An New Algorithm for Simultaneous Gate Sizing and V_t Assignment

Yifang Liu and Jiang Hu

Electrical and Computer Engineering

Texas A&M University

Agenda

- Introduction
- Algorithm Overview
- Relaxation and Restoration
- Iterative Refinement
- Experiment
- Conclusion

Circuit Optimization

- Gate sizing: common approach for circuit power-performance tradeoff
- Vt assignment: reduce leakage power without sacrificing performance
- Simultaneous gate sizing and Vt assignment
  - Simultaneous method offers better solutions than sequential
  - Gate sizing and Vt assignment can be easily integrated in simultaneous optimization

Problem Formulation

- Given a combinational logic circuit in DAG (Directed Acyclic Graph) G(V,E), select an implementation option for each gate to minimize the total power,

  $$\sum_{v_i \in V} p(v_i)$$

  subject to timing constraints,

  $$q(v_i) \geq a(v_i) \qquad \forall v_i \in V$$

Continuous v.s. Discrete Approach

- Continuous (most previous work on gate sizing)
  - + Fast
  - - Rounding error
  - - Hard to fit with lookup table model

- Discrete
  - - Slow to get good solution
  - + No rounding error
  - + Friendly to lookup table model

Our choice: discrete algorithm

Our effort: make it efficient

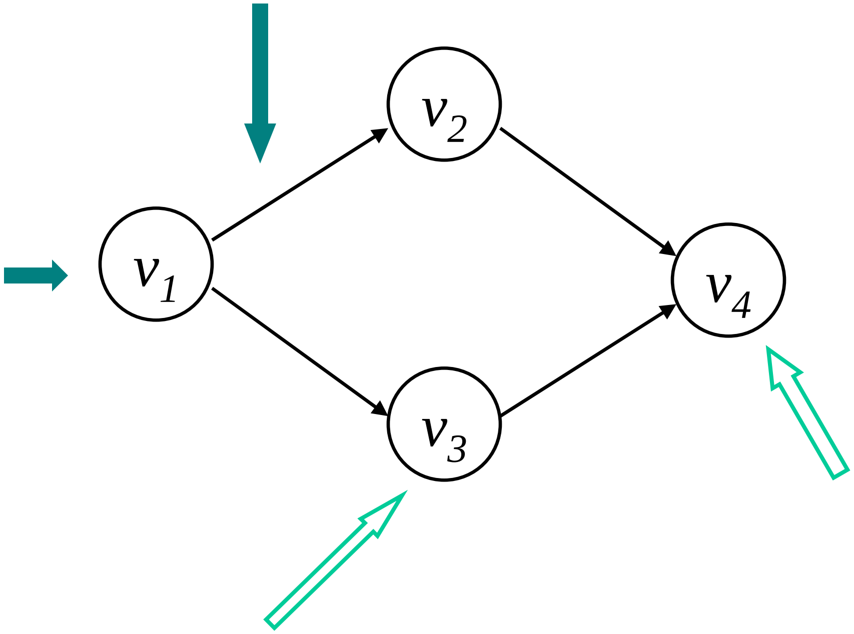
Discrete Simultaneous Gate Sizing and Vt Assignment

- Greedy heuristic (most previous work on Vt assignment)
  - Fast but non-ideal solution quality
- Simulated annealing
  - Slow, inefficient solution search
- Dynamic programming (DP)
  - Systematic solution search
  - Hard to handle reconvergence paths in DAG
  - Slow with multi-objective pruning
- Our approach
  - Based on DP-like search
  - Can handle reconvergent paths
  - Efficiently handle multiple objectives
  - Reasonable runtime

Agenda

- Introduction
- Algorithm Overview
- Relaxation and Restoration
- Iterative Refinement
- Experiment
- Conclusion

