More Realistic Power Grid Verification Based on Hierarchical Current and Power constraints

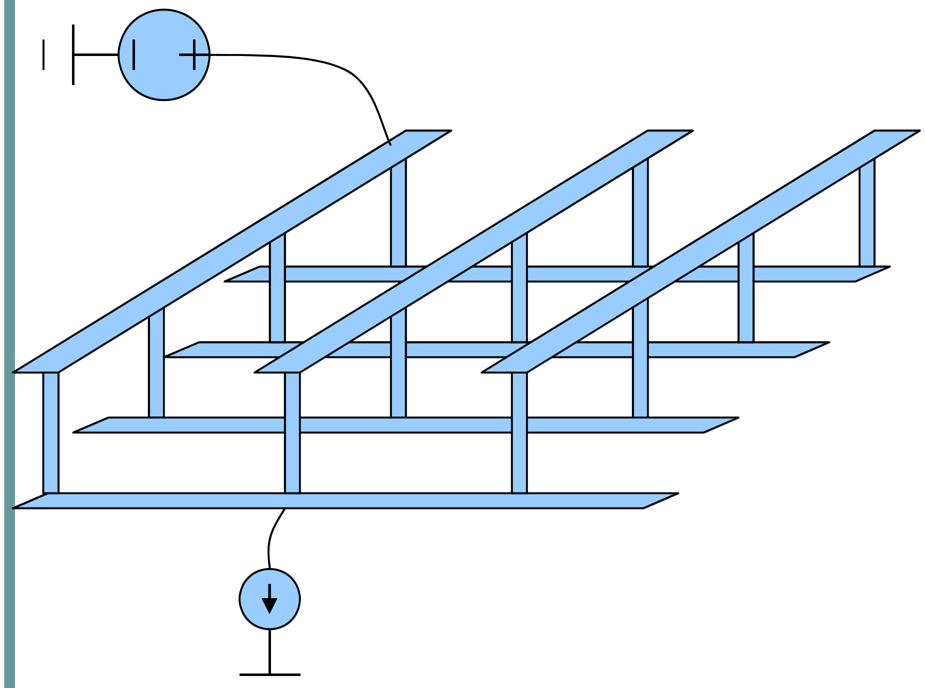
[2]Chung-Kuan Cheng, [2]Peng Du, [2]Andrew B. Kahng,
[1]Grantham K. H. Pang, [1]§Yuanzhe Wang, [1]Ngai Wong

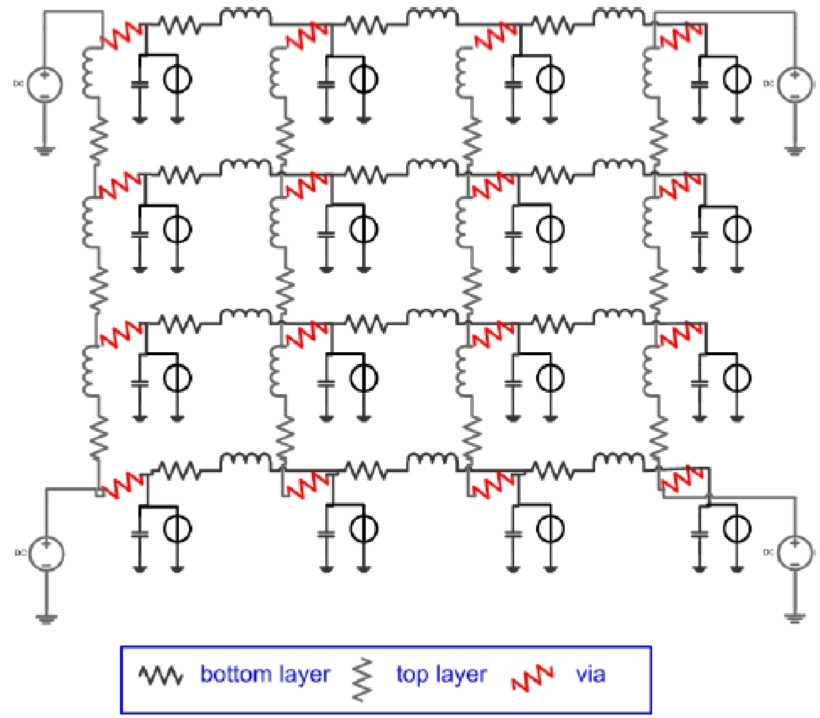
1. The University of Hong Kong
2. University of California, San Diego

