UBCSAT

An Implementation and Experimentation Environment for SLS Algorithms for SAT and MAX-SAT

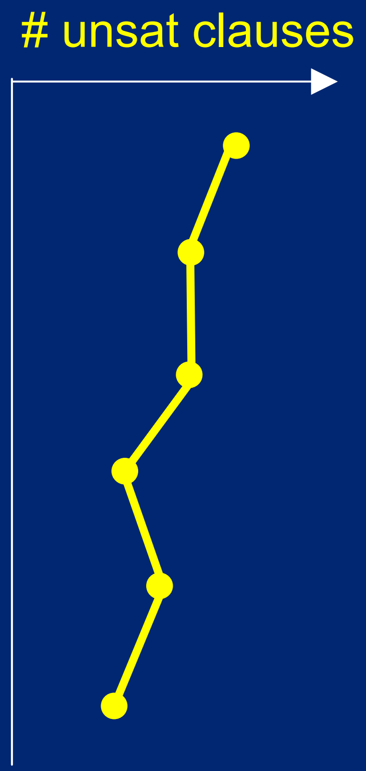
Dave Tompkins and Holger Hoos
University of British Columbia

SAT 2004   Vancouver, BC

Stochastic Local Search

- *Typical* Local Search Algorithm:

complete var. assignment:	clauses are sat or unsat:
00111010111010101010	(0)(0)(1)(0)(0)(1)(1)(1)
00111010111000101010	(0)(1)(1)(1)(0)(1)(1)(0)
00111010111000101011	(0)(1)(1)(1)(1)(1)(0)(0)
01111010111000101011	(1)(1)(1)(1)(1)(1)(0)(1)
01111011111000101011	(1)(0)(0)(1)(1)(1)(1)(1)
00111011101000101011	(1)(1)(1)(1)(1)(0)(1)(1)



WalkSAT/TABU Example

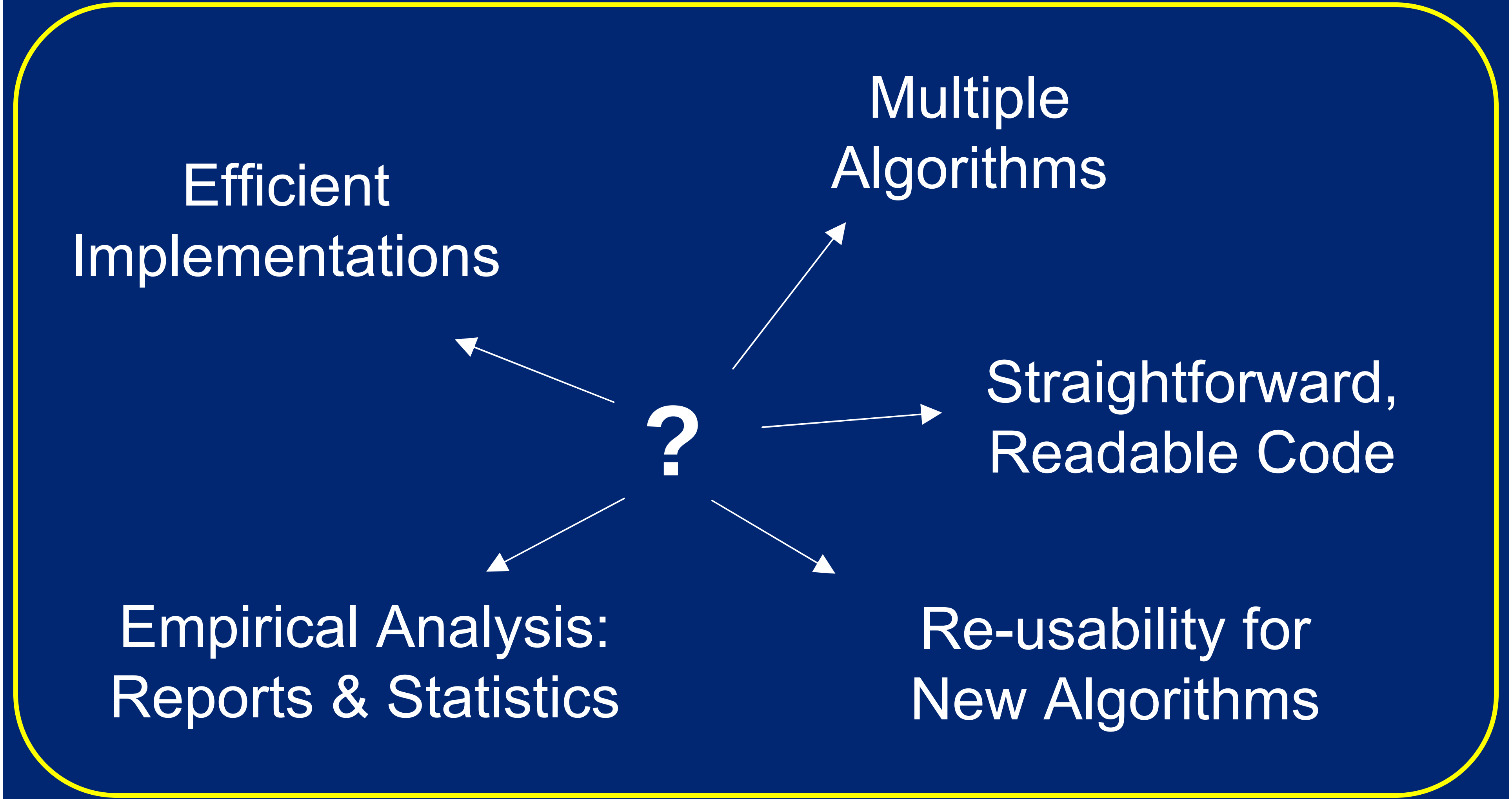
- Quick Refresher on WalkSAT/TABU
  McAllester, Selmen, Kautz (AAAI ’97)