Notations



An interconnect

A gate

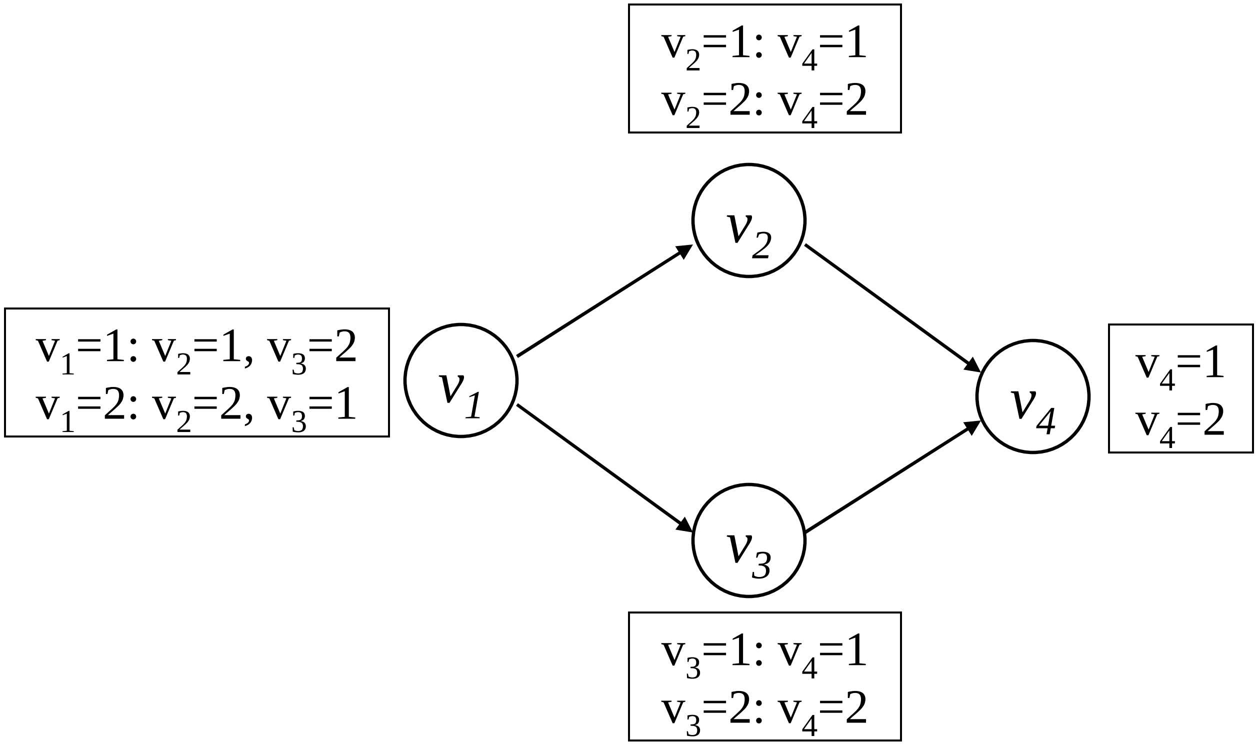
$v_4=k$

The k_{th} implementation option of v_4

$v_3=h : v_4=k$

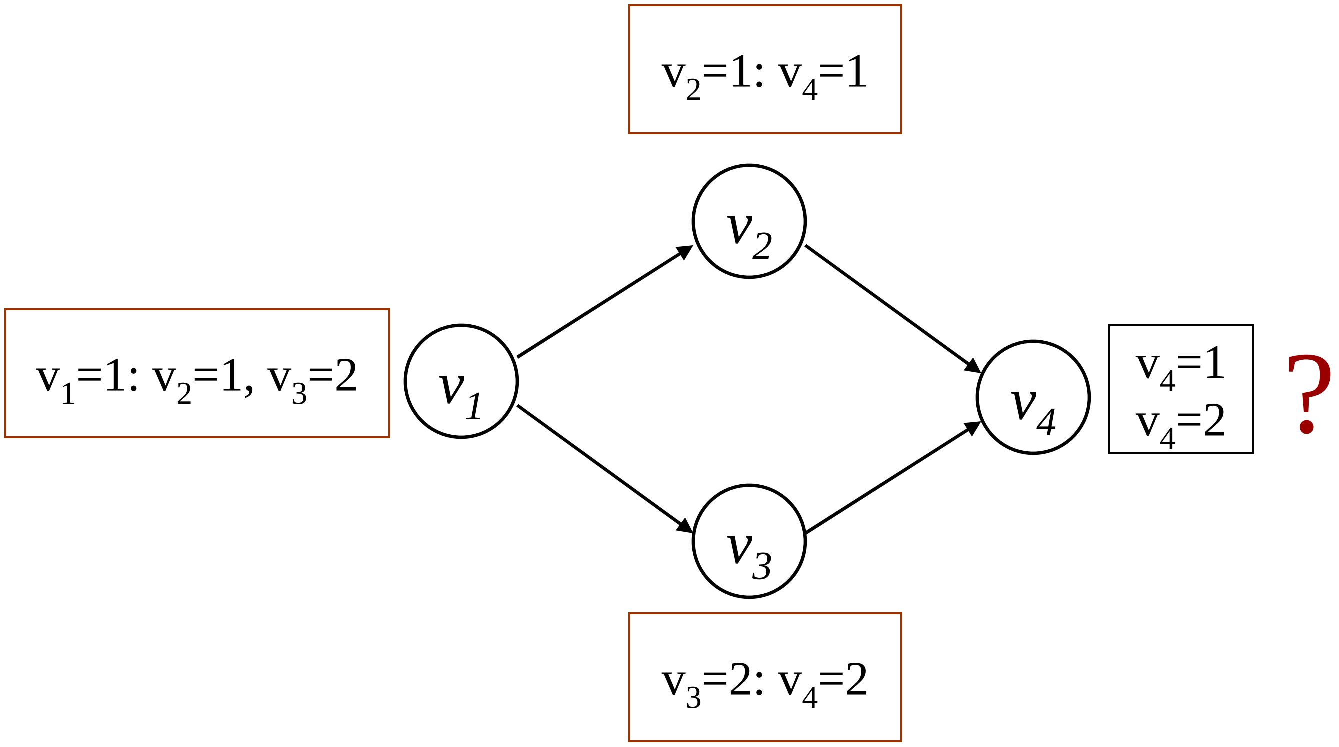
$v_3=h$ is implemented based on $v_4=k$

Difficulty1: Reconvergent Paths



Difficulty1: Reconvergent Paths

- History consistency needs to be warranted



Difficulty2: High Dimension Solution Space

- A partial solution is characterized by
  - Timing (q)
  - Loading effect (c)
  - Power (p)
- A solution is inferior if it is worse on all of q, c, p, and can be pruned
- Hard to prune in 3D solution space => many partial solutions => slow

Algorithm Strategy

- Reconvergent paths ⇐ consistency relaxation and iterative restriction
- High dimension solution space ⇐ Lagrangian relaxation [Chen, Chu and Wong, TCAD99]

Algorithm Overview

- Iterative relaxation and restriction

PHASE I: Global Optimization

- Consistency Relaxation by backward propagation;
- Consistency Restoration by forward propagation;

PHASE II: Iterative Refinement

- repeat
  - Backward Solution Search;
  - Forward Solution Search;

  until *no improvement*

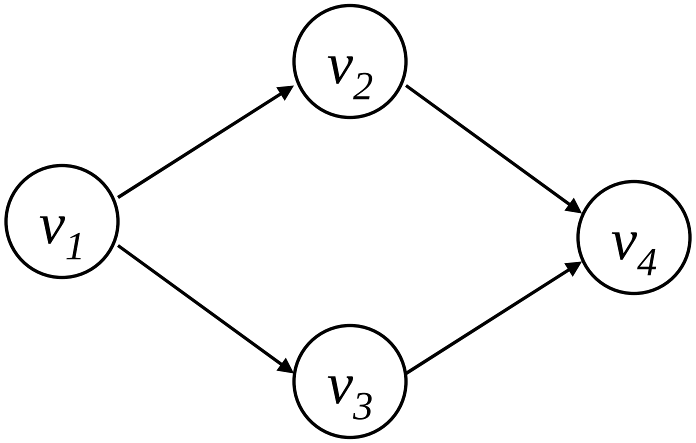
Agenda

- Introduction
- Algorithm Overview
- Relaxation and Restoration
- Iterative Refinement
- Experiment
- Conclusion

Consistency Relaxation

backward solution propagation

(v$_2$=1: v$_4$=1)[c=3, q=7]
(v$_2$=2: v$_4$=2)[c=2, q=5]

