Early Power Grid Verification Under Circuit Current Uncertainties*

Imad A. Ferzli
Department of ECE
University of Toronto
Toronto, Ontario, Canada
ferzli@eecg.utoronto.ca

Farid N. Najm
Department of ECE
University of Toronto
Toronto, Ontario, Canada
f.najm@utoronto.ca

Lars Kruse
Magma Design Automation
Eindhoven, The Netherlands
lars@magma-da.com

ABSTRACT

As power grid safety becomes increasingly important in modern integrated circuits, so does the need to start power grid verification early in the design cycle and incorporate circuit uncertainty of the early stages into useful power grid information. This work adopts the framework of capturing circuit uncertainty via constraints on circuit currents, and follows a geometric approach to transform a problem whose solution requires as many linear programs as there are power grid nodes, to another involving a user-limited number of solutions of one linear system.

Categories and Subject Descriptors

B.7.2 [**Design Aids**]: Verification

General Terms

Verification, Algorithms, Design

Keywords

power grid, hyperplane, hypercube, vertex, subset-sum problem

1. INTRODUCTION

Power grid verification must start at design-time. For one thing, power routing resources must be committed in the early stages, often prior to the very completion of circuit design. Uncertainty in circuit design is, therefore, part and parcel of the power grid verification problem, as stated in [1]:"*The crux of the problem in designing a power grid is that there are many unknowns until the very end of the design cycle. Nevertheless, decisions about the structure, size and layout of the power grid have to be made at very early stages when a large part of the chip design has not even begun.*" Although this statement dates back almost ten years, it still is true today: Existing power grid verification tools assume a fully designed circuit and are only useful after placement and routing, or for final signoff. As a matter of practice, many design groups often start with what they view, based on previous experience, as "safe", over-designed grids, but have no way to estimate the extent of over-design, and no way to tell whether some sections of the grid are in reality susceptible to voltage violations, because of the lack of a systematic way to incorporate circuit uncertainties. Therefore, there is a need to *prototype the grid* early in the design flow, i.e., estimate its worst-case voltage drops, while accounting for circuit uncertainty.

A critical aspect of circuit uncertainty for the power grid, that which is the focus of this work, is the imprecise characterization of circuit currents. To capture this uncertainty, we work in the frame of current constraints [2], which specify a feasible space for currents during circuit operation. Grid verification becomes a matter of computing the maximum voltage drops on the grid, for all feasible currents. This is not an easy problem, and previous work [2] tackled it by solving a linear program (LP) for every node on the grid. Full grid verification would therefore require as many LPs as there are nodes, which is not practicable for large grids. The present work follows starkly different methods, exploiting the particularities of the power grid at design time and the geometry of the feasibility space, as described by the current constraints, to derive an efficient solution.

*This research was supported in part by Intel Corp., by Altera Corp., and by the SRC.

ISLPED'07, August 27–29, 2007, Portland, Oregon, USA.

2. PROBLEM FORMULATION

2.1 Constraint-Based Framework

Consider a power grid with n nodes, m of which