- Search Step:
  – Select unsatisfied clause at random
  – Pick the “best” variable to flip that’s NOT tabu
  – Variables are tabu for *“TabuTenure”* steps

Objectives of UBCSAT:

1. Efficient, conceptually simple, and accurate implementations of existing SLS algorithms
2. Facilitate Integration of new algorithms (and variants)
3. Empirical analysis w/out compromising implementation efficiency
4. Explicit support for MAX-SAT
5. Open source
6. Platform-Independent, standard ANSI ‘C’

Design Challenge of UBCSAT:



Efficient Implementations

Multiple Algorithms

?

Straightforward, Readable Code

Empirical Analysis: Reports & Statistics

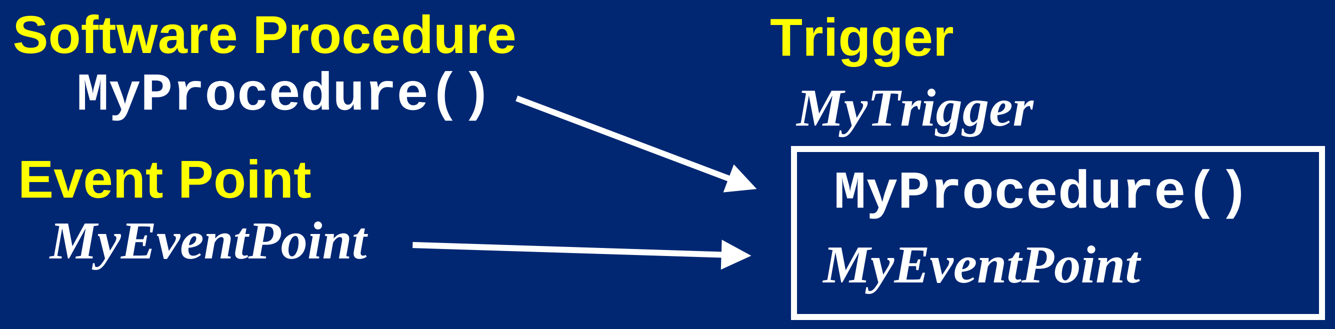
Re-usability for New Algorithms

Event Points

```
ReadInInstance
CreateData
CreateStateInfo
while (Run < NumRuns)
    while (Step < Cutoff) and (not TerminateRun)
        CheckRestart
        if Restart or Step = 1 then
            InitData
            InitStateInfo
        else
            ChooseCandidate
            PreFlip
            FlipCandidate
            PostFlip
        PostStep
        CheckTerminate
```

