LAGRANGIAN RELAXATION FOR GATE IMPLEMENTATION SELECTION

Yi-Le Huang, Jiang Hu and Weiping Shi

Department of Electrical and Computer Engineering

Texas A&M University

TEXAS A&M ENGINEERING

OUTLINE

- Introduction and motivation
- Projection-based descent method for solving Lagrangian dual problem
- Distribution of Lagrangian multipliers
- Experimental results
- Conclusion

GATE IMPLEMENTATION SELECTION

- Gate implementation options
  - size
  - P/N ratio
  - threshold voltage (Vt)
  - ...
- Large problem size
  - commonly, hundreds of thousands of gates
  - sometimes millions of gates
- Essential for circuit power and performance

PREVIOUS WORK (CONTINUOUS)

- Solved by
  - Linear programming
  - Convex programming
  - Network flow
- Round fractional solutions to integers
- Fast
- Rounding errors
- Restrictions on delay/power models
- Difficult to handle P/N ratio unless transistor level

PREVIOUS WORK (DISCRETE)

- No rounding error
- Compatible with different power/delay models
- Sensitivity based heuristics
  - Simple
  - Quick
  - Greedy
- Dynamic programming-like search
  - Relatively systematic solution search

LAGRANGIAN RELAXATION (LR)

- Handle conflicting objectives or complex constraints
- With continuous optimization
  - Faster convergence (Chen, Chu and Wong, TCAD 1999)
- With dynamic programming-like search
  - Alleviate the curse of dimensionality

OVERVIEW OF LR

Original problem

Minimize $A(x)$

Subject to: $B(x) \leq 0$

$C(x) \leq 0$

LR subproblem

Lagrangian multiplier

Minimize $A(x) + \lambda \cdot B(x)$

Subject to: $C(x) \leq 0$

Lagrangian dual problem

Find $\lambda \rightarrow$ max optimal solution of subproblem

LR FOR GATE IMPLEMENTATION SELECTION

Original problem

$$\begin{aligned} \text{Min} \quad & P(\vec{x}) \\ \text{s.t.} \quad & a_j \le A \qquad j \in \text{PO} \\ & a_i + D_{ij} \le a_j \qquad i \in \text{in}(j) \\ & D_i \le a_i \qquad i \in \text{PI} \end{aligned}$$

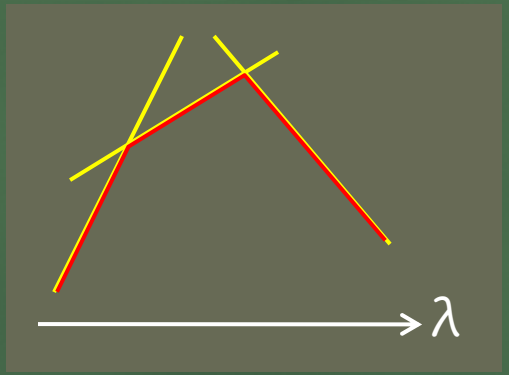
x: implementation decision
P: power
D: delay
a: arrival time

LR subproblem

$$\begin{aligned} \text{Min} \quad & P(\vec{x}) + \Sigma \lambda_j (a_j - A) \\ & + \Sigma \lambda_{ij} (a_i + D_{ij} - a_j) \\ & + \Sigma \lambda_{i0} (D_i - a_i) \end{aligned}$$

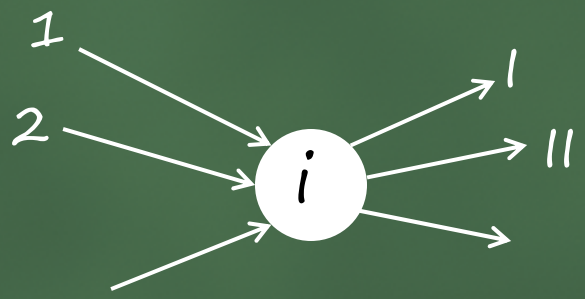
Subgradient == -slack

LAGRANGIAN DUAL PROBLEM



- Ideally
  - a piece-wise convex function
  - solved by subgradient method
    - variant of steepest descent
- In practice
  - no guarantee for optimal subproblem solution
  - dual problem is no longer convex
- How to solve non-convex dual problem?
  - not well studied