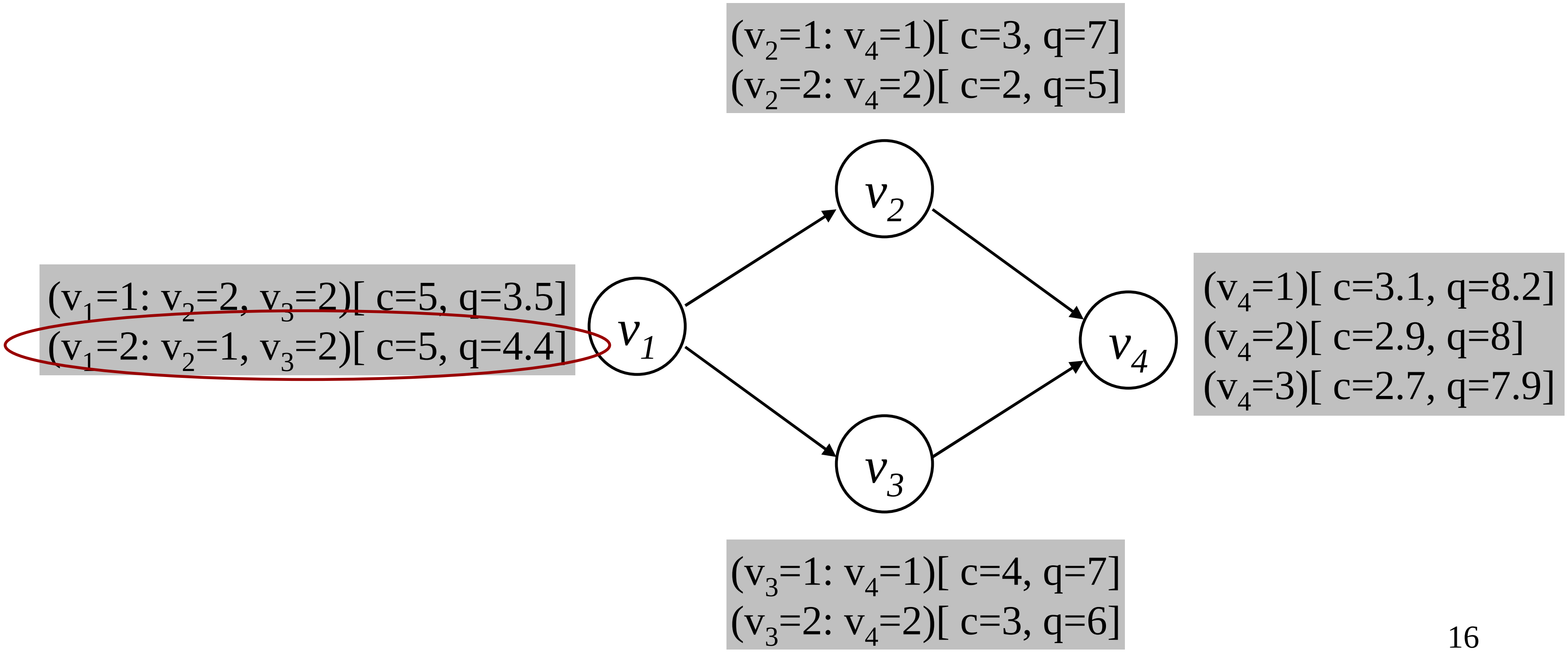


(v$_1$=1: v$_2$=2, v$_3$=2)[c=5, q=3.5]
(v$_1$=2: v$_2$=1, v$_3$=2)[c=5, q=4.4]

(v$_4$=1)[c=3.1, q=8.2]
(v$_4$=2)[c=2.9, q=8]
(v$_4$=3)[c=2.7, q=7.9]

(v$_3$=1: v$_4$=1)[c=4, q=7]
(v$_3$=2: v$_4$=2)[c=3, q=6]

Consistency Relaxation



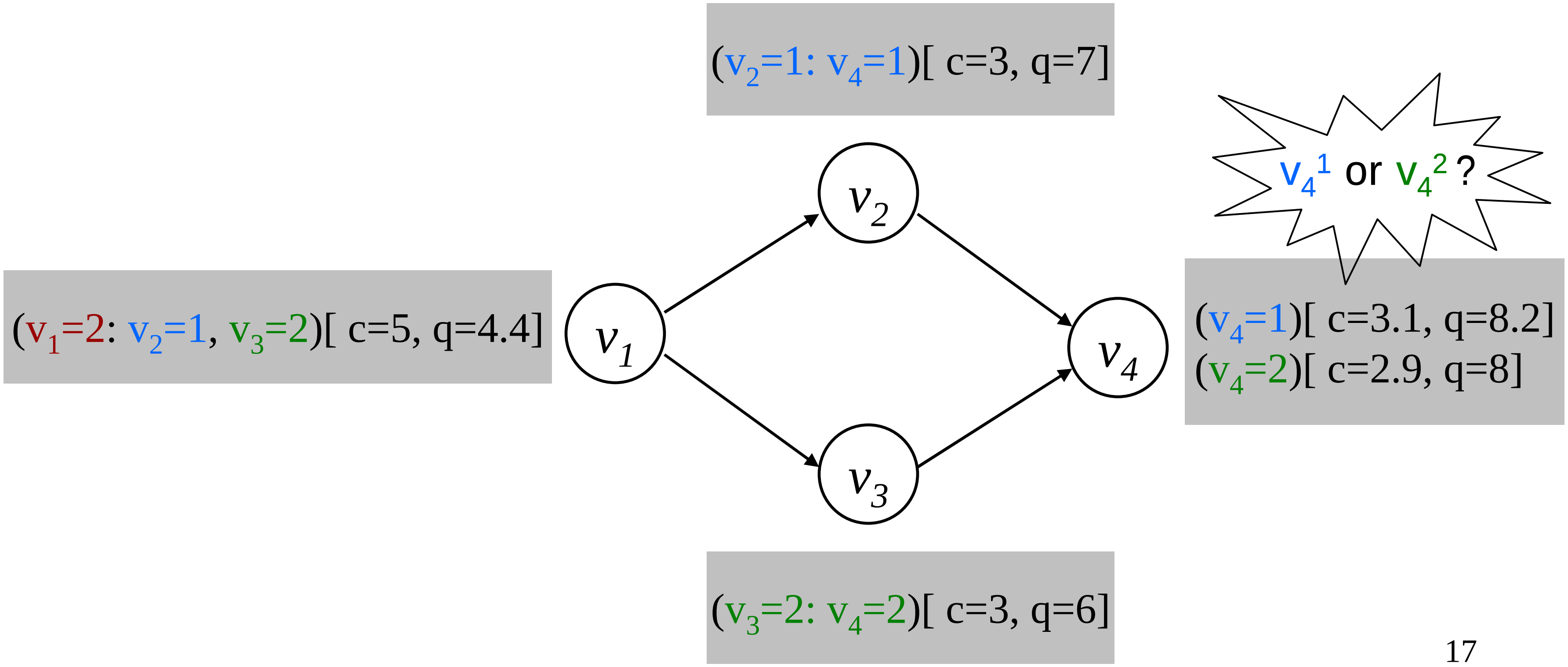
$(v_2=1: v_4=1)$[c=3, q=7]
$(v_2=2: v_4=2)$[c=2, q=5]

$(v_1=1: v_2=2, v_3=2)$[c=5, q=3.5]
$(v_1=2: v_2=1, v_3=2)$[c=5, q=4.4]

$(v_4=1)$[c=3.1, q=8.2]
$(v_4=2)$[c=2.9, q=8]
$(v_4=3)$[c=2.7, q=7.9]

$(v_3=1: v_4=1)$[c=4, q=7]
$(v_3=2: v_4=2)$[c=3, q=6]

Consistency Relaxation



(v_2=1: v_4=1)[c=3, q=7]

v_4^1 or v_4^2 ?