Triggered Procedure Architecture

- Triggers associate procedures & events



Software Procedure

`MyProcedure()`

Event Point

MyEventPoint

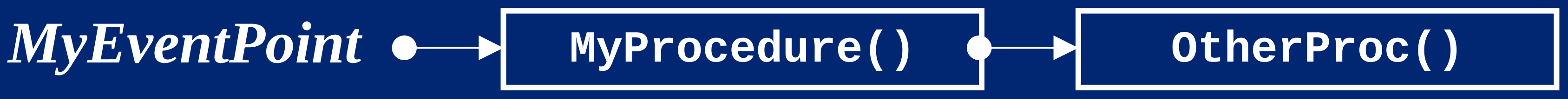
Trigger

MyTrigger

`MyProcedure()`

MyEventPoint

- *"Activating"* a trigger adds a procedure to an event list



MyEventPoint

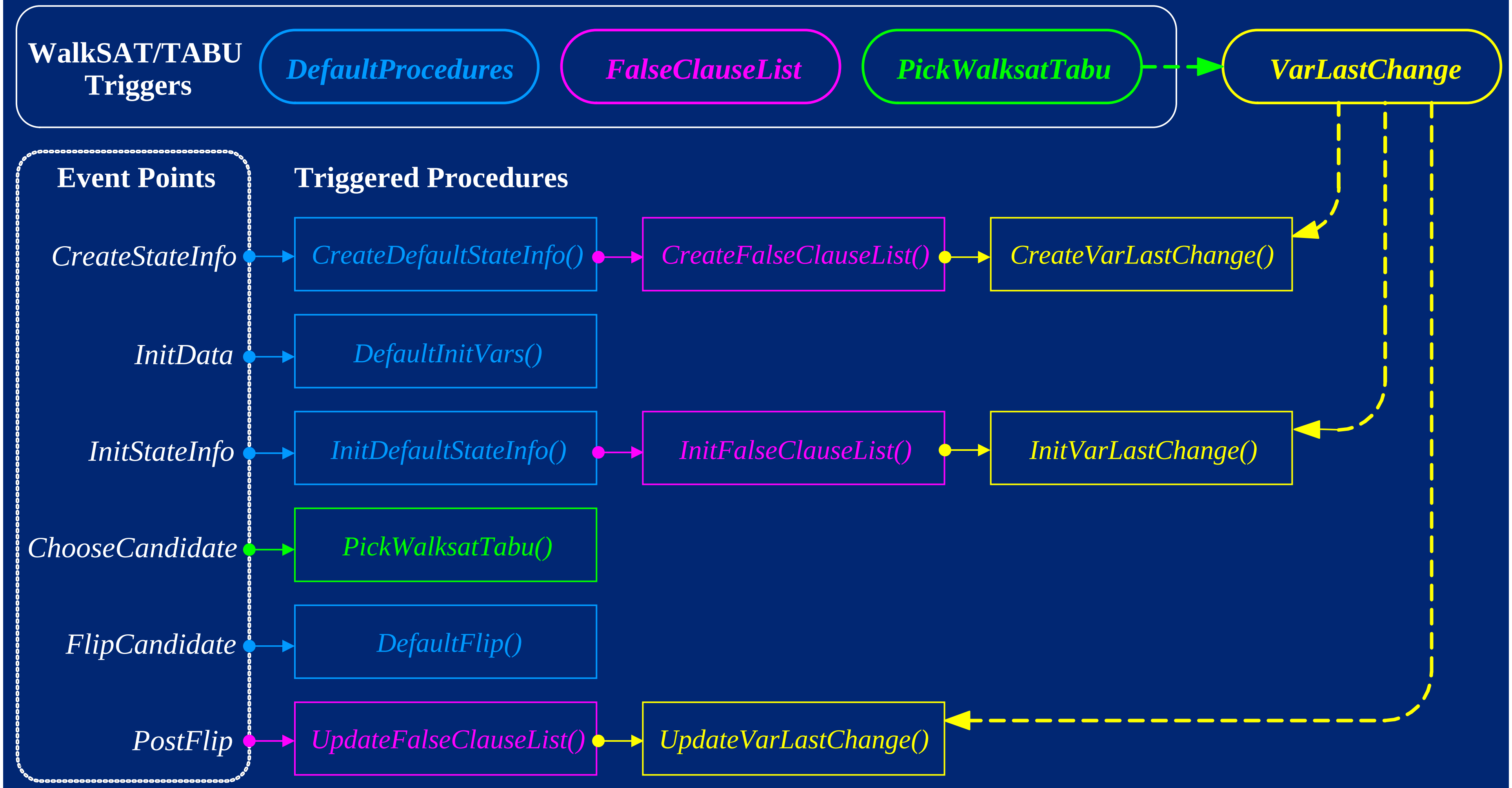
`MyProcedure()`

`OtherProc()`

Advanced Trigger features

- Container triggers
- Dependencies
- Precedence information