KARUSH-KUHN-TUCKER (KKT) CONDITIONS

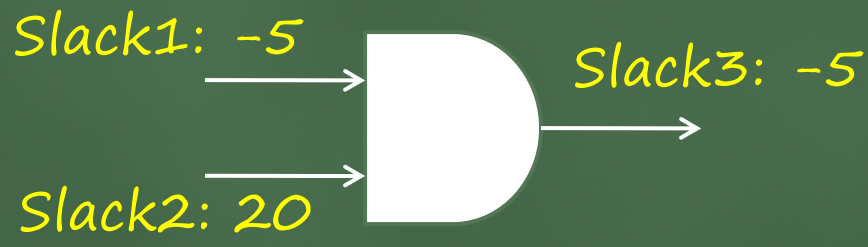


$$\lambda_{1i} + \lambda_{2i} + \ldots = \lambda_{iI} + \lambda_{iII} + \ldots$$

Flow conservation

(Chen, Chu and Wong TCAD99)

PROBLEM OF SUBGRADIENT METHOD

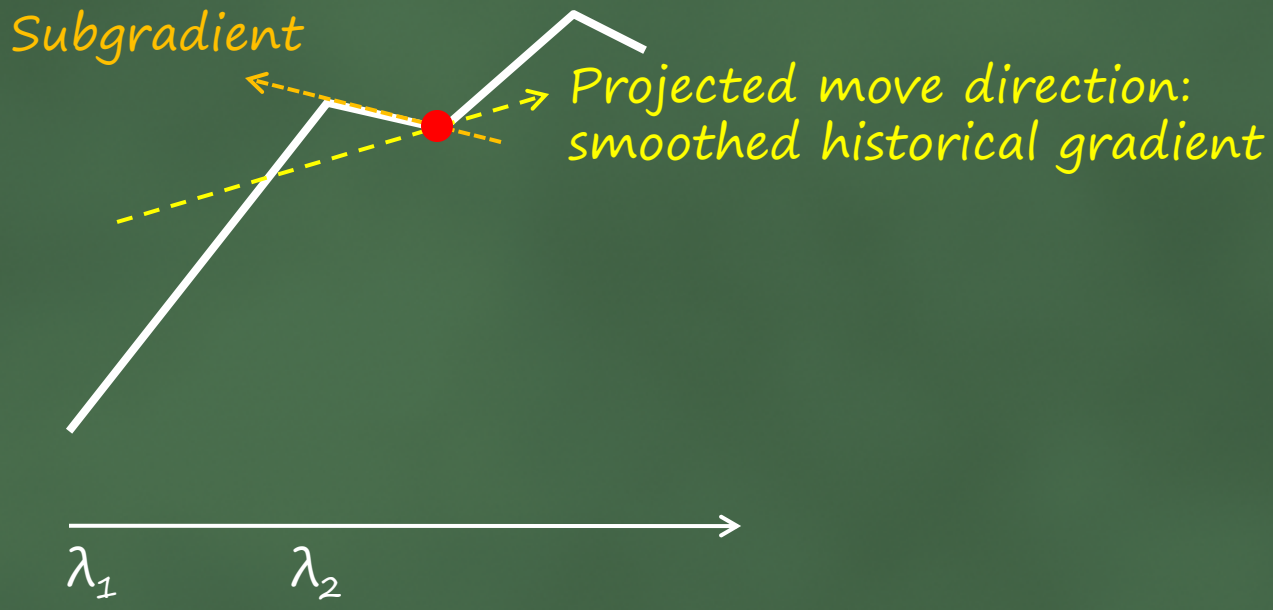


$\Delta\lambda_1 = 5\rho$, $\Delta\lambda_2 = -20\rho$, $\Delta\lambda_3 = 5\rho$

ρ: step size in subgradient method

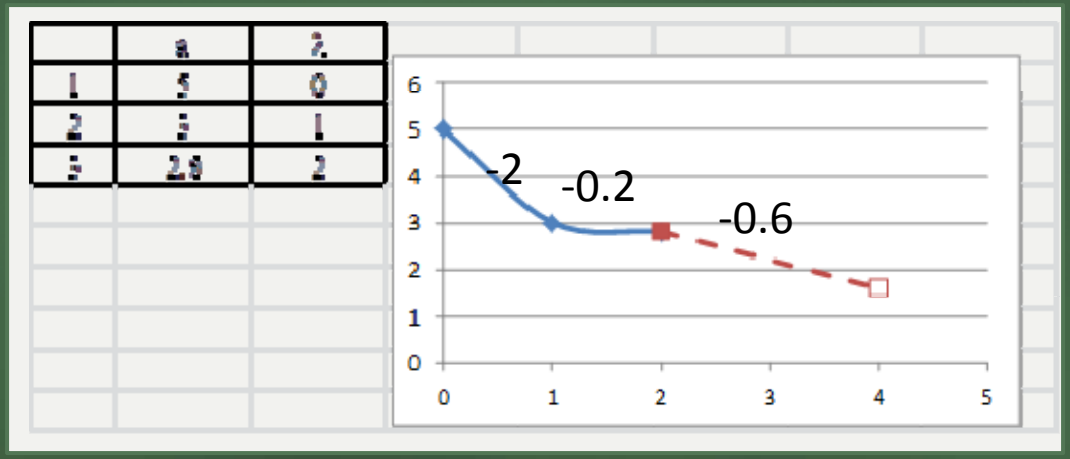
$\Delta\lambda_1 + \Delta\lambda_2 \neq \Delta\lambda_3$