Outline

- **Background**
- Problem formulation
- Efficient solver
- Experimental results
- Conclusion

Background





Power grid -> RCL network

External voltage sources -> ideal voltage sources

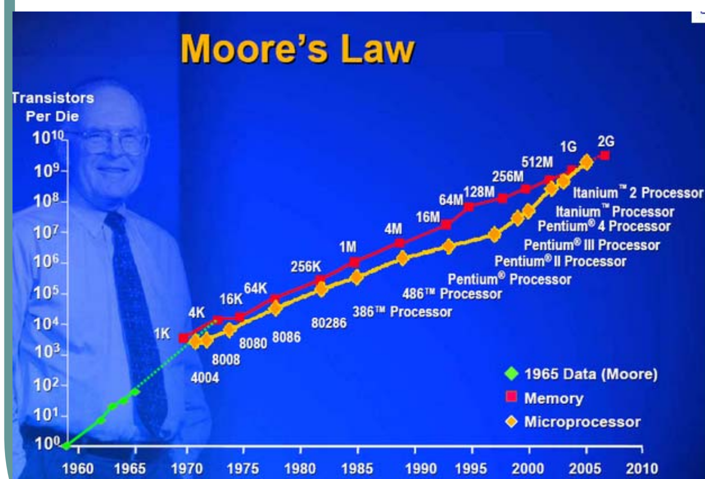
Transistors, logic gates, etc -> ideal current sources

Background

External voltage source → power grid → transistors, logic gates, etc

Voltage+Z_s Current

Voltage drop → Logic error, Timing error



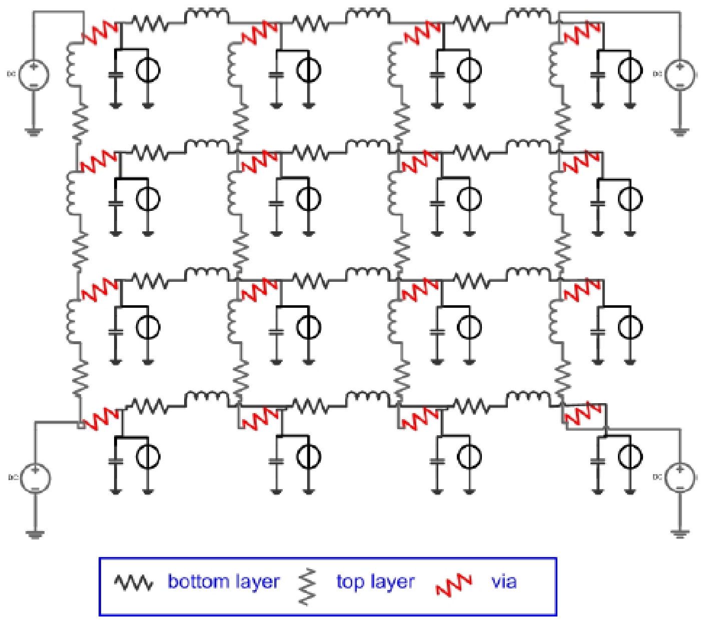
Gates ↑, Current density ↑, Line width ↓ → Voltage drop ↑

External voltage ↓ → Noise margin ↓

Power grid voltage drop verification is becoming indispensable

Background

1. Simulation-based power grid verification



capacitance inductance

admittance impedance

input distribution matrix

$$C\dot{x}(t) + Gx(t) = Bu(t)$$

nodal voltages

current patterns

current patterns $\xrightarrow{\text{transient analysis}}$ voltage drops

Background

2. Worst case power grid verification

max *Voltage_Drop*

subject to: *Current_Constraints*

Design experience

Design requirements

1. Early-stage verification – current patterns unknown
2. Uncertain working modes – too many possible current patterns

Check : max{*Voltage_Drop*} <*Noise_Margin*

Background

$$\max \quad x_k = c^T i$$

$$\text{subject to} \quad \begin{cases} Ui < I_G \\ 0 < i < I_L \end{cases}$$

x_k : nodal voltage at k

i : current sources

I_L : local current bounds

I_G : global current bounds

c : relationship between voltage and current

U : current distribution matrix

Worst-case voltage drop prediction via solving linear programming problems

D. Kouroussis and F. N. Najm, *A static pattern-independent technique for power grid voltage integrity verification*, 2003

Background

Solving linear programs:

1. Simplex algorithm: theoretically NP-hard; $O(n^3)$ in practice.
2. Ellipsoid algorithm: $O(n^4)$
3. Interior-point algorithm: $O(n^{3.5})$

n is usually large (> millions)

