NiVER: Non Increasing Variable Elimination Resolution for preprocessing SAT Instances

Sathiamoorthy Subbarayan
IT-University of Copenhagen
Denmark

Dhiraj Pradhan
University of Bristol
UK

Resolution Inference Rule

$(a+b'+c)\quad (c'+d) \quad \Rightarrow \quad (a+b'+d)$

$(a+b+c)\quad (a'+b) \quad \Rightarrow \quad (b+c)$

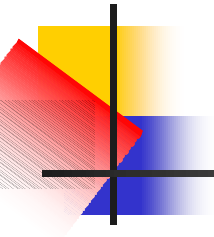
Subsumes (a+b+c)

$(a+b+c)\ (a+b'+c') \quad \Rightarrow (a+b+b')$

Tautology

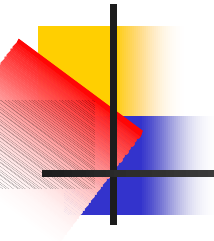
$(a+b+c)\ (a'+d)\ (b'+d) \Rightarrow (d+c)$

Hyper Resolution



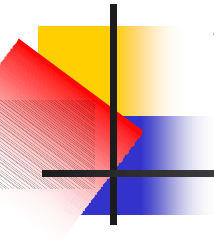
Davis Putnam Procedure

- Original algorithm for SAT solving (1960)
- Variable Elimination Resolution (VER)
- Add all resolvents of a variable and eliminate the clauses with the variable
- Variable Elimination preserves `satisfiability´



DPLL procedure

- Search based (1962)
- Basis for most of the modern SAT solvers
- State of the art SAT solvers augment clause learning, decision heuristics, etc., to DPLL



VER vs DPLL

Number of Iterations

- VER – Linear (N eliminations)
- DPLL – Exponential (2^N permutations)

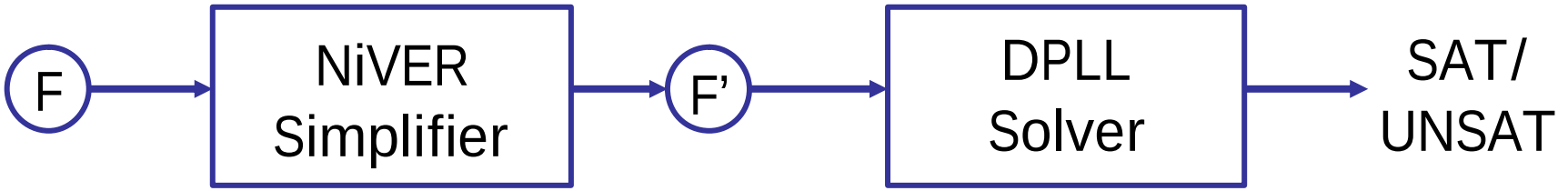
Space Complexity

- VER – Exponential
- DPLL – Linear

Time Complexity

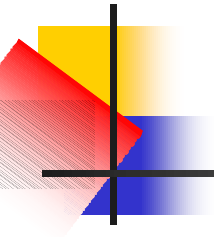
- VER – Exponential
- DPLL – Exponential

NiVER : Non increasing VER



Strategy : Use `linear´ VER to simplify the SAT problem, as much as possible without increase in space, then solve the result with DPLL search

- Number of Iterations : Linear
- Space Complexity : Linear



NiVER Preprocessing

- VER as a preprocessor to eliminate variables
- NiVER eliminates variable by VER only if there will be no increase in space
- Most of the real life instances have such variables, around 50% in many cases

NiVER Example

Positive clauses of variable C

(a'+ c)
(b'+ c)
(c + d')

Negative clauses of variable C

(a + b + c')
(e'+ f' + g' +c' +d')

Old_Num_Lits = 14
Number of clauses deleted = 5

Added Resolvents

(a' + e' + f' + g' + d)
(b' + e' + f' + g' + d)
(d' + a + b)

Discarded Resolvents (Tautologies)

(b' + a + b)
(a' + a + b)
(d' + e' + f' + g+ d)

New_Num_Lits = 13
Number of clauses added = 3

VER of 'C' results in one less variable, reduces number of literals and number of clauses, by one and two respectively.