- Deactivation list

WalkSAT/TABU



WalkSAT/TABU Triggers

DefaultProcedures

FalseClauseList

PickWalksatTabu

VarLastChange

Event Points

Triggered Procedures

CreateStateInfo

CreateDefaultStateInfo()

CreateFalseClauseList()

CreateVarLastChange()

InitData

DefaultInitVars()

InitStateInfo

InitDefaultStateInfo()

InitFalseClauseList()

InitVarLastChange()

ChooseCandidate

PickWalksatTabu()

FlipCandidate

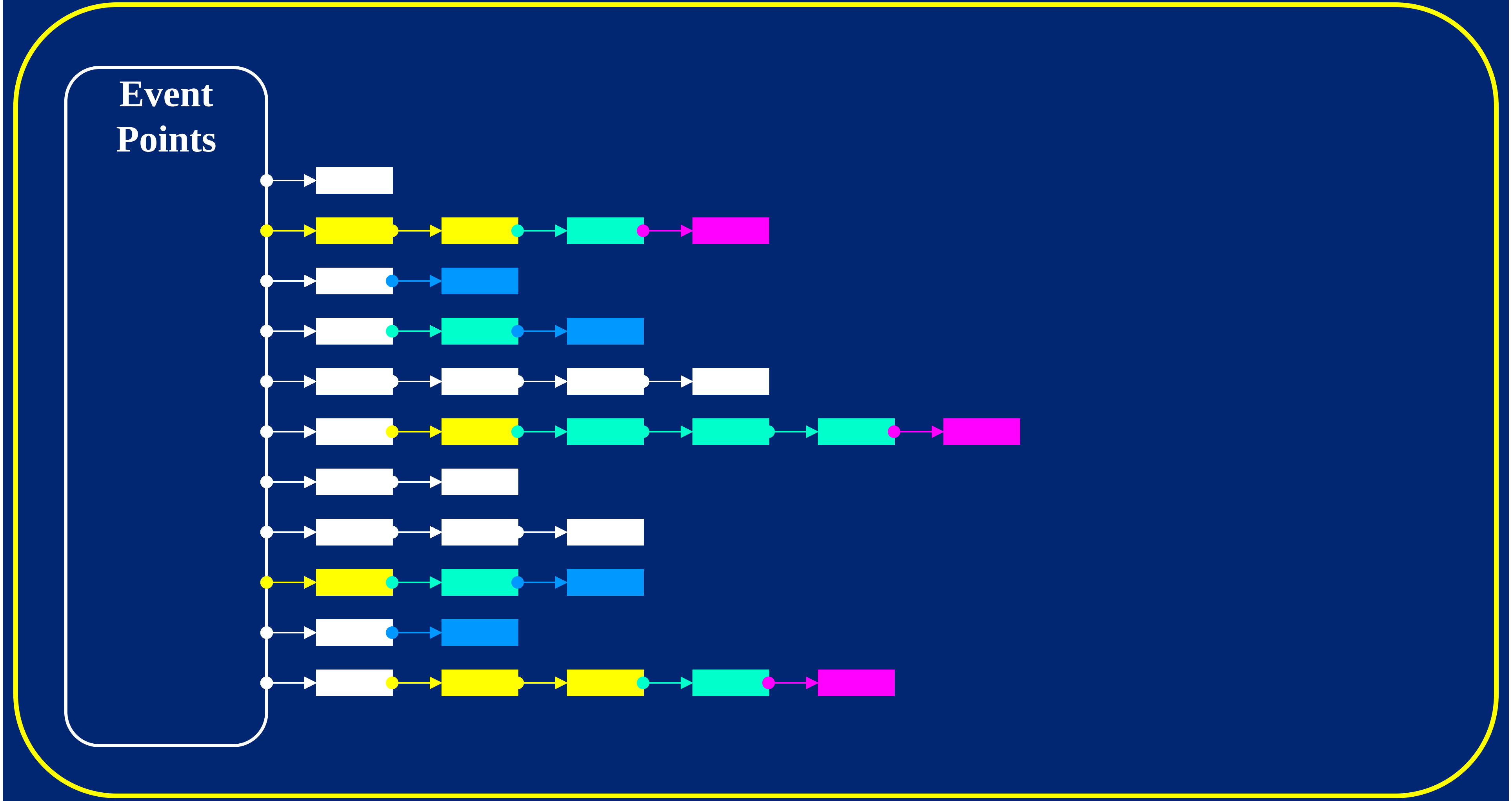
DefaultFlip()