Existing work (for higher efficiency):

Geometric method -> trade-off with accuracy (Ferzli, ICCAD '07)

Dual algorithm -> still large complexity (convex optimization) (Xiong, DAC '07)

Outline

- Background
- **Problem formulation**
- Efficient solver
- Experimental results
- Conclusion

Problem formulation

Relationship between voltage drop and currents

$$C\dot{x}(t) + Gx(t) = Bu(t)$$

Backward Euler

$$(G + \frac{C}{\Delta t})x(t+\Delta t) = \frac{C}{\Delta t}x(t) + Bu(t+\Delta t)$$

Numerically equivalent to transient analysis

$$x(k_t \Delta t) = \sum_{k=1}^{k_t} \mathcal{M}^{k_t - k} \mathcal{N} u(k \Delta t)$$

$$\mathcal{M} = \left(G + \frac{C}{\Delta t}\right)^{-1} \frac{C}{\Delta t}, \quad \mathcal{N} = \left(G + \frac{C}{\Delta t}\right) B$$

Problem formulation

Hierarchical current and power constraints

1: Local current constraints

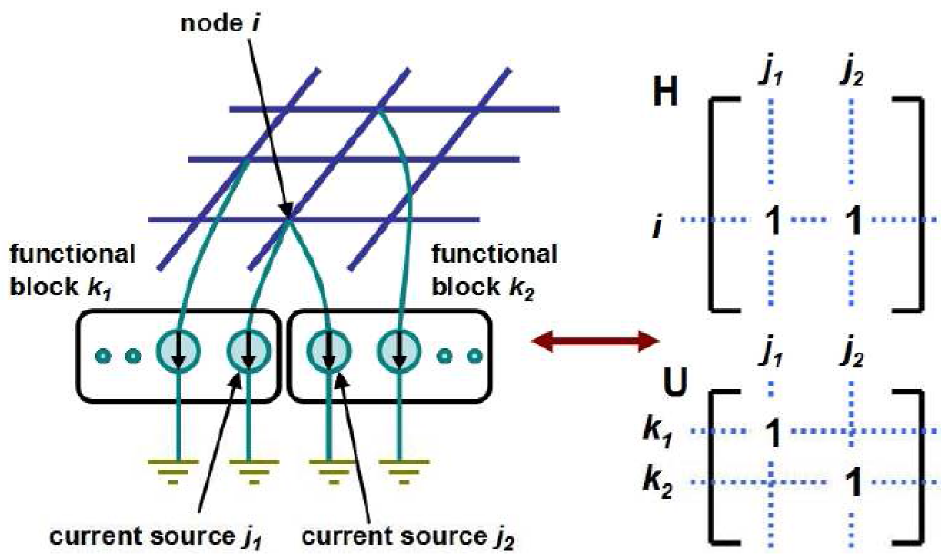
$$0 \le u(t) \le I_L \quad \text{or} \quad 0 \le u(k\Delta t) \le I_L$$

2: Block-level current constraints

$$Uu(t) \le I_G \quad \text{or} \quad Uu(k\Delta t) \le I_G$$

Different from previous work, U is a “0/1” matrix with each column containing at most one “1”.