(v_1=2: v_2=1, v_3=2)[c=5, q=4.4]

(v_4=1)[c=3.1, q=8.2]
(v_4=2)[c=2.9, q=8]

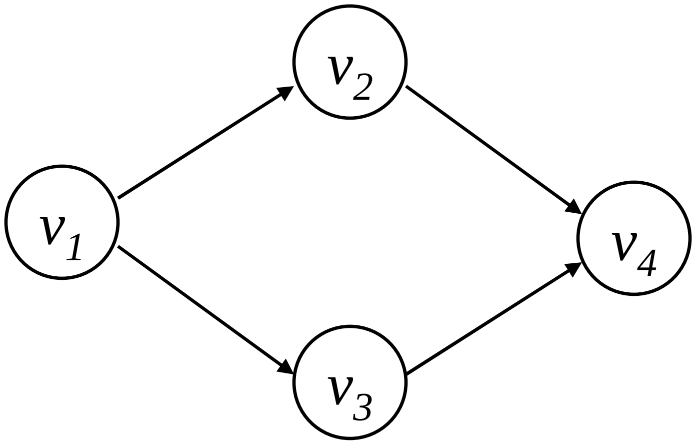
(v_3=2: v_4=2)[c=3, q=6]

Consistency Restoration

⇨ forward solution propagation

(v_2=1) [a=3.6]

(v_1=2) [a=2]



(v_3=2) [a=3.6]

(v_4=1) [a= ,q=8.2]
(v_4=2) [a= ,q=8]
(v_4=3) [a= ,q=7.9]

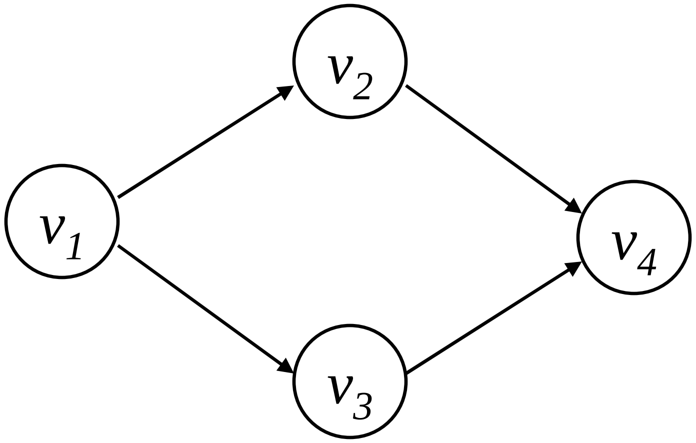
D(v_i,v_j)	V_4=1	V_4=2	V_4=3
V_2=1	1.2	3	2.2
V_3=2	3	2	2.2

Consistency Restoration

⇨ forward solution propagation

(v_2=1) [a=3.6]

(v_1=2) [a=2]



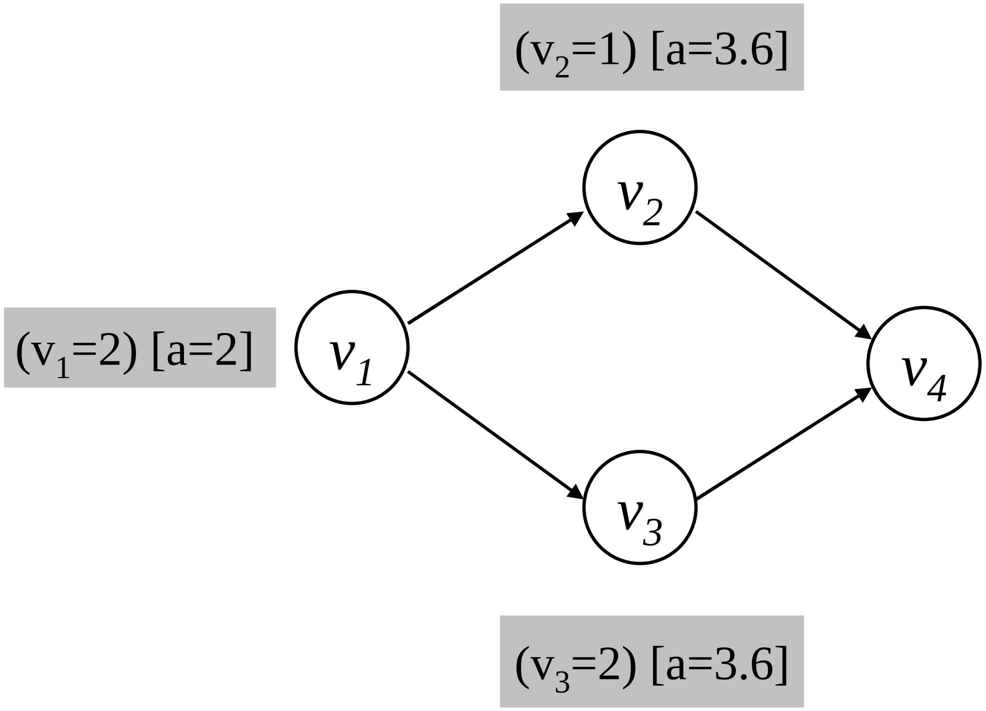
(v_4=1) [a=6.6 ,q=8.2]
(v_4=2) [a=6.6 ,q=8]
(v_4=3) [a=5.8 ,q=7.9]

(v_3=2) [a=3.6]