NiVER Procedure

NiVER(F)

∀ V∈Var(F)

Pc = {C / C∈F, v∈C}, Nc = {C / C∈F, v'∈C}

R={}

∀ P∈Pc . ∀ N∈Nc

R=R∪Resolve(P,N)

Old_Num_Lits = No of literals in (Pc∪Nc)

New_Num_Lits = No of literals in R

if(New_Num_Lits £ Old_Num_Lits)

F=F-(Pc∪Nc), F=F+R

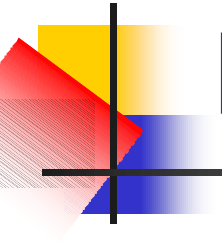
return F

Effect of NiVER Simplifier on fifo8_400

#Variables-Org :	259762	#Literals-Org :	1601865
#Variables-Pre :	68790 (-74%)	#Literals-Pre :	858776 (-46.4%)
#Clauses-Org :	707913		
#Clauses-Pre :	300842 (-58%)	NiVER Time :	14.3 Seconds

Best Worst Case Upper Bounds for SAT Solving in terms of

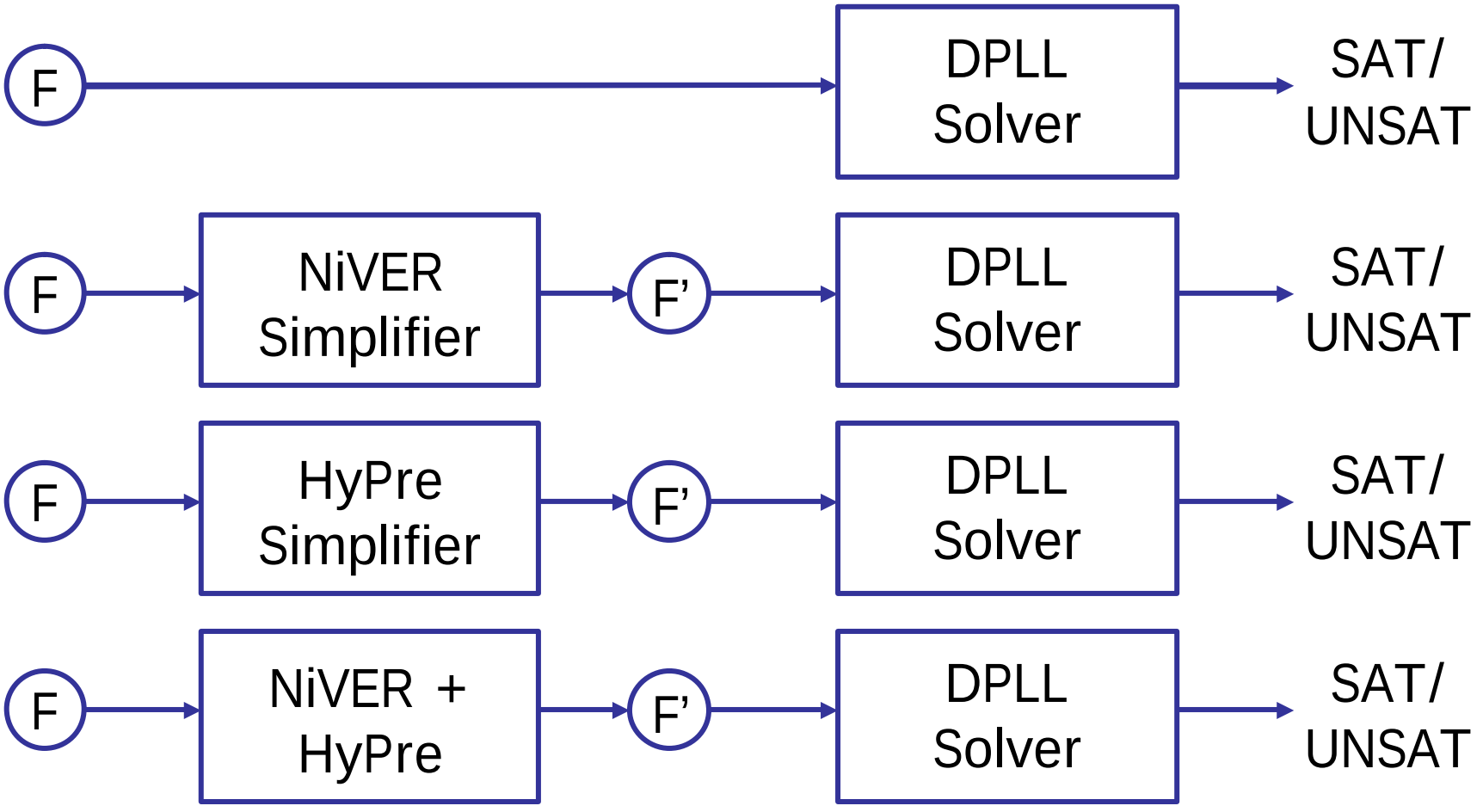
- Number of Variables (N) : 2^{N}
- Number of Clauses (K) : $2^{0.30897K}$
- Number of Literals (L) : $2^{0.10299L}$

