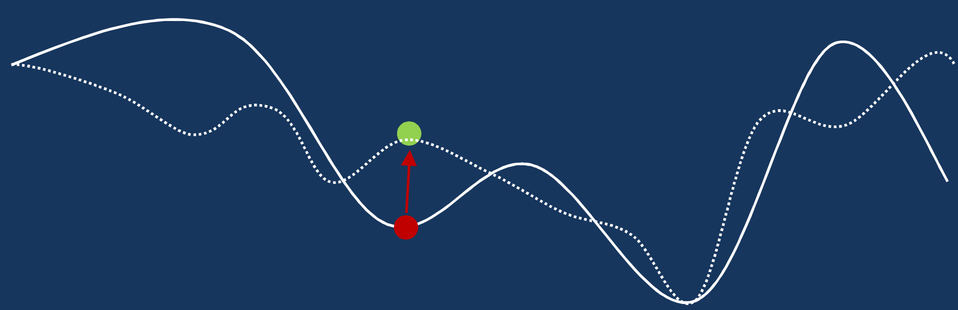
- When a local minimum occurs:

  Σ_{make} c.penalty $\leq$ Σ_{break} c.penalty

  increment the penalty for unsatisfied clauses

- Eventually, will no longer be in a local minimum

score = Σ_{make} c.penalty – Σ_{break} c.penalty



SAPS Algorithm

- Enhancement of existing algorithm
  - Exponentiated Sub-Gradient (ESG)
    [Schuurmans et. al, 2002]

- Multiplicative Scaling

  c.penalty := c.penalty · α

- Probabilistic Smoothing

  with probability (Ps):

    c.penalty := c.penalty + (1-ρ) · avg.penalty

- Scaling And Probabilistic Smoothing (SAPS)