PostFlip

UpdateFalseClauseList()

UpdateVarLastChange()

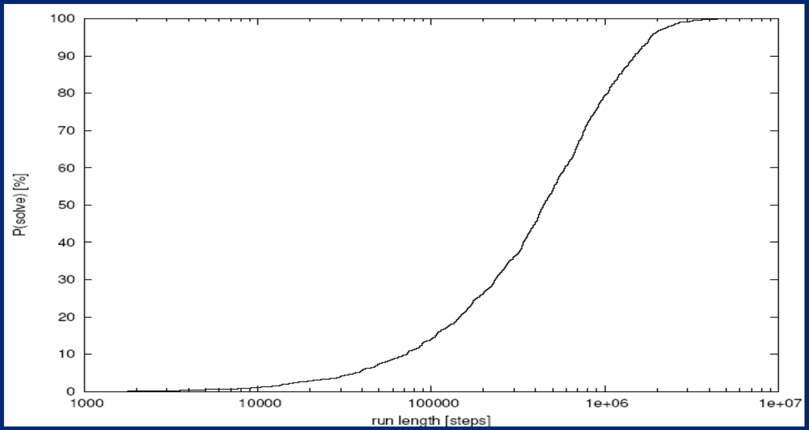
Configuration: Run-time Parameters

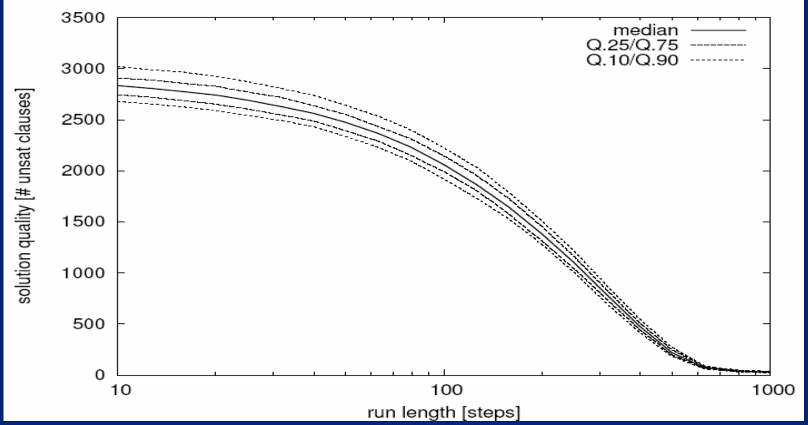


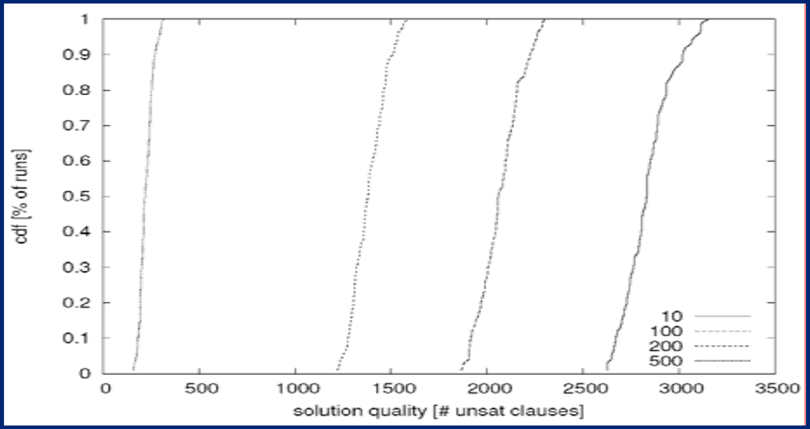
New Algorithm Example

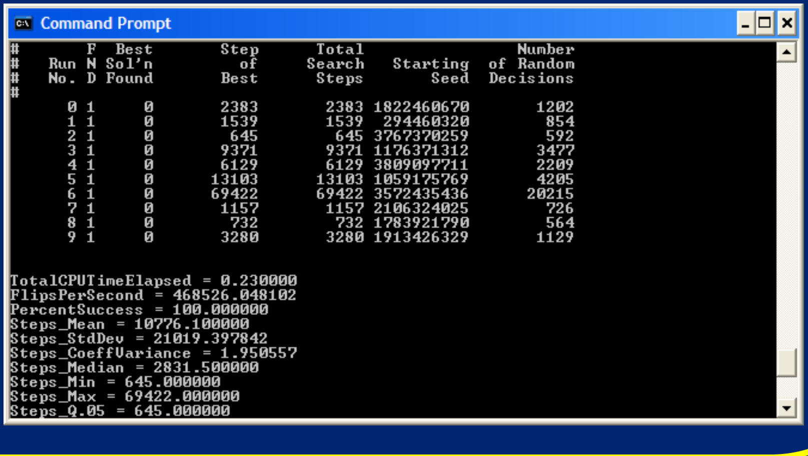
- Add new variant: WalkSAT/TABU-NoNull
  - If all variables in the selected clause are “tabu”, then pick one at random instead of a “null” flip
- Create new procedure: `PickWalksatTabuNoNull()`
- Add a new trigger `PickWalksatTabuNoNull()` • *ChooseCandidate*
- “Inherit” all of the data triggers and parameters from WalkSAT/TABU

Empirical Analysis









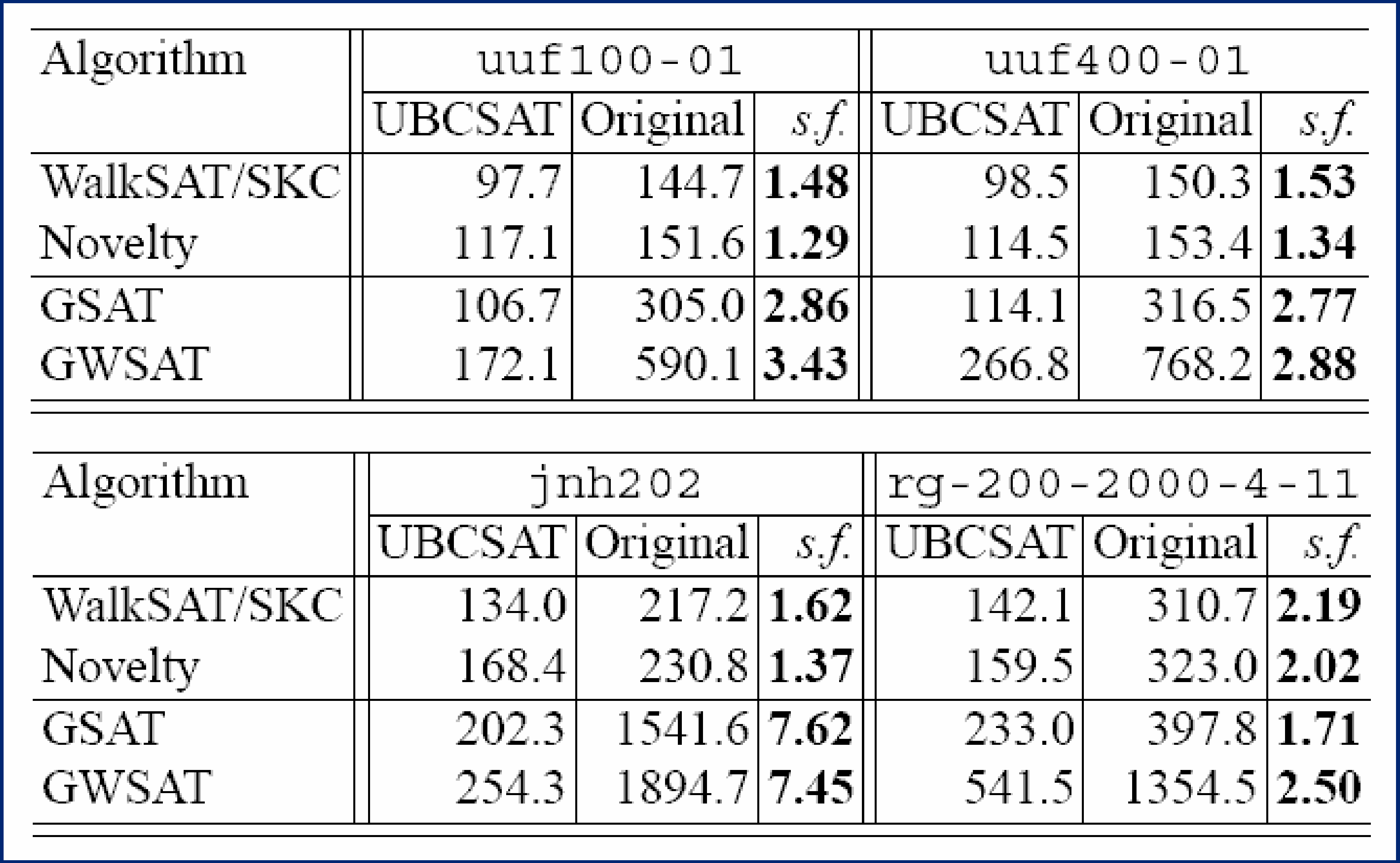
```
#      F  Best          Step       Total                  Number
#   Run N Sol'n           of      Search   Starting    of Random
#   No. D Found         Best       Steps       Seed    Decisions
#
      0 1     0         2383        2383 1822460670         1202
      1 1     0         1539        1539  294460320          854
      2 1     0          645         645 3767370259          592
      3 1     0         9371        9371 1176371312         3477
      4 1     0         6129        6129 3809097711         2209
      5 1     0        13103       13103 1059175769         4205
      6 1     0        69422       69422 3572435436        20215
      7 1     0         1157        1157 2106324025          726
      8 1     0          732         732 1783921790          564
      9 1     0         3280        3280 1913426329         1129

TotalCPUTimeElapsed = 0.230000
FlipsPerSecond = 468526.048102
PercentSuccess = 100.000000
Steps_Mean = 10776.100000
Steps_StdDev = 21019.397842
Steps_CoeffVariance = 1.950557
Steps_Median = 2831.500000
Steps_Min = 645.000000
Steps_Max = 69422.000000
Steps_Q.05 = 645.000000
```