$D(v_i,v_j)$	V_4=1	V_4=2	V_4=3
V_2=1	1.2	3	2.2
V_3=2	3	2	2.2

Consistency Restoration

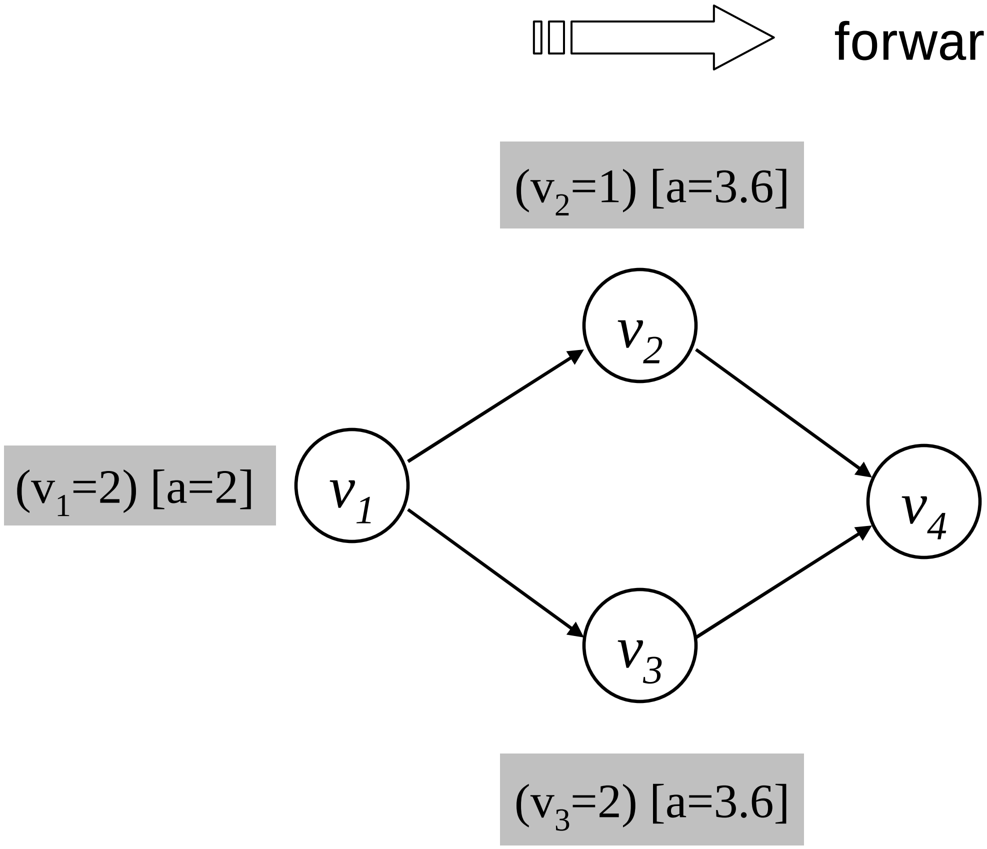
⇨ forward solution propagation



$(v_4=1)$ [a=6.6 ,q=8.2]
$(v_4=2)$ [a=6.6 ,q=8]
$(v_4=3)$ [a=5.8 ,q=7.9]

$D(v_i,v_j)$	$V_4=1$	$V_4=2$	$V_4=3$
$V_2=1$	1.2	3	2.2
$V_3=2$	3	2	2.2

Consistency Restoration



forward solution propagation

(v_2=1) [a=3.6]

(v_1=2) [a=2]

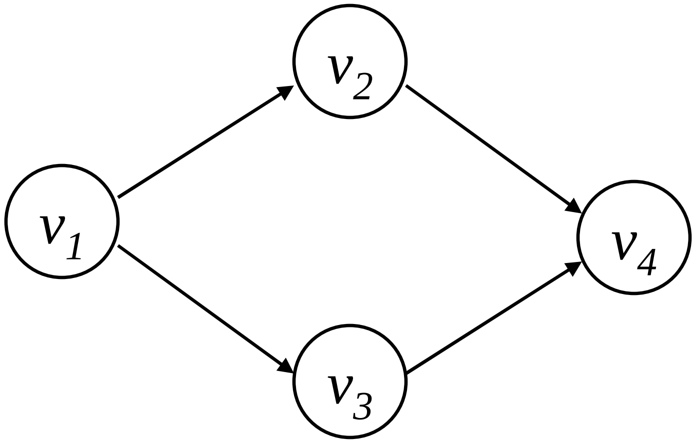
(v_3=2) [a=3.6]

(v_4=1) [a=6.6 ,q=8.2]
(v_4=2) [a=6.6 ,q=8]
(v_4=3) [a=5.8 ,q=7.9]

D(v_i,v_j)	V_4=1	V_4=2	V_4=3
V_2=1	1.2	3	2.2
V_3=2	3	2	2.2

Consistency Restoration

$(v_2=1)$ [a=3.6]



$(v_1=2)$ [a=2]

$(v_4=1)$ [a=6.6 ,q=8.2]
$(v_4=2)$ [a=6.6 ,q=8]
$(v_4=3)$ [a=5.8 ,q=7.9]

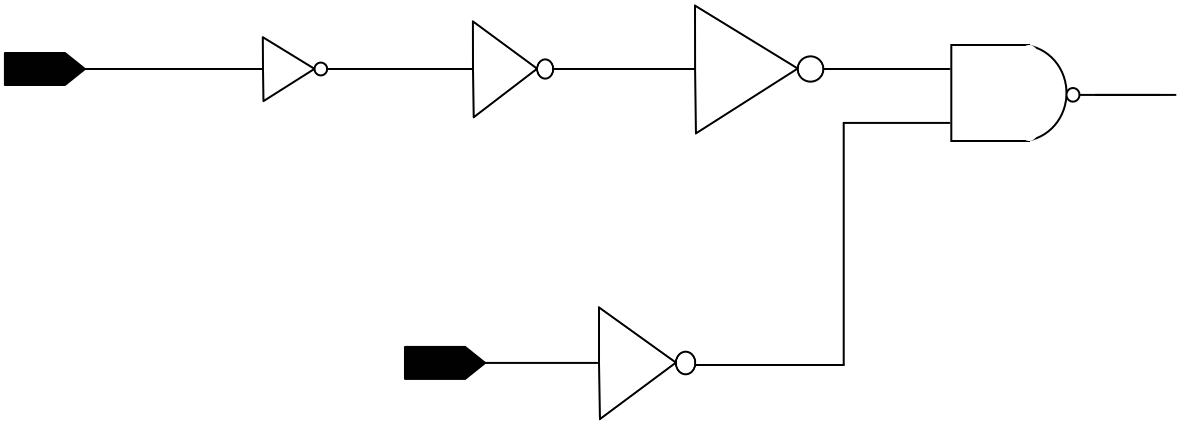
$(v_3=2)$ [a=3.6]

Agenda

- Introduction
- Algorithm Overview
- Relaxation and Restoration
- Iterative Refinement
- Experiment
- Conclusion