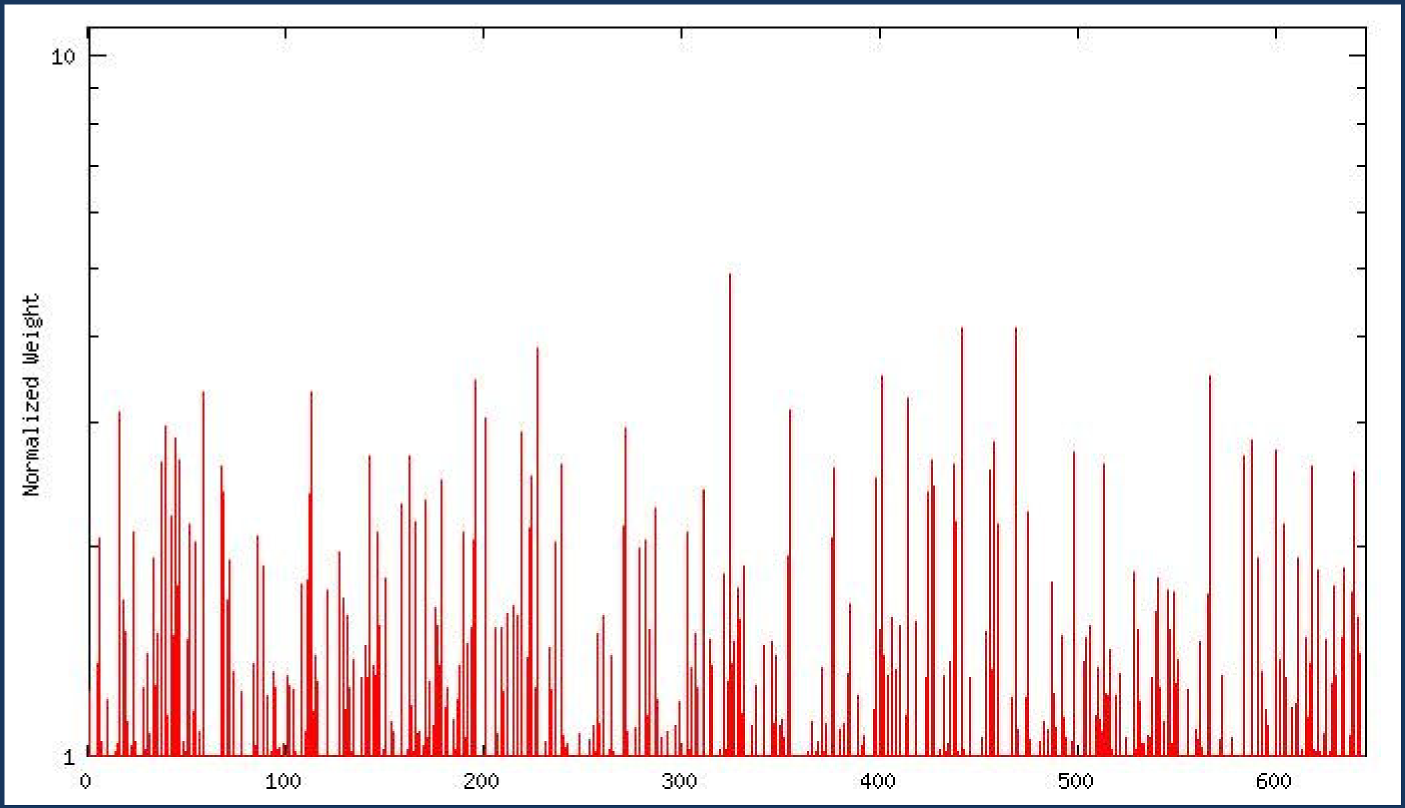
SAPS Algorithm

- Dominated the performance of its predecessor (ESG)
- Still amongst the state-of-the-art solvers
- Led to the work in other chapters

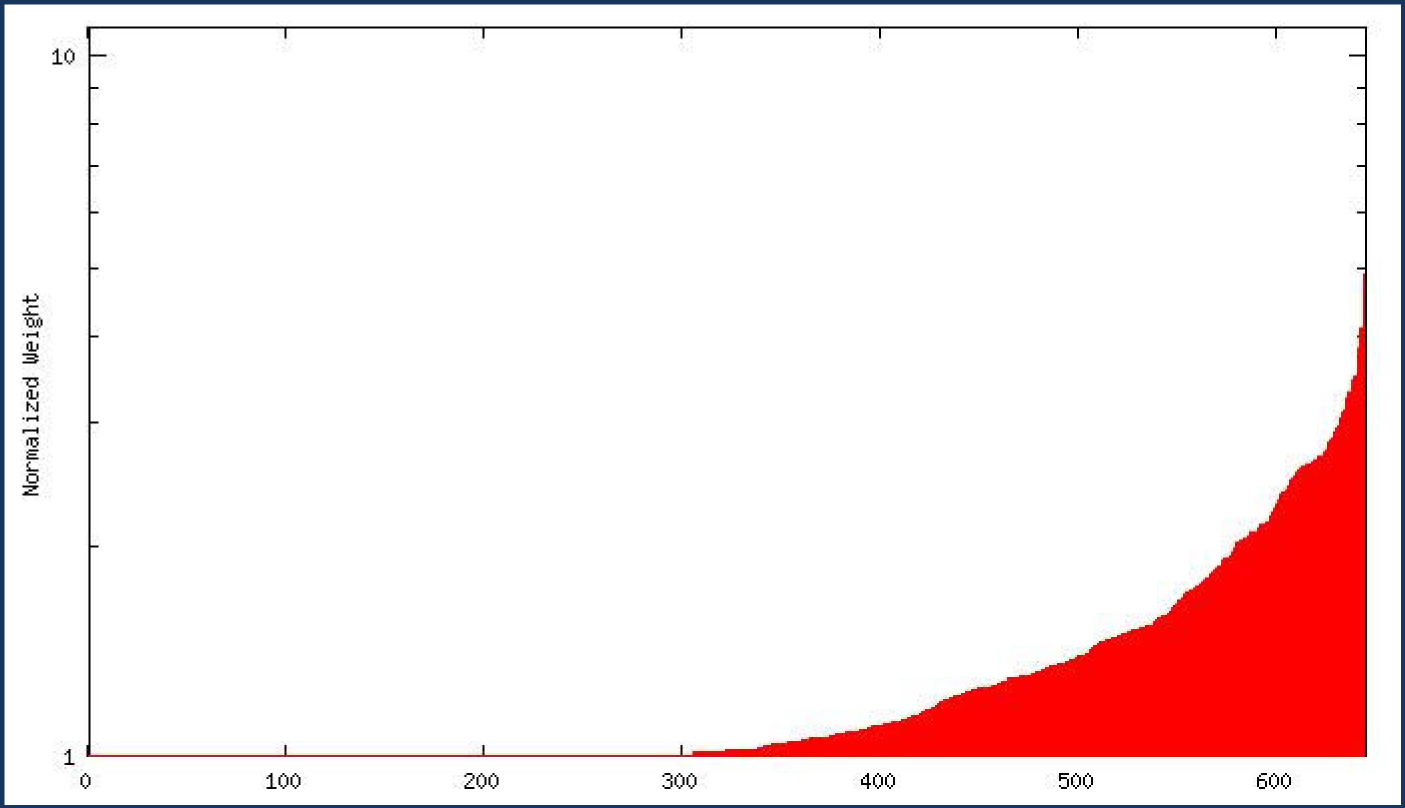
Key Contributions

1. Developed UBCSAT
2. Created SAPS, a Clause Penalty (CP) algorithm
3. Analyzed CP algorithm behaviour
4. Analyzed random decisions in SLS algorithms
5. Introduced a new conceptual model for SLS algorithms with Variable Expressions (VEs)
   – Developed a new Design Architecture (DAVE)