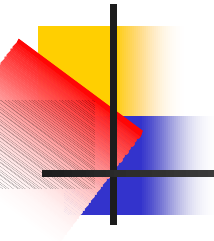


Effect of NiVER Simplifier

Benchmark	N-org	N-pre	%NRem	K-org	%KDec	L-org	%LDec	Time
6 pipe	15800	15067	5	394739	0.4	1157225	0.2	0.5
f2clk_40	27568	10408	**62**	80439	**45**	234655	**32.8**	1.3
ip50	66131	34393	**48**	214786	**31**	512828	**22.3**	5.2
fifo8_400	259762	68790	**74**	707913	**58**	1601865	**46.4**	14.3
cache_10	227210	129786	**43**	879754	**31**	2191576	**23.3**	20.1
longmult15	7807	3629	**54**	24351	**34**	58557	**21.6**	0.2
barrel9	8903	4124	**54**	36606	**43**	102370	**35.2**	0.4
w08_14	120367	69151	**43**	425316	24	1038230	**17.3**	5.45
guidance-1-k56	98746	45111	**54**	307346	**37**	757661	**27.0**	2.74

Experimental Setup





Experimental Setup

- Two state of the art DPLL SAT solvers are used: Berkmin and Siege
- Two solvers have different strategies in SAT solving, hence the effect of NiVER on them can be studied

Experiments with Berkmin

BenchMark	Berkmin	NiVER+Ber	N+H+Ber	HyPre+Ber
6pipe	**210**	222	392	395
6pipe_6_ooo	276	**253**	738	771
7pipe	**729**	734	1165	1295
9vliw_bp_mc	**90**	100	1010	1031
comb2	305	**240**	271	302
comb3	817	407	**337**	368
fifo8_ 300	16822	13706	**244**	440
fifo8_400	42345	1290	**667**	760
ip38	256	99	**52**	105
ip50	341	313	**87**	224
barrel9	106	39	**34**	114
barrel8	368	34	**10**	38
ibm-rule20_k45	5806	8423	**757**	1611

Experiments with Berkmin

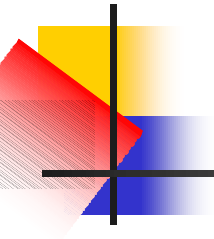
BenchMark	Berkmin	NiVER+Ber	N+H+Ber	HyPre+Ber
abp4-1-k31	1640	949	1056	**610**
avg-checker-5-34	1361	1099	**595**	919
guidance-1-k56	90755	17736	**14970**	22210
w08_14	3657	4379	**1381**	1931
ooo.tag14.ucl	18	**8**	399	1703
cache.inv14.ucl	36	**7**	396	2502
cache_05	3430	**1390**	2845	3529
cache_10	22504	55290	**12449**	15212
f2clk_30	100	61	**29**	53
f2clk_40	2014	1848	1506	**737**
longmult15	183	160	128	**54**
longmult12	283	233	180	**39**
cnt10	4170	2799	193	**134**

Experiments with Siege

Benchmark	Siege	NiVER+Sie	N+H+Sie	HyPre+Sie
6 pipe	79	**70**	360	361
6pipe_6_ooo	187	**156**	743	800
7pipe	185	**177**	1095	1183
9vliw_bp_mc	52	**46**	975	1014
comb2	407	266	**257**	287
comb3	550	419	396	**366**
fifo8_ 300	519	310	**229**	281
fifo8_400	882	657	**404**	920
ip38	146	117	**85**	115
ip50	405	258	**131**	234
barrel9	59	**12**	16	54
barrel8	173	25	**6**	16
ibm-rule20_k45	1537	1422	1308	**827**

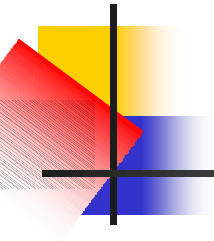
Experiments with Siege

Benchmark	Siege	NiVER+Sie	N+H+Sie	HyPre+Sie
abp4-1-k31	455	489	**303**	346
avg-checker-5-34	619	621	**548**	690
guidance-1-k56	9972	8678	**6887**	20478
w08_14	1251	**901**	1365	1931
ooo.tag14.ucl	15	**6**	396	1703
cache.inv14.ucl	39	**13**	396	2503
cache_05	238	**124**	2805	3540
cache_10	1373	**669**	10130	13053
f2clk_30	70	48	53	**41**
f2clk_40	891	988	802	**519**
longmult15	325	198	169	**54**
longmult12	471	256	292	**72**
cnt10	236	139	193	**134**



Results Summary

- NiVER+Berkmin beats Berkmin in 23/33 instances
- In cases where NiVER+Berkmin takes more time, mostly the difference is negligible
- NiVER+Siege beats Siege in 30/33 instances
- In the three cases where NiVER+Siege takes more time, the difference is negligible
- Poor performance on some instances is due to decision heuristics



Conclusion

- The advantage of VER was annulled by associated exponential space complexity
- NiVER does VER space efficiently, by not allowing space increasing resolutions
- Like clause learning and decision heuristics, NiVER can also be augmented to general purpose SAT solvers
- Sampling can be used to classify instances and efficiently solve them using proper simplifiers