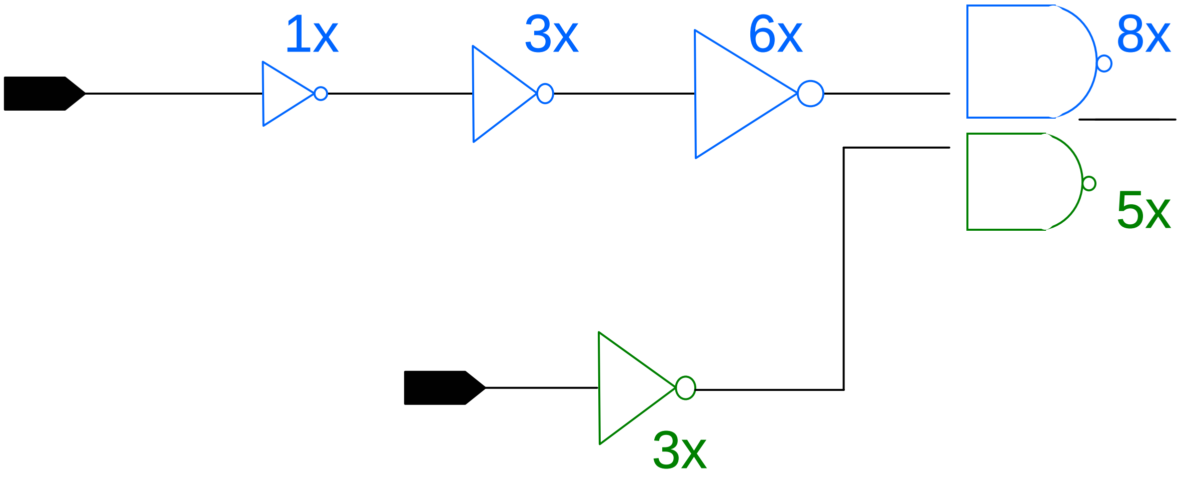
Iterative Refinement

Circuit in consideration



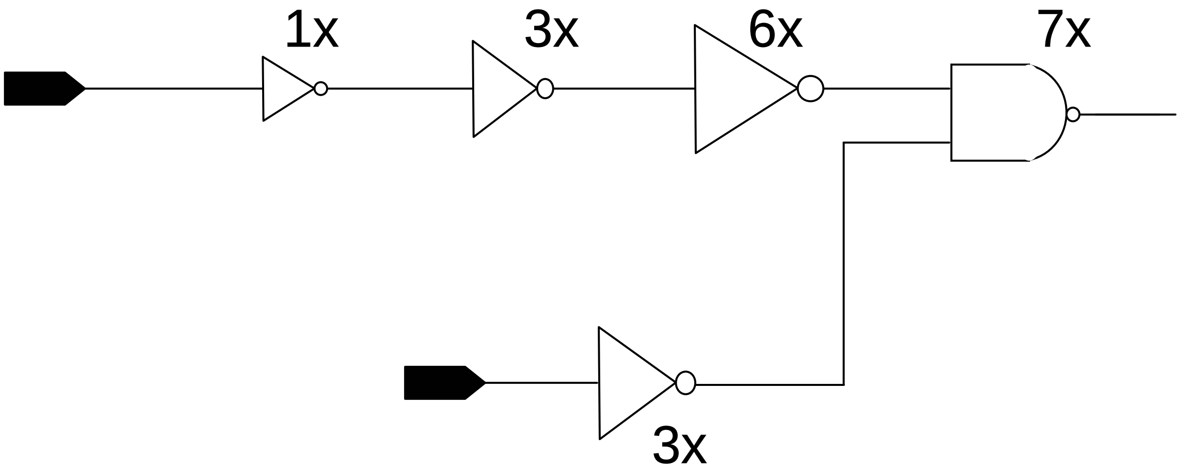
Iterative Refinement

After relaxation



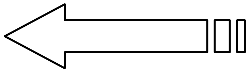
Iterative Refinement

After Phase I restoration

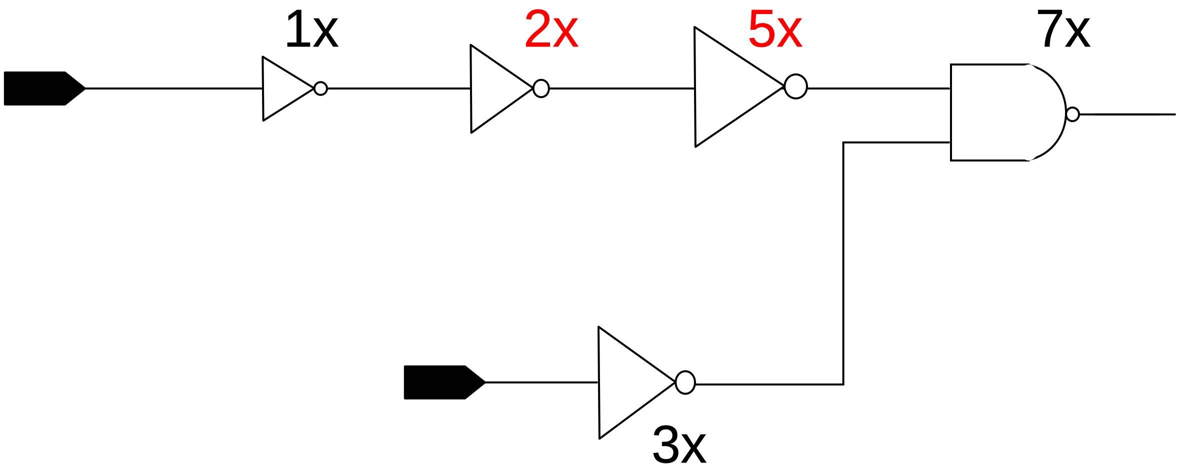


Iterative Refinement

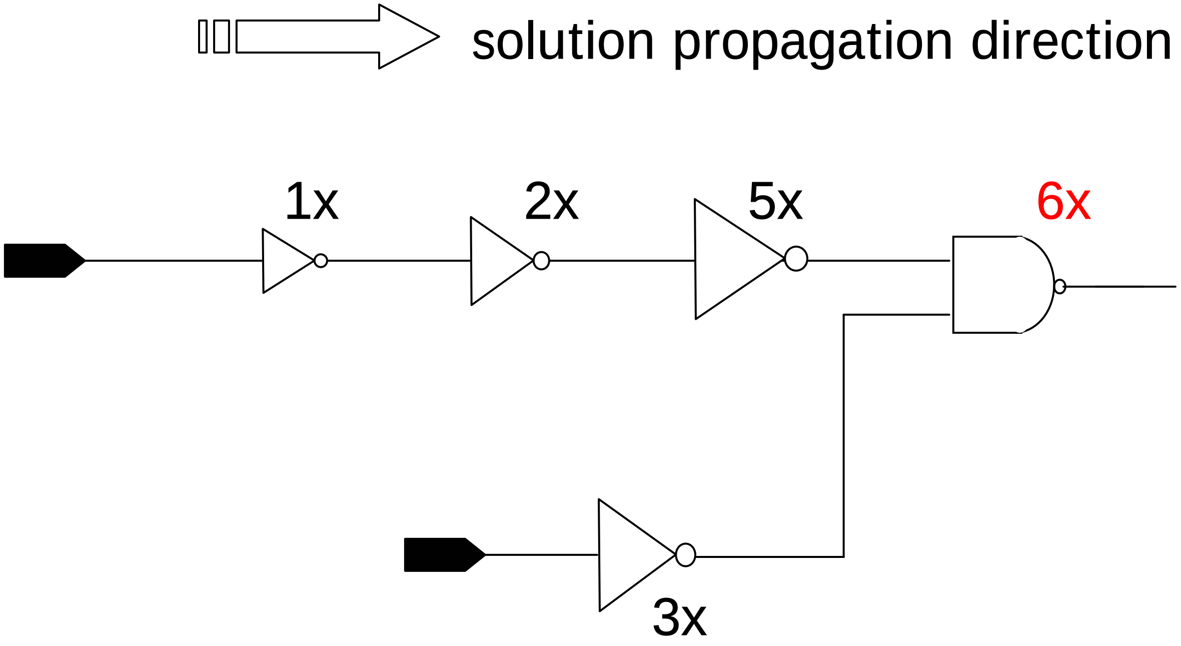
Phase II begins



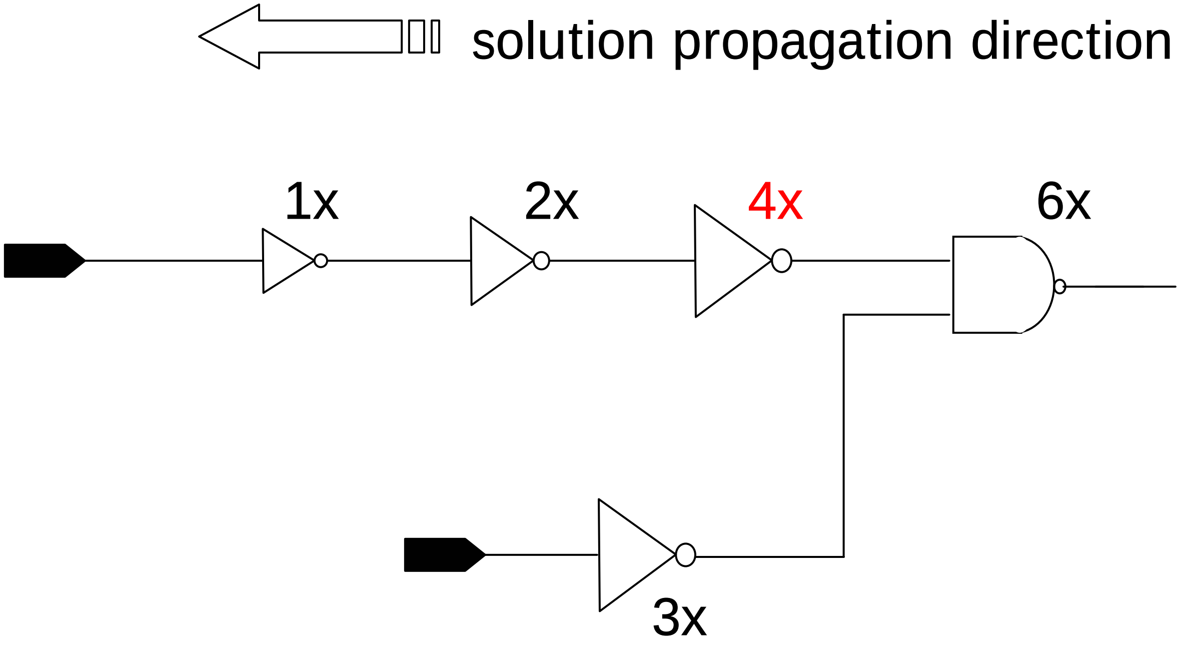
solution propagation direction



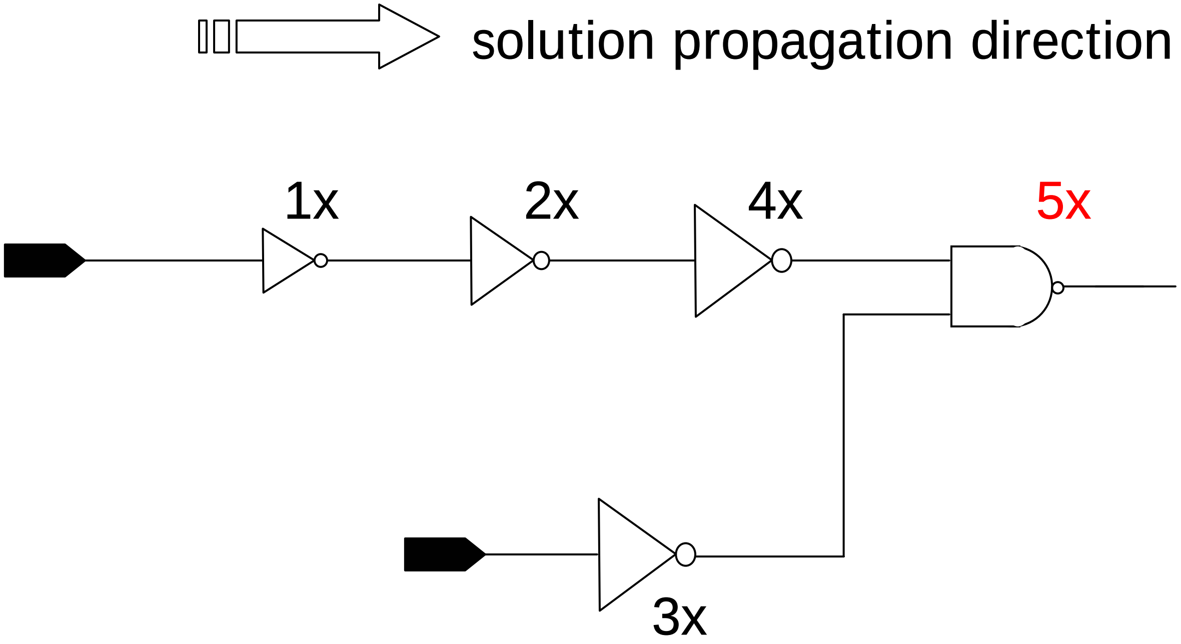
Iterative Refinement



Iterative Refinement

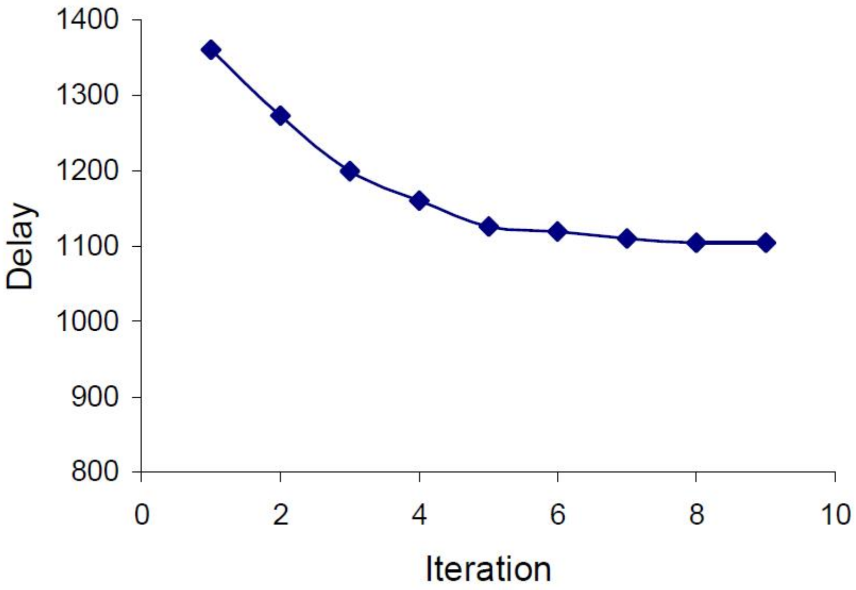

solution propagation direction

Iterative Refinement


solution propagation direction

Iterative Refinement

- Monotonic improvement of solution by iterative refinement