PROJECTION-BASED DESCENT METHOD



PROJECTION ESTIMATION

- Table of (a, λ) in previous iterations
- Gradient history: $(a_i - a_{i-1})/(\lambda_i - \lambda_{i-1})$
- Projection direction
  - Weighted average of historical gradients
  - More weight for recent gradients



STEP SIZE

- $\Delta\lambda = (q - a_{cur}) / \eta$
  - q: required arrival time
  - a_{cur}: current arrival time
  - η: projected move direction

MULTIPLIER UPDATE FLOW

- Multipliers at PO are updated by projection
- They are distributed to entire circuit in reverse-topological order
  - like network flow
- Alternatively, from PI distribute in topological order

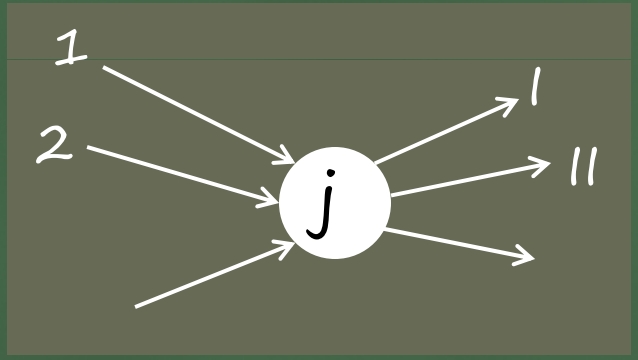
MULTIPLIER DISTRIBUTION

- Ensure flow conservation
- Try to equalize slack: different slacks imply room for power saving

- Given outgoing flow, find x

$$\sum_{i \in in(j)} \frac{x - a_i}{\eta_{ij}} = \sum_{k \in out(j)} \Delta\lambda_{jk}$$

x is the target arrival time



- $\Delta\lambda_{ij} = (x - a_i)/\eta_{ij}$

EXPERIMENT SETUP

- ISCAS85 benchmark
- Cell library based on 70nm technology
- Synthesized by SIS
- Placed by mPL
- Elmore delay
- Analytical power model
- LR subproblem is solved by greedy heuristic
- Compare our approach (projection+greedy) with baseline (subgradient+greedy)