Dynamic Clause Penalties



Clause Penalty Distributions



Clause Penalty Analysis

- We identified instances with "Problem Clauses"
  - We constructed *weighted* instances...
    ... that were *easier* for SLS algorithms to solve
    (80x faster for Adaptive Novelty$^+$)

Clause Penalty Analysis

- A quest for problem clauses
- Analyzed penalty behaviour
- Hardness of warped landscapes
- History ("memory") of the search

- Ultimately: problem clauses are rarely helpful
- Key element of CP algorithms: diversification

Key Contributions

1. Developed UBCSAT
2. Created SAPS, a Clause Penalty (CP) algorithm
3. Analyzed CP algorithm behaviour
4. Analyzed random decisions in SLS algorithms
5. Introduced a new conceptual model for SLS algorithms with Variable Expressions (VEs)
   – Developed a new Design Architecture (DAVE)

Random Decisions

- Stochastic Local Search
- *Quality* of random decision
  - SLS Algorithms are robust (existing random number generators are good enough)
- *Quantity* of random decisions
  - Simple derandomizations can be effective
  - SLS Algorithms exhibit 'chaotic'-like behaviour
  - No real advantage to derandomizing

Key Contributions

1. Developed UBCSAT
2. Created SAPS, a Clause Penalty (CP) algorithm
3. Analyzed CP algorithm behaviour
4. Analyzed random decisions in SLS algorithms
5. Introduced a new conceptual model for SLS algorithms with Variable Expressions (VEs)
   – Developed a new Design Architecture (DAVE)

Variable Properties

- Scoring Properties

  make = # of clauses that become satisfied if we flip x

  break = … unsatisfied …

  score = make – break

Variable Properties

- Scoring Properties
  make = # of clauses that become satisfied if we flip x
  break = ... unsatisfied ...
  score = make – break
- Dynamic Properties
  age = # of steps since x was flipped [TABU, Glover 1986]
  flips = # of times x has been flipped [HSAT, Gent & Walsh 1992]

Variable Properties

- Scoring Properties
  make = # of clauses that become satisfied if we flip x
  break = ... unsatisfied ...
  score = make – break
- Dynamic Properties
  age = # of steps since x was flipped [TABU, Glover 1986]
  flips = # of times x has been flipped [HSAT, Gent & Walsh 1992]
- Static Properties

Variable Expressions (VEs)

- combinations of variable *properties*
  in mathematical expressions:

  make – break
  age
  (make – break) + 3 · $\log_2$(age) + age/flips

- Most existing SLS algorithms use straightforward VEs
  … we explore more complex VEs

- Our work was inspired by:
  Variable Weighting Algorithm VW2 [Prestwich, 2005]

Combining Properties

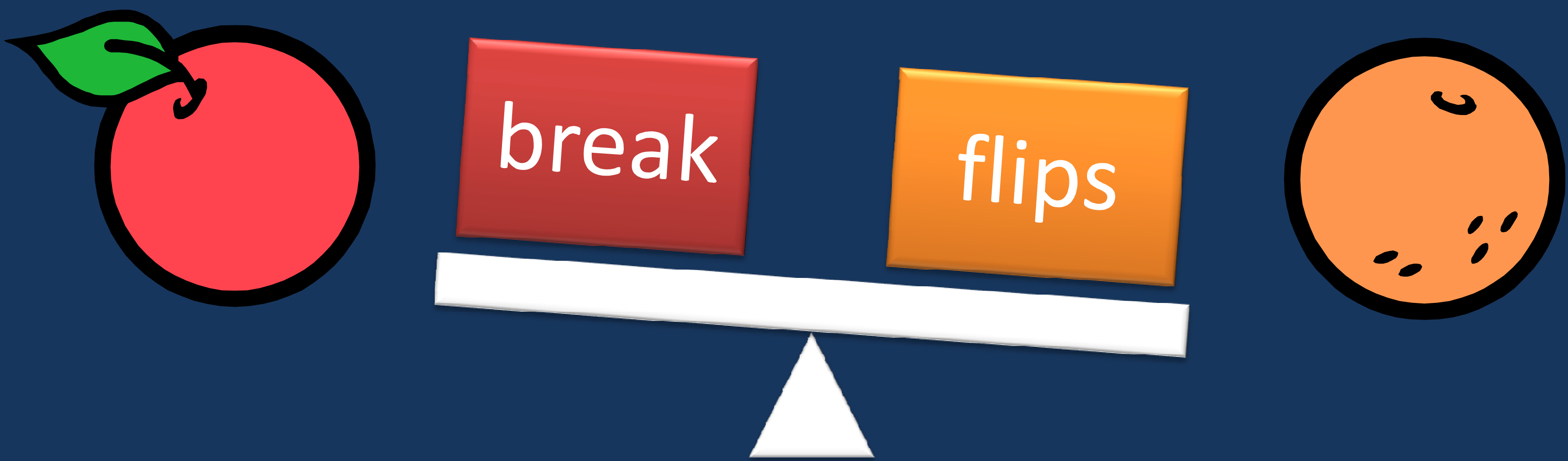
Select variable with minimum value of:

break + c·flips

Combining Properties

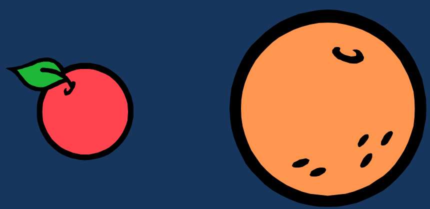
Select variable with minimum value of:

break + c·flips

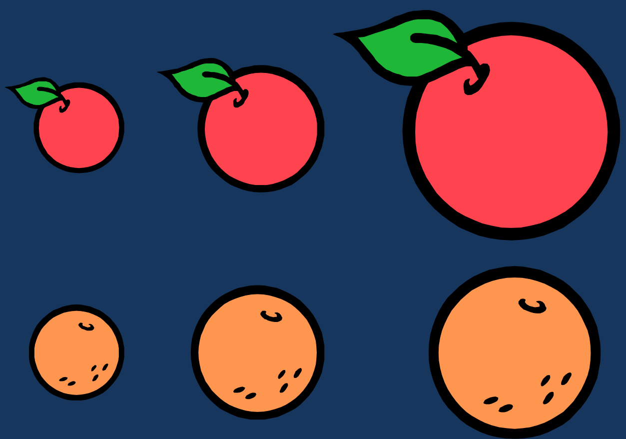


Combining Properties

- Normalize properties values to [0...1] *amongst the "candidate" variables*



- Allow for non-linear normalization

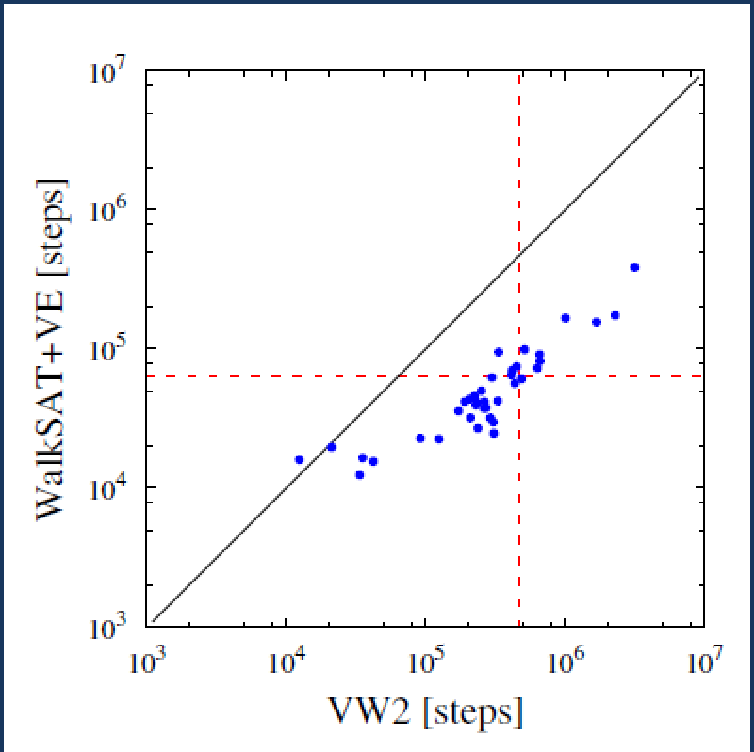


Modifying Existing Algorithms with VEs

- WalkSAT with more complex VE

- Speedup factor:
  7.2x (steps)
  3.1x (time)

- (compared to original WalkSAT)
  > 4000x (steps)
  > 2000x (time)



Our New SLS Model

Our New SLS Model

Separation of: VEs & Selection Mechanism

- Novelty Algorithm [McAllester, Selman & Kautz, 1997]

- Select "best" variable with maximum of:

  (make – break)

  breaking ties by

  (age)

- If the best variable has the minimum

  (age)

  then, with probability p, select 2nd best var.