Agenda

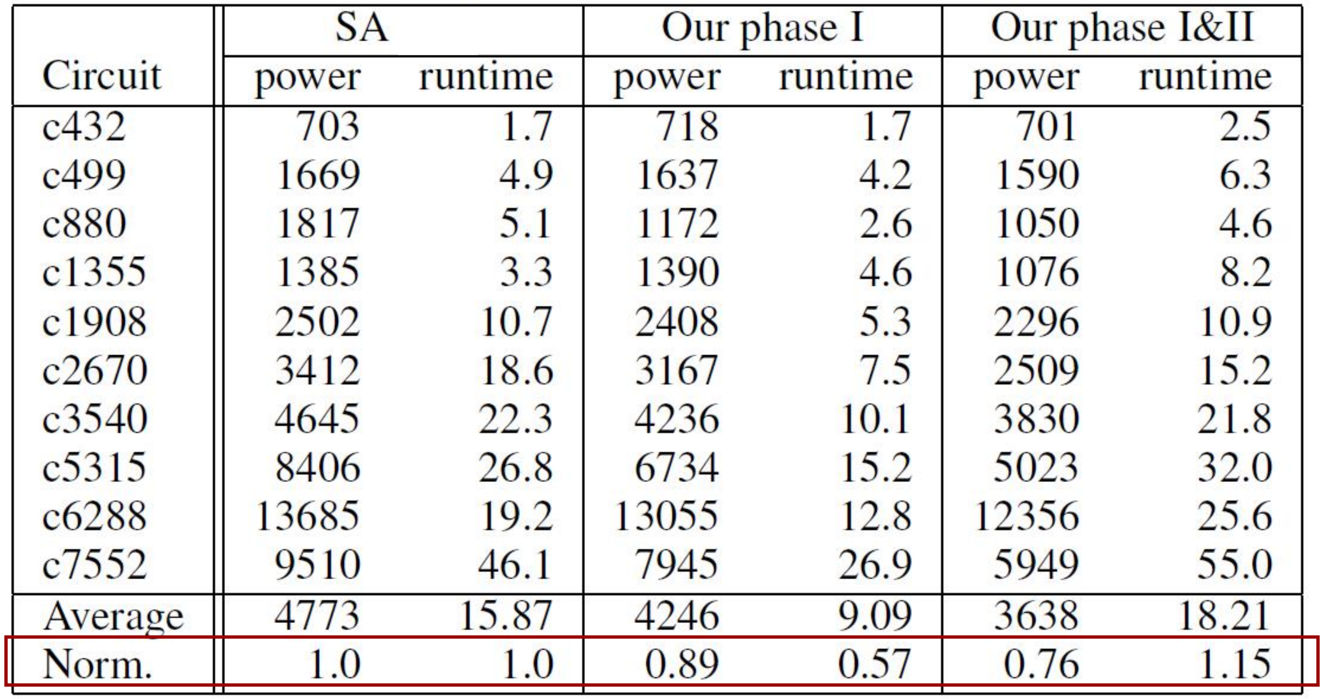
- Introduction
- Algorithm Overview
- Relaxation and Restoration
- Iterative Refinement
- Experiment
- Conclusion

Experiment Setup

- ISCAS85' benchmark circuit
- 70nm technology
- 4 V_t levels, 7 size options for each gate
- 3GHz Pentium CPU, 2GB Memory
- Compared with a previous work based on slack-allocation (SA) [Nguyen, et al., ISLPED03]

Experimental Result

- Comparison on power (*uW*) and CPU runtime (seconds). All solutions satisfy timing constraints

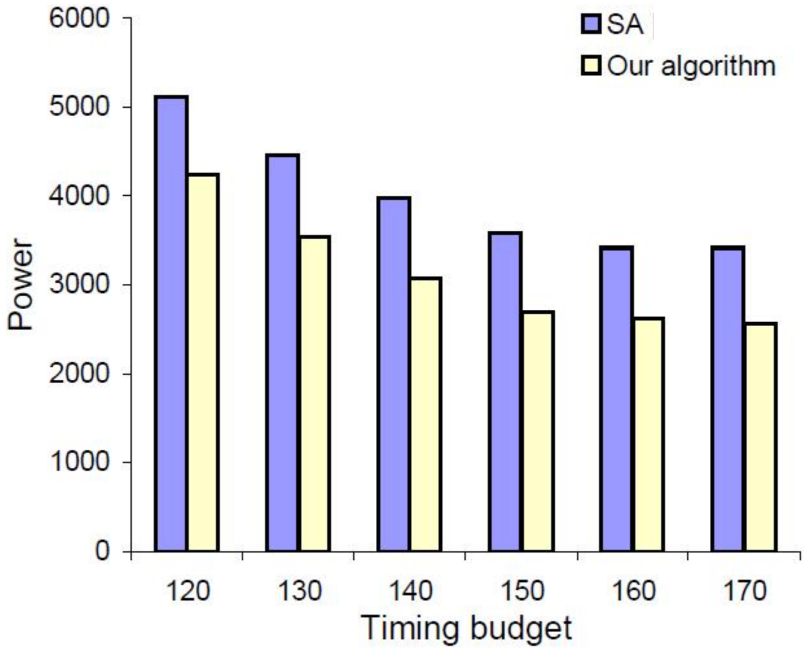


Circuit	SA		Our phase I		Our phase I&II	
	power	runtime	power	runtime	power	runtime
c432	703	1.7	718	1.7	701	2.5
c499	1669	4.9	1637	4.2	1590	6.3
c880	1817	5.1	1172	2.6	1050	4.6
c1355	1385	3.3	1390	4.6	1076	8.2
c1908	2502	10.7	2408	5.3	2296	10.9
c2670	3412	18.6	3167	7.5	2509	15.2
c3540	4645	22.3	4236	10.1	3830	21.8
c5315	8406	26.8	6734	15.2	5023	32.0
c6288	13685	19.2	13055	12.8	12356	25.6
c7552	9510	46.1	7945	26.9	5949	55.0
Average	4773	15.87	4246	9.09	3638	18.21
Norm.	1.0	1.0	0.89	0.57	0.76	1.15

-13%

Experimental Result

- Power minimization under different timing constraints



Conclusion

- We proposed a systematic yet fast algorithm for simultaneous gate sizing and Vt assignment
- The reconvergence paths are solved by consistency relaxation and iterative restriction
- More than 20% improvement over previous work
- This method can be applied to many other combinatorial optimizations on DAG