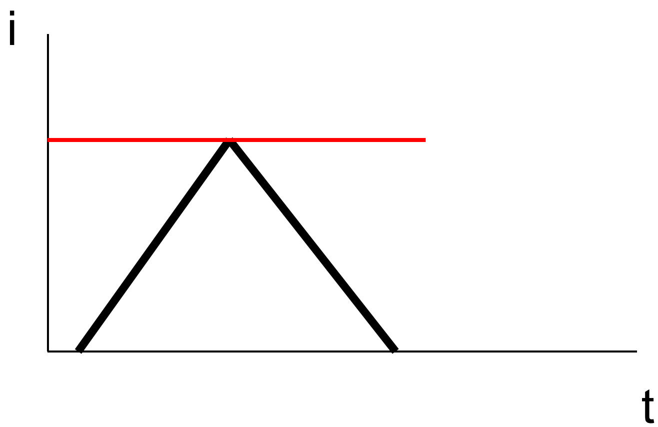
This is the requirement of hierarchy



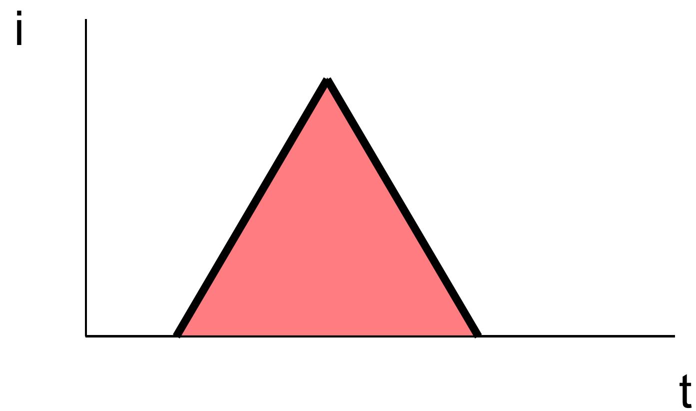
Background

3: Block-level power constraints

$$U\left(\sum_{k=1}^{k_t} u(k\Delta t)\right) \leq \frac{k_t}{V_{dd}} P_B$$



current constraints => peak value of current waveform



power constraints => area under current waveform

Problem formulation

4: High level power constraints

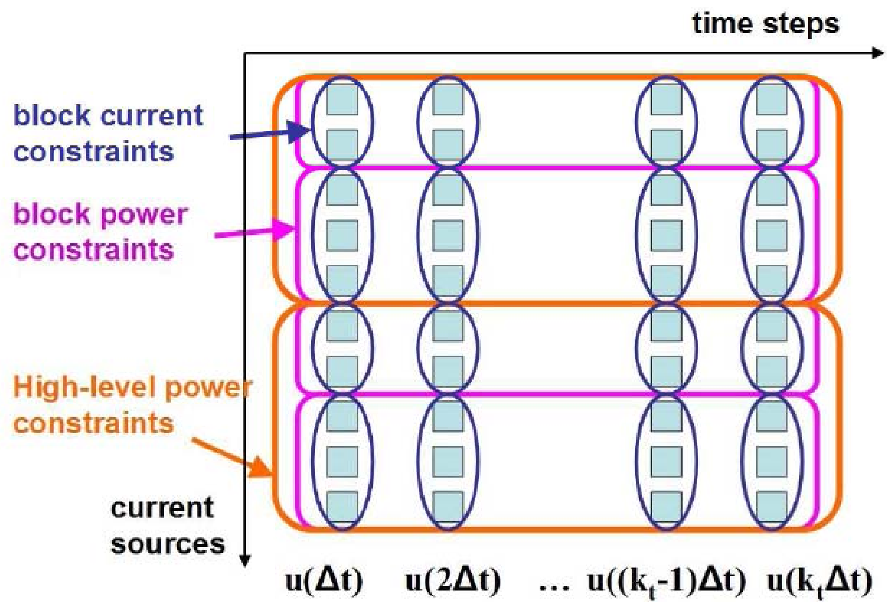
$$[1^{st}\ \text{level}]:\ U_1 U \left(\sum_{k=1}^{k_t} u(k\Delta t) \right) \leq \frac{k_t}{V_{dd}} P_{T1};$$

$$\cdots$$

$$[r^{th}\ \text{level}]:\ U_r U_{r-1} \cdots U_1 U \left(\sum_{k=1}^{k_t} u(k\Delta t) \right) \leq \frac{k_t}{V_{dd}} P_{Tr}.$$

$U_1, U_2, \ldots, U_r$ are 0/1 matrices with each column containing at most one “1”

Hierarchical Constraints



Problem formulation

Worst-case voltage drop occurs at the final time step (see the paper for detailed proof).

Thus the linear programming problem reads:

$$\max_{i\in\Omega} x_i(k_t\Delta t) = \sum_{k=1}^{k_t} c_{i,k} u(k\Delta t)$$

$$\text{s.t.} \begin{cases} 0 \le u(k\Delta t) \le I_L, \quad Uu(k\Delta t) \le I_G, \\ U\left(\sum_{k=1}^{k_t} u(k\Delta t)\right) \le \frac{k_t}{V_{dd}} P_B, \\ U_{r'}U_{r'-1}\cdots U\left(\sum_{k=1}^{k_t} u(k\Delta t)\right) \le \frac{k_t}{V_{dd}} P_T \ (r'=1...r) \end{cases}$$

$c_{i,k}$ is the i^{th} row of $\mathcal{M}^{k_t-k}\mathcal{N}$.