Separation of: VEs & Selection Mechanism

- Novelty Algorithm [McAllester, Selman & Kautz, 1997]

- Select "best" variable with maximum of:

  (VE_1)

  breaking ties by

  (VE_2)
- If the best variable has the minimum

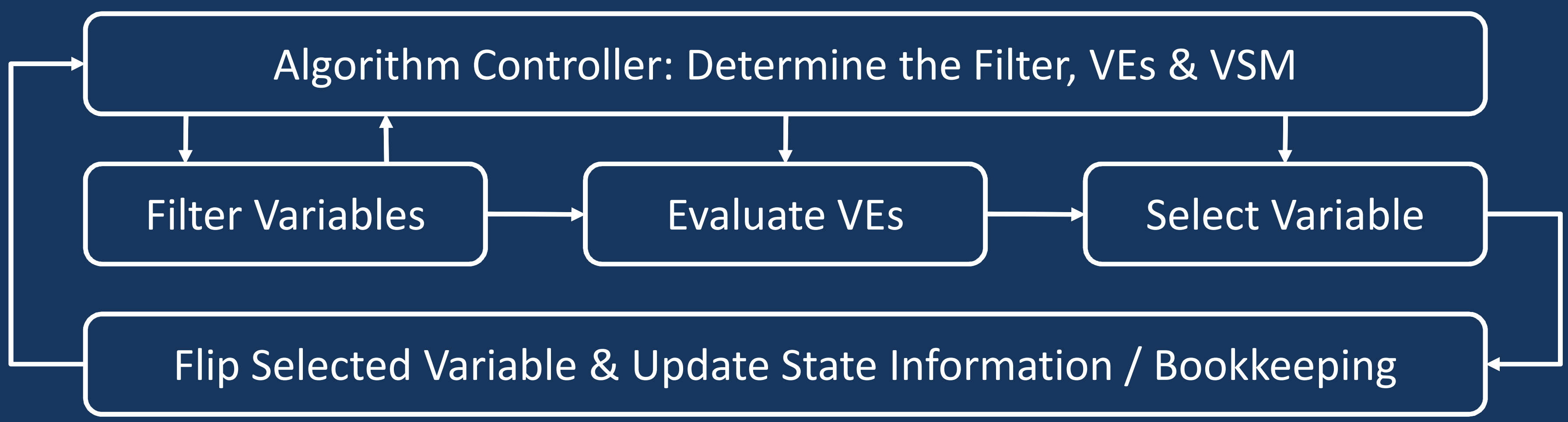
  (VE_3)

  then, with probability p, select 2nd best var.