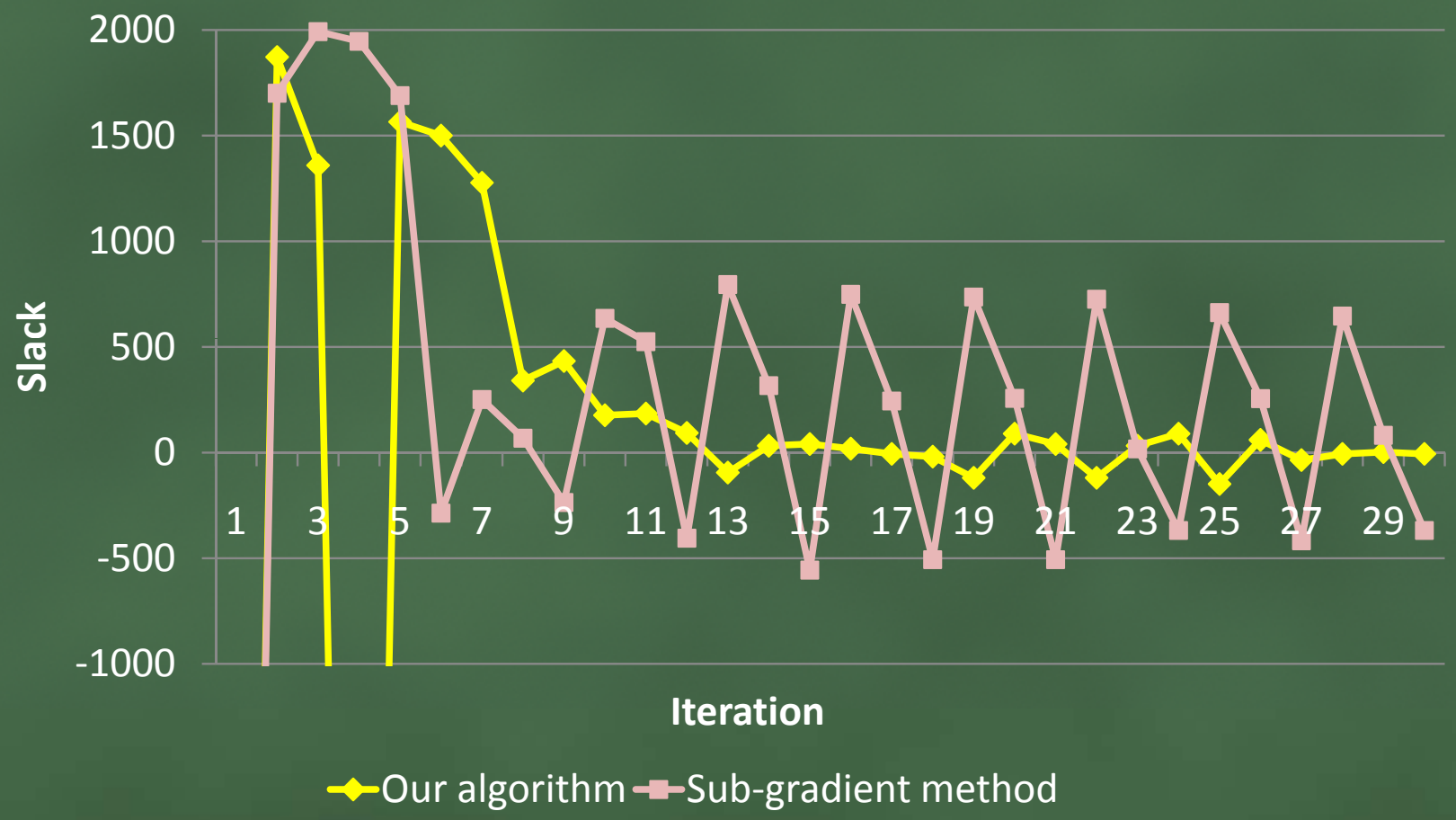
RESULTS WITH TIGHT TIMING CONSTRAINTS

		Initial		Sub-gradient method			Our method		
testcase	# of gates	power	slack	power	slack	run time	power	slack	run time
chain	11	9.3	-295.6	60.8	-13.9	0.0	104.4	0.1	0.4
c432	289	221.7	-10379.8	832.4	-33.1	0.8	803.7	0.6	1.2
c499	539	418.8	-5389.7	1545.4	-11.5	1.5	1522.8	1.7	2.2
c880	340	259.4	-4239.1	515.5	-31.7	0.9	549.4	15.6	1.7
c1355	579	426.6	-5353.7	1470.0	-5.3	1.7	1403.9	7.6	2.5
c1908	722	582.8	-7286.4	1452.7	-12.8	2.2	1402.7	5.9	3.2
c2670	1082	725.1	-16177.1	1465.9	-32.8	2.8	1312.6	9.1	4.1
c3540	1208	994.5	-7369.0	2650.5	-116.6	3.7	3016.6	20.0	5.5
c5315	2440	1941.7	-9956.3	3627.4	-199.0	7.4	4088.7	7.8	10.9
c6288	2342	1819.7	-10476.1	6305.5	-29.4	7.6	5382.4	3.4	11.4
c7552	3115	2390.0	-21197.9	6875.7	-97.2	9.7	5433.9	20.6	14.8
Sum		9790				38.38			57.82
# of violation			11		11			0	