Outline

- Background
- Problem formulation
- **Efficient solver**
- Experimental results
- Conclusion

Efficient solver

Coefficient computation

$$\max_{i \in \Omega} x_i(k_t \Delta t) = \sum_{k=1}^{k_t} c_{i,k} u(k\Delta t)$$

$$c_{i,k} = e_i^T \left[\left(G + \frac{C}{\Delta t} \right)^{-1} \frac{C}{\Delta t} \right]^{k_t - k} \left(G + \frac{C}{\Delta t} \right)^{-1} H$$

We do not have to solve $c_{i,k}$ for every i
(those i to be solved form a set Ω)

- Solving those nodes with current sources attached (# current sources usually < # of nodes)
- Solving those "critical nodes" which have great influence on circuit performance

Efficient solver

$$c_{i,k} = e_i^T \left[\left(G + \frac{C}{\Delta t} \right)^{-1} \frac{C}{\Delta t} \right]^{k_t - k} \left(G + \frac{C}{\Delta t} \right)^{-1} H$$

A *parallel* algorithm without matrix inversion is desired.

$$\xrightarrow{\text{transpose}} c_{i,k}^T = H^T \left(G^T + \frac{C^T}{\Delta t} \right)^{-1} \left[\frac{C^T}{\Delta t} \left(G^T + \frac{C^T}{\Delta t} \right)^{-1} \right]^{k_t - k} e_i$$

$$\xrightarrow[\text{(sparse-LU)}]{G^T + \frac{C^T}{\Delta t} = L_d U_d} c_{i,k}^T = H^T U_d^{-1} L_d^{-1} \underbrace{\left(\frac{C^T}{\Delta t} \right) U_d^{-1} L_d^{-1} \cdots \left(\frac{C^T}{\Delta t} \right) U_d^{-1} L_d^{-1}}_{k_t - k \text{ times}} e_i$$

1. Requiring one sparse-LU and k_t forward/backward substitutions
2. Parallelizable

Efficient solver

$$\max_{i\in\Omega} x_i(k_t\Delta t) = \sum_{k=1}^{k_t} c_{i,k} u(k\Delta t)$$

$c_{i,k}$ known now

Rename variables by treating each entry of each u(kΔt) as independent variables

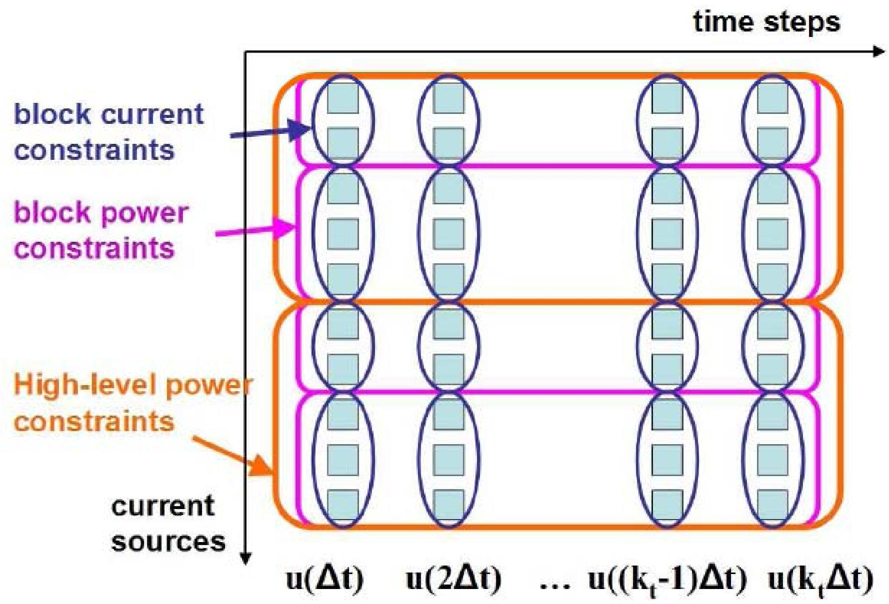
$$\left.\begin{array}{ccc} \tilde{c}_1 = e_1^T c_{i,1}; & \cdots & \tilde{c}_m = e_m^T c_{i,1}; \\ \vdots & \ddots & \vdots \\ \tilde{c}_{(k_t-1)m+1} = e_1^T c_{i,k_t}; & \cdots & \tilde{c}_{k_t m} = e_m^T c_{i,k_t}; \end{array}\right| \begin{array}{ccc} \tilde{u}_1 = u_1(\Delta t); & \cdots & \tilde{u}_m = u_m(\Delta t); \\ \vdots & \ddots & \vdots \\ \tilde{u}_{(k_t-1)m+1} = u_1(k_t\Delta t); & \cdots & \tilde{u}_{k_t m} = u_m(k_t\Delta t). \end{array}$$