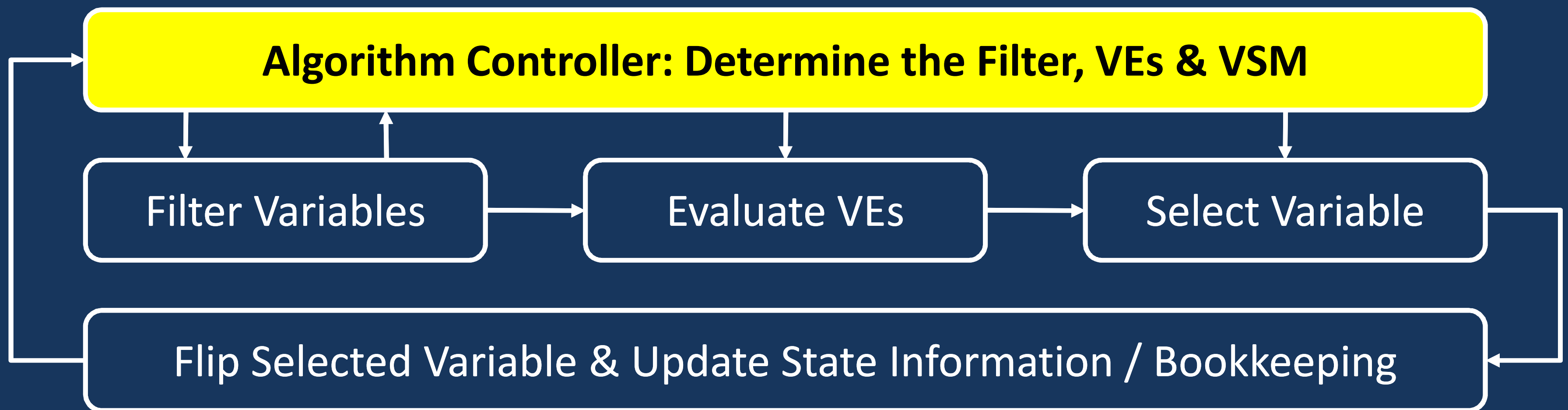
Our New SLS Model

Filter Variables → Evaluate VEs → Select Variable

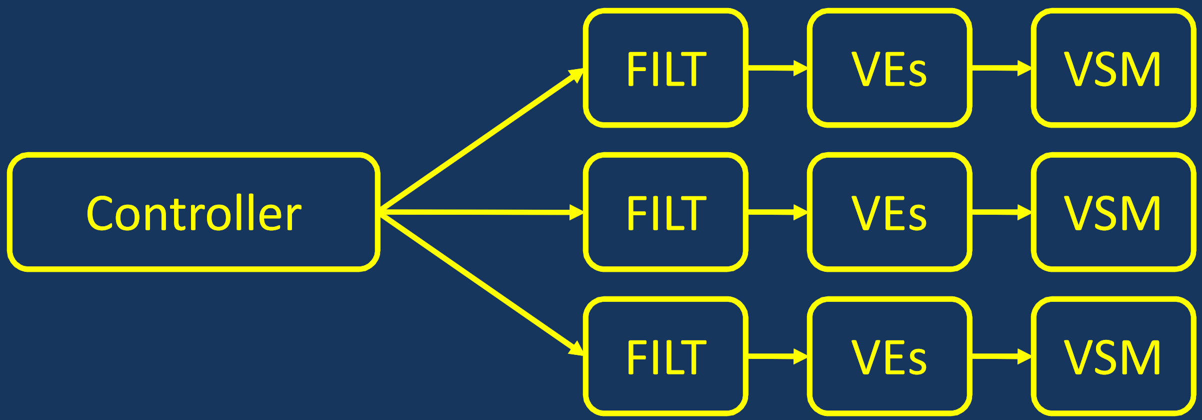
Our New SLS Model



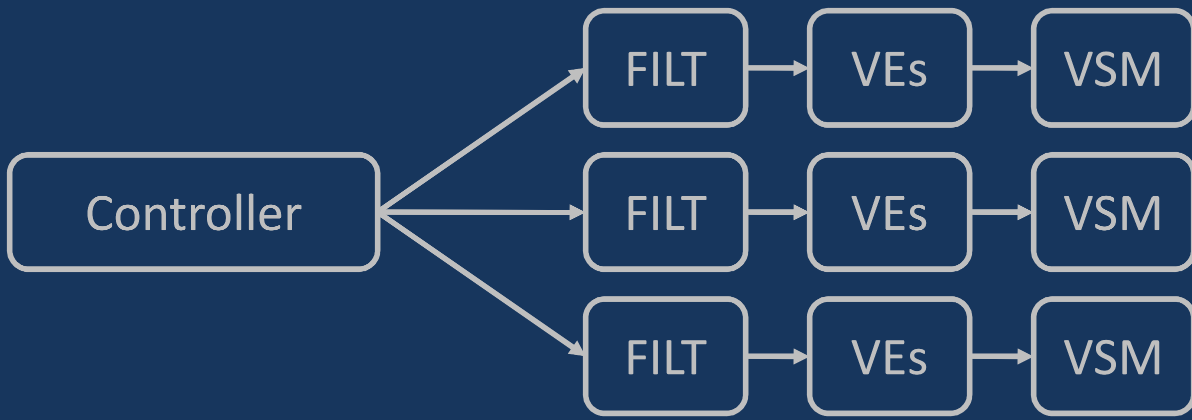
Our New SLS Model

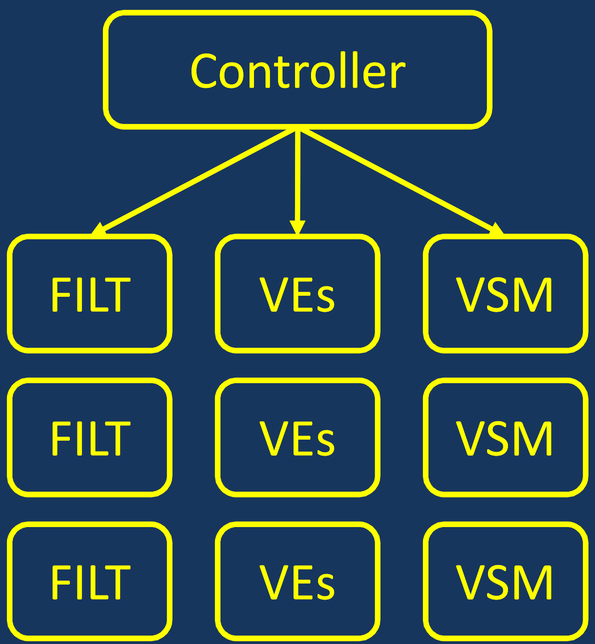


Algorithm Controllers

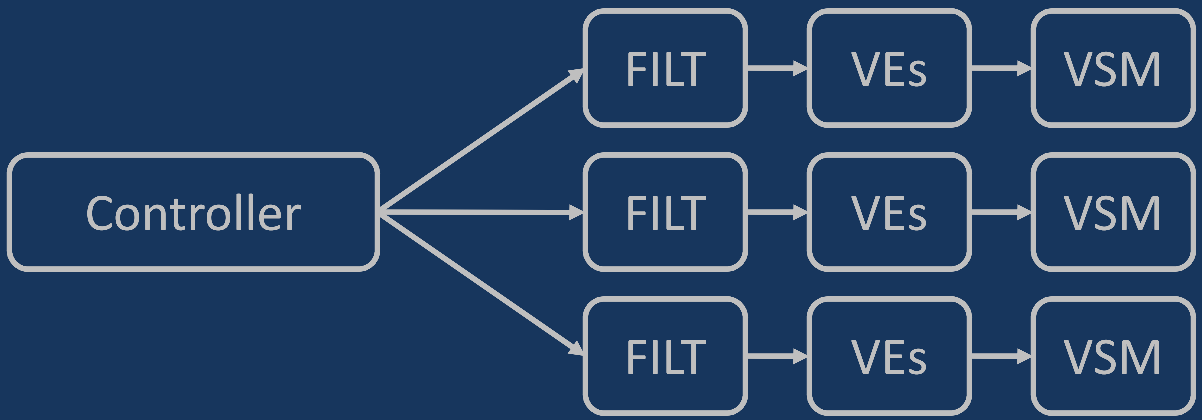


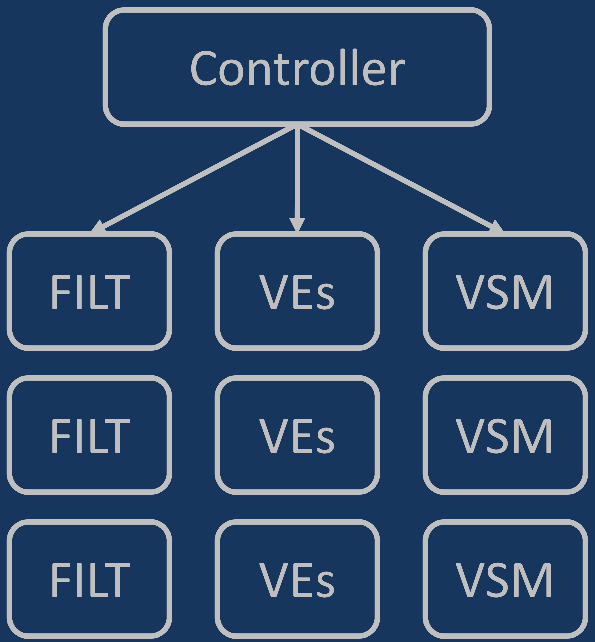
Algorithm Controllers

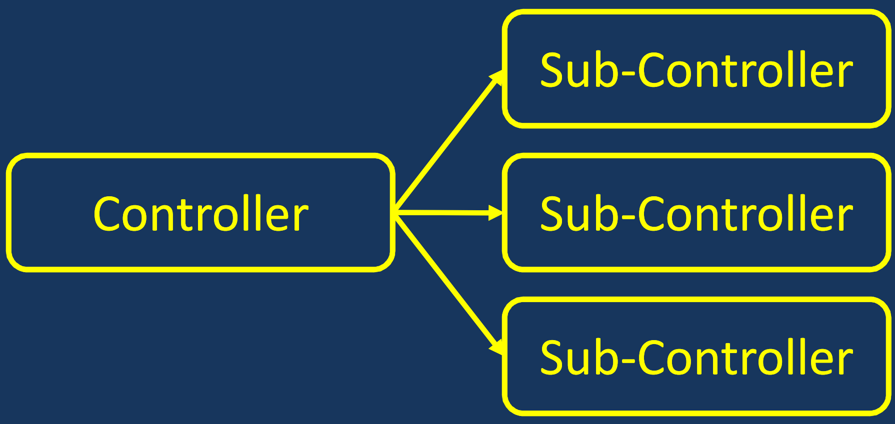




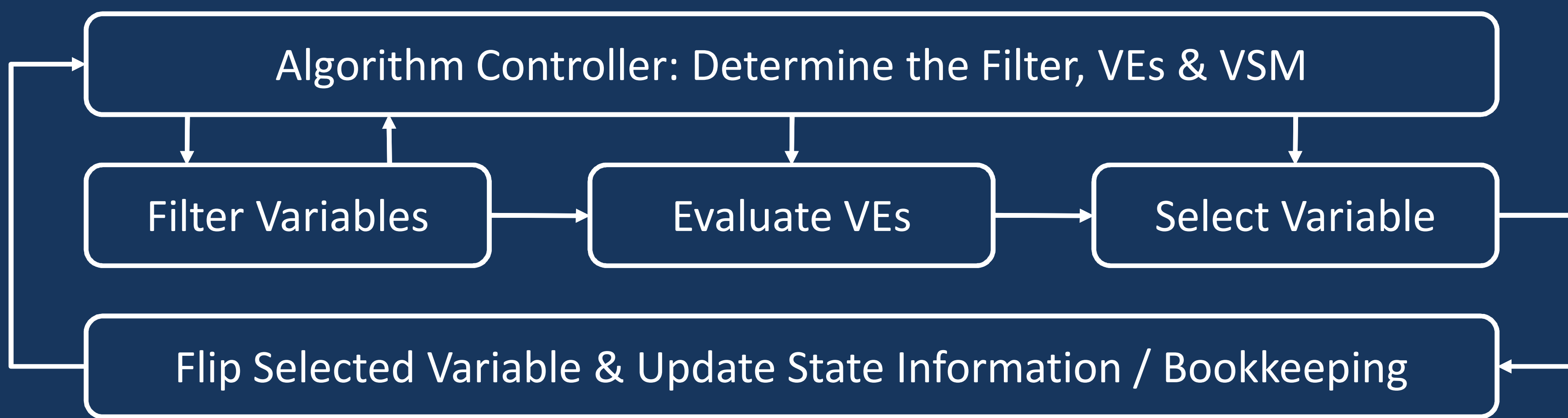
Algorithm Controllers







Our New SLS Model



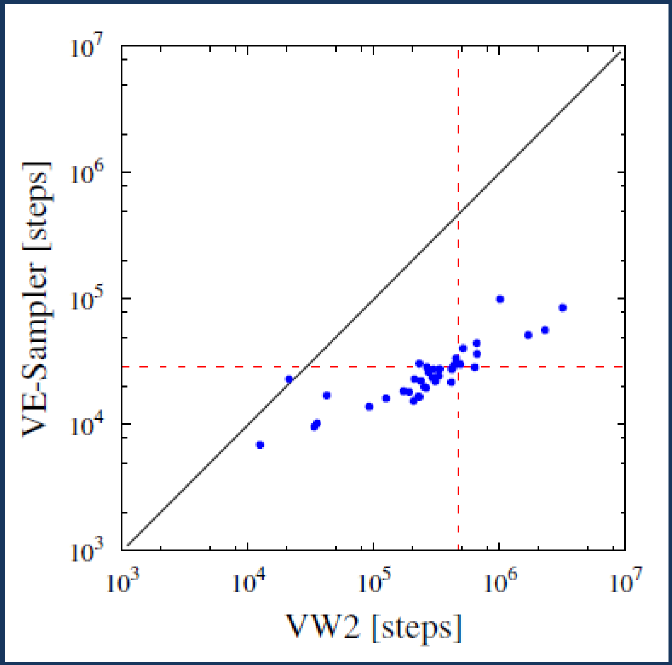
Software Implementation

- Design Architecture for Variable Expressions (DAVE)
  - Entire algorithm specified at runtime
    - Controllers, filters, VEs, selection mechanisms
  - Arbitrary complex VEs (interpreted)
  - Sophisticated macro system
    - Aids the use of automated configurators
- Extension of UBCSAT (2.0)

New Model & DAVE

- Concept of VEs
- New Model
- New Architecture

- Demonstrated our work in conjunction with an automated configurator