UBCSAT 1.0 – Included Algorithms

- GSAT
- GWSAT
- GSAT/TABU
- HSAT
- HWSAT
- WalkSAT/SKC
- WalkSAT/TABU
- Novelty / R-Novelty
- Novelty$^+$ / R-Novelty$^+$
- Adaptive Novelty$^+$
- SAPS
- RSAPS
- SAPS/NR
- SAMD
- IRoTS

UBCSAT & Original Implementations



Algorithm	uuf100-01			uuf400-01		
	UBCSAT	Original	*s.f.*	UBCSAT	Original	*s.f.*
WalkSAT/SKC	97.7	144.7	**1.48**	98.5	150.3	**1.53**
Novelty	117.1	151.6	**1.29**	114.5	153.4	**1.34**
GSAT	106.7	305.0	**2.86**	114.1	316.5	**2.77**
GWSAT	172.1	590.1	**3.43**	266.8	768.2	**2.88**

Algorithm	jnh202			rg-200-2000-4-11		
	UBCSAT	Original	*s.f.*	UBCSAT	Original	*s.f.*
WalkSAT/SKC	134.0	217.2	**1.62**	142.1	310.7	**2.19**
Novelty	168.4	230.8	**1.37**	159.5	323.0	**2.02**
GSAT	202.3	1541.6	**7.62**	233.0	397.8	**1.71**
GWSAT	254.3	1894.7	**7.45**	541.5	1354.5	**2.50**

UBCSAT compared to GSAT v41 and WalkSAT v43 on UNSAT instances

Conclusions

- ✓ Efficient, simple, & accurate implementations of existing SLS algorithms
- ✓ Easy to add new algorithms & variants
- ✓ Advanced empirical analysis (reports & statistics) w/out compromising efficiency
- ✓ Open source & publically available: `http://www.satlib.org/ubcsat`

Future Work

- More algorithms
- More reports & statistics
- More features
- External support: gnuplot, ‘R’ scripts

- Please make suggestions
- Contact us to include your algorithm or “wish list”

- `email: ubcsat-help@cs.ubc.ca`
- `subscribe to: ubcsat-updates@cs.ubc.ca`