The objective function can be rewritten as

$$\max \ x_{i_{node}} = \left(\sum_{i=1}^{mk_t} \tilde{c}_i \tilde{u}_i\right)$$

Efficient solver

Each constraint represents that the sum of some variables belonging to a set is smaller than a bound



$$\sum_{i \in \mathcal{L}_\kappa} \tilde{u}_i \leq \ell_\kappa (\kappa = 1, \ldots, \kappa_t)$$

- κ_t: the total number of constraints. Here $\kappa_t = k_t m + k_t p + p + p_1 + \cdots + p_q$;
- $\mathcal{L}_\kappa$: the set of indices of variables involved in the κ^{th} constraint;
- ℓ_κ: the bound of the κ^{th} constraint;

Efficient solver

The problem is rewritten as

$$\max \; x_{i_{node}} = \left(\sum_{i=1}^{mk_t} \tilde{c}_i \tilde{u}_i \right) \quad \text{s.t.} \quad \sum_{i \in \mathcal{L}_\kappa} \tilde{u}_i \leq \ell_\kappa (\kappa = 1, \ldots, \kappa_t)$$

The constraints here are ***hierarchical***, which follows that for any two sets $\mathcal{L}_{\kappa_1}$ $\mathcal{L}_{\kappa_2}$, at least one of the 3 equations holds:

$$\text{(i) } \mathcal{L}_{\kappa_1} \bigcap \mathcal{L}_{\kappa_2} = \emptyset; \text{ (ii) } \mathcal{L}_{\kappa_1} \subset \mathcal{L}_{\kappa_2}; \text{ (iii) } \mathcal{L}_{\kappa_1} \supset \mathcal{L}_{\kappa_2}.$$

(a) (b)

Efficient solver

$$\max \; x_{i_{node}} = \left(\sum_{i=1}^{mk_t} \tilde{c}_i \tilde{u}_i \right) \quad \text{s.t.} \quad \sum_{i \in \mathcal{L}_\kappa} \tilde{u}_i \leq \ell_\kappa (\kappa = 1, \ldots, \kappa_t)$$

Lemma: The objective function reaches maximum when all the variables $\tilde{u}_i$'s associated with negative $\tilde{c}_i$'s are set to zero.

Intuitive interpretation:

1. The objective function is smaller when variables with negative variables are positive;
2. Set these variables to zero will not decrease the feasible set defined by constraints.

Efficient Solver

Set all the variables associated with negative coefficients as zero and sort the remaining coefficients in the descending order: $\bar{c}_1 \geq \cdots \geq \bar{c}_{\bar{k}} > 0$

The problems becomes

$$\max x_{i_{node}} = \sum_{i=1}^{\bar{k}} \bar{c}_i \bar{u}_i \quad \text{s.t.} \quad \sum_{i \in \mathcal{L}_\kappa} \bar{u}_i \leq \ell_\kappa \;\; (\kappa = 1, \ldots, \bar{\kappa})$$

Then it can be proven that a sorting-deletion algorithm can give the optimal solution.

Efficient solver

$$\max x_{i_{node}} = \sum_{i=1}^{\bar{k}} \bar{c}_i \bar{u}_i \quad \text{s.t.} \quad \sum_{i \in \mathcal{L}_\kappa} \bar{u}_i \le \ell_\kappa \quad (\kappa = 1, \ldots, \bar{\kappa})$$

Algorithm 1 : Sorting-deletion algorithm

for $i = 1,\ldots,\bar{k}$ **do**

- (1) Select all the sets $\mathcal{L}_\kappa$ that satisfy $i \in \mathcal{L}_\kappa$. The subscripts of these $\mathcal{L}_\kappa$ form a set $\mathcal{K}_i$;
- (2) Set $\bar{u}_i$ to be $\min\{\ell_\kappa | \kappa \in \mathcal{K}_i\}$;
- (3) $\ell_\kappa = \ell_\kappa - \bar{u}_i$ for all $\kappa \in \mathcal{K}_i$;

Intuitive interpretation:

Give the variable associated with the largest coefficient the largest possible value. Then delete this variable from the problem and do the same procedure again.

Efficient solver

Complexity of the sorting-deletion algorithm

1. Coefficient sorting: (using the most efficient sorting algorithm)

$$O(mk_t \log mk_t)$$

2. Deletion procedure: (r is the # of level in the hierarchical structure, mk_t is # of variables)

$$O(mk_t r)$$

Much lower than standard algorithms

$$O(mk_t \log mk_t) + O(mk_t r) << O\left((mk_t)^3\right)$$

Outline

- Background
- Problem formulation
- Efficient solver
- **Experimental results**
- Conclusion

Experimental results

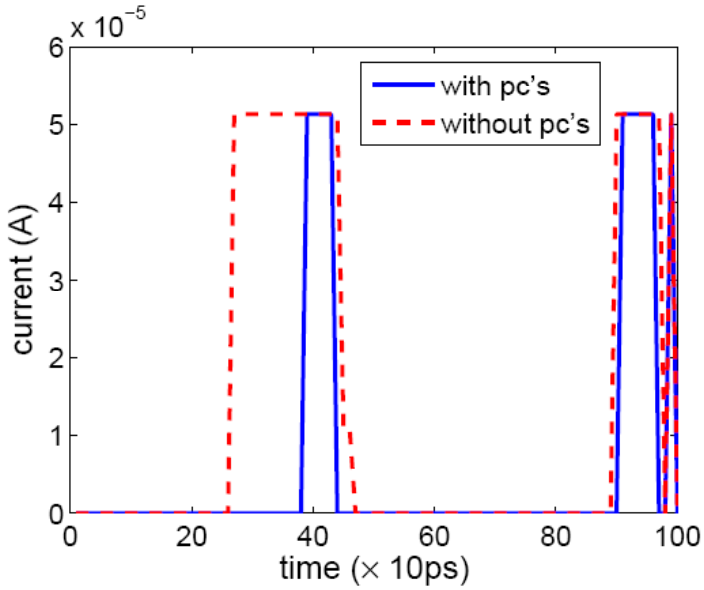
Models used:

3-D power grid structure with 4 metal layers

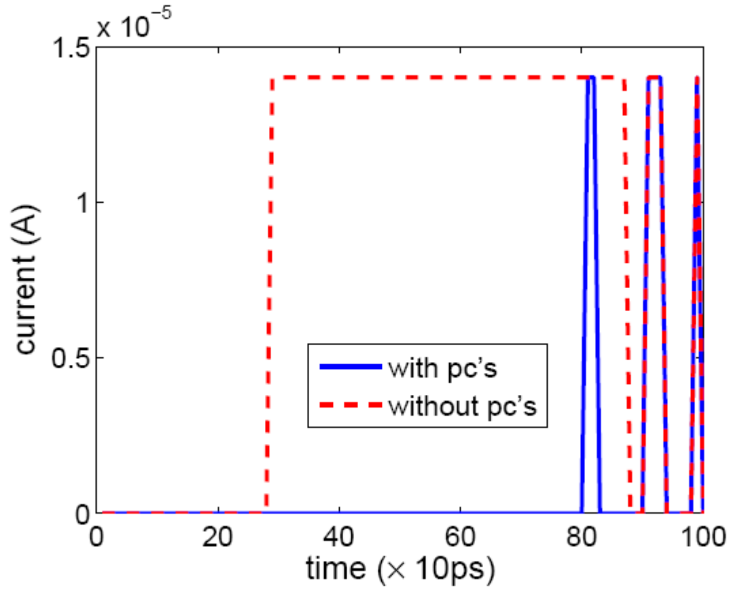
	Power grid models						LP problems	
	Nodes (N)	Sources (m)	Matrix size (n)	No. of R's	No. of C's	No. of L's	Variables	$\|\Omega\|$
Power grid 1	75,762	37,881	113,499	54,350	37,684	37,684	3.7M	100
Power grid 2	980,313	490,157	146,9755	608,792	394,444	394,444	690M	100

1. Compare the voltage drop predictions with and without power constraints
2. Compare the CPU time using sorting-deletion algorithm and standard algorithms

Experimental results



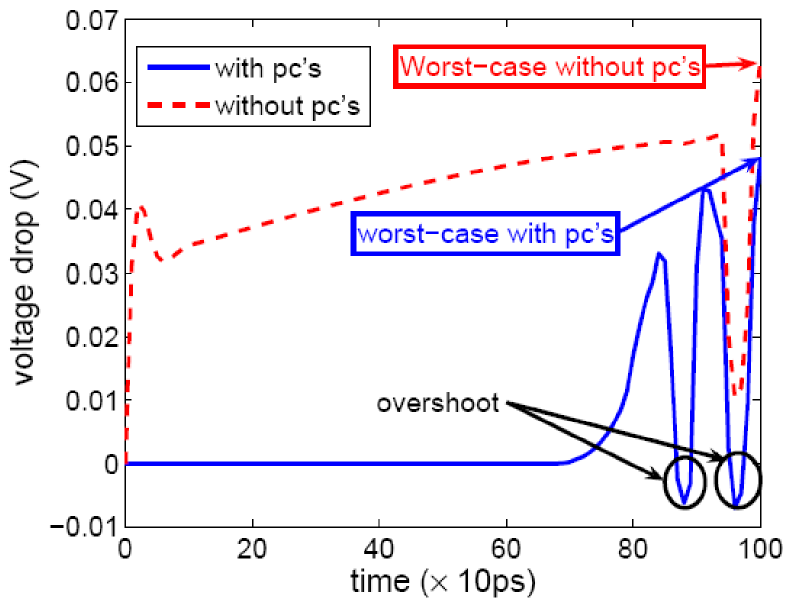
(a) Current source 18,941 of power grid 1



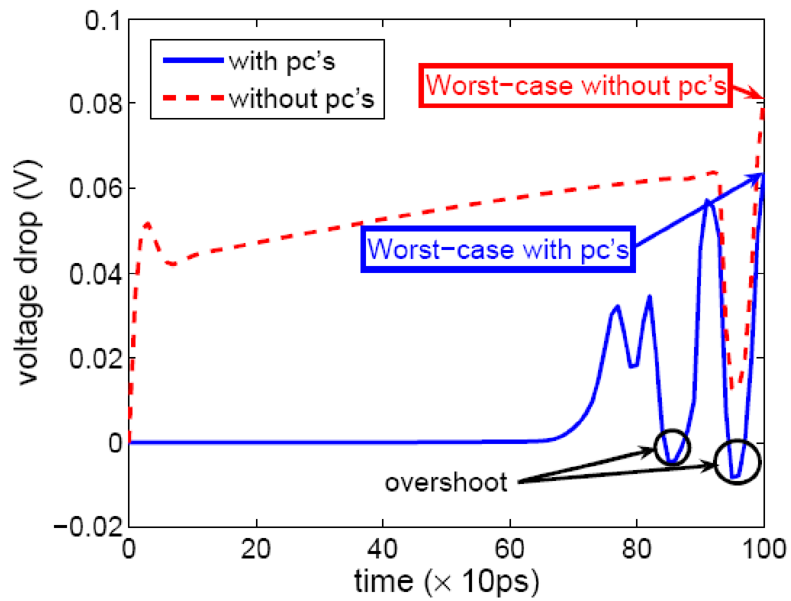
(b) Current source 113,990 of power grid 2

Worst-case current patterns with and without power constraints (pc's). Introduction of power constraints may reduce over-pessimism.

Experimental results



(a) Power grid 1



(b) Power grid 2

Worst-case voltage drop predictions with and without power constraints (pc's). Introduction of power constraints may reduce over-pessimism.

Experimental results

			Without pc's			With pc's		
			Standard method	Proposed algorithm	Speed-up	Standard method	Proposed algorithm	Speed-up
Power grid 1	Single node	Setup	9.86 sec	9.86 sec	–	9.86 sec	9.86 sec	–
		Solving	6.08 sec	0.71 sec	8.56×	–(1)	0.77 sec	*inf*
	$\lvert\Omega\rvert$ nodes	Setup	901 sec	901 sec	–	901 sec	901 sec	–
		Solving	577 sec	70.2 sec	8.22×	–(1)	76.5 sec	*inf*
Power grid 2	Single node	Setup	278 sec	278 sec	–	278 sec	278 sec	–
		Solving	74.4 sec	9.91 sec	7.51×	–(1)	10.87 sec	*inf*
	$\lvert\Omega\rvert$ nodes	Setup	417 min	417 min	–	417 min	417 min	–
		Solving	120 min	15.4 min	7.83×	–(1)	17.1 min	*inf*

(1) Standard LP solver fails due to too many iterations

CPU time comparison between standard algorithms and sorting-deletion algorithm. Significant speed-up is achieved.

Conclusion

- Introduction of power constraints provide more realistic current patterns and less pessimistic voltage drops.
- Efficient and parallelizable coefficient computation is proposed.
- Sorting-deletion algorithm significantly reduces the CPU time to solve the linear programming problems.