- Speedup factor:
16.2x (steps)
9.0x (time)

Key Contributions

1. Developed UBCSAT
2. Created SAPS, a Clause Penalty (CP) algorithm
3. Analyzed CP algorithm behaviour
4. Analyzed random decisions in SLS algorithms
5. Introduced a new conceptual model for SLS algorithms with Variable Expressions (VEs)
   – Developed a new Design Architecture (DAVE)

Primary Goal

"to advance the state-of-the-art
for SLS algorithms for SAT"

- *Explicitly:* develop new SLS algorithms that can outperform existing algorithms
- *Implicitly:* advance our understanding of current algorithms and introduce tools for developing new algorithms

Future Work

- Extend our methods to other domains
- Incorporate the use of automated tools
- Dynamic instances, distributed systems
- Generalized clause penalty solver
- Problem clauses & encodings
- New algorithm constructions

Selected Publications

- Dave A. D. Tompkins and Holger H. Hoos. Dynamic Scoring Functions with Variable Expressions: New SLS Methods for Solving SAT in SAT 2010, p. 278-292, 2010.

- Dave A.D. Tompkins and Holger H. Hoos, On the Quality and Quantity of Random Decisions in Stochastic Local Search for SAT in AI 2006, p. 146-158, 2006.
  [Awarded Best Paper]

- Dave A.D. Tompkins and Holger H. Hoos, UBCSAT: An Implementation and Experimentation Environment for SLS Algorithms for SAT and MAX-SAT in SAT 2004, p. 306-320, 2005. [Google Scholar citations: 62]

- Dave A. D. Tompkins and Holger H.Hoos. Warped Landscapes and Random Acts of SAT Solving in AI&M 2004. [Google Scholar citations: 25]

- Frank Hutter, Dave A. D. Tompkins, and Holger H. Hoos, Scaling and Probabilistic Smoothing: Efficient Dynamic Local Search for SAT in CP 2002, p. 233-248, 2002.
  [Google Scholar citations: 119]

Special Thanks To:

- Supervisor:
  - Holger H. Hoos

Special Thanks To:

- Committee members:
  - Will Evans, Alan Hu (& Lee Iverson)
- Co-Authors:
  - Holger H. Hoos & Frank Hutter
- Additional technical help
  - Kevin Smyth, Lin Xu, Chris Fawcett
- BETA lab members
- Proofreaders
- Family & friends

Special Thanks To:

Questions...