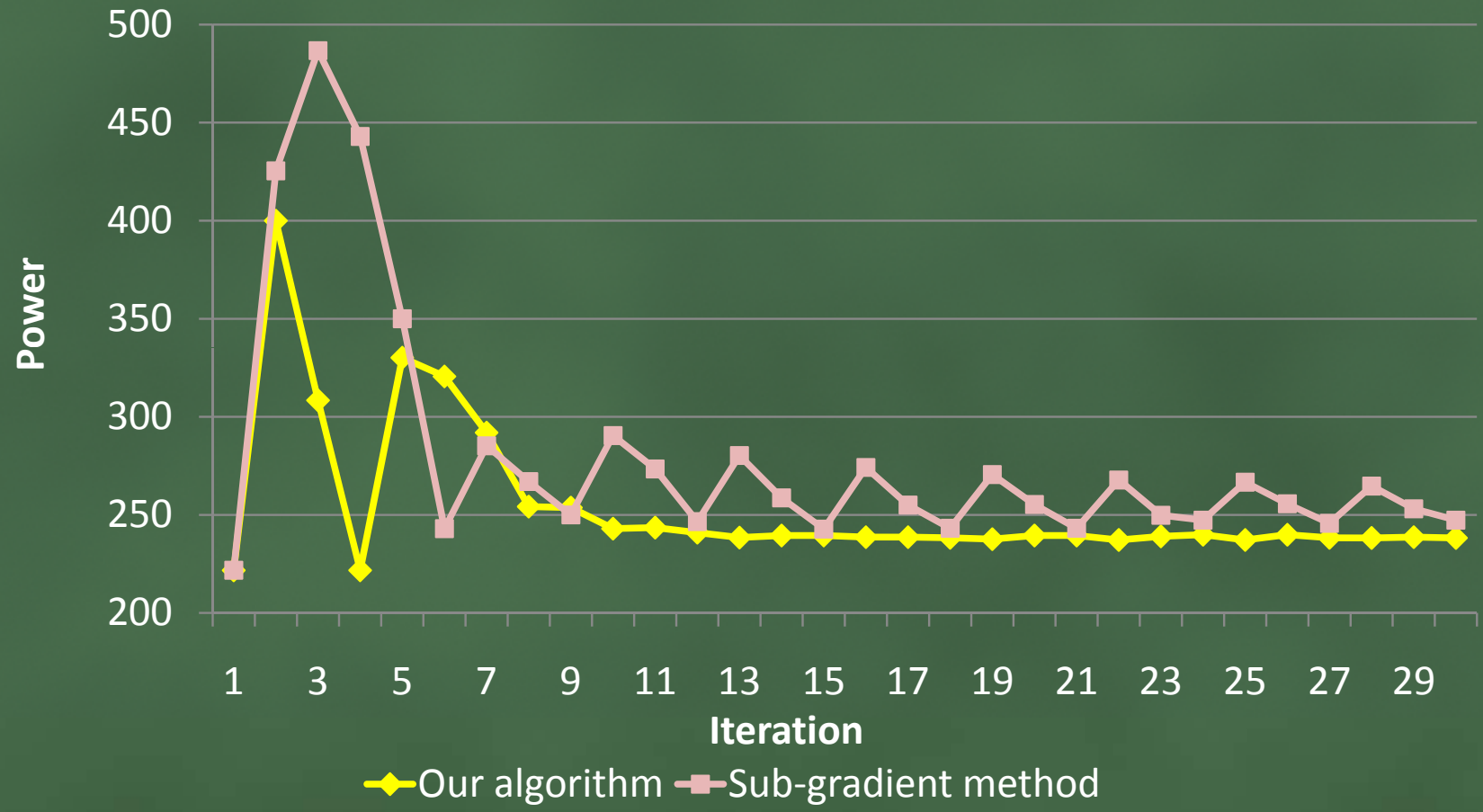
RESULTS WITH LOOSE TIMING CONSTRAINTS

		Initial		Sub-gradient method			Our method		
testcase	# of gates	power	slack	power	slack	run time	power	slack	run time
chain	11	9.3	-215.5	27.2	5.0	0.0	27.2	5.0	0.1
c432	289	221.7	-8033.3	249.8	17.0	0.8	238.7	18.0	1.1
c499	539	418.8	-4198.2	874.5	614.0	1.5	498.8	7.0	2.2
c880	340	259.4	-3219.2	327.8	231.0	0.9	279.8	1.0	1.3
c1355	579	426.6	-4084.3	736.4	38.0	1.7	522.1	3.0	2.5
c1908	722	582.8	-5716.4	878.0	22.0	2.2	666.7	70.0	3.1
c2670	1082	725.1	-12969.7	760.0	711.0	2.8	734.2	115.0	4.0
c3540	1208	994.5	-5873.4	2012.5	718.0	3.7	1147.6	7.0	5.5
c5315	2440	1941.7	-8156.4	3165.6	1033.0	7.4	2171.9	17.0	10.8
c6288	2342	1819.7	-7786.7	3951.3	310.0	7.5	2518.8	13.0	11.5
c7552	3115	2390.0	-16899.7	3897.8	1386.0	9.7	2445.5	107.0	14.3
Sum		9790		16881		38.25	11251		56.34
# of violation			11		0			0	

SLACK OVER ITERATIONS (C432)



POWER OVER ITERATIONS (C432)



CONCLUSIONS AND FUTURE RESEARCH

- Drawbacks of subgradient method are investigated
- New techniques are proposed to solve Lagrangian dual problem for gate implementation selection
- They lead to better solutions and faster convergence
- In future, we will integrate them with dynamic programming-like search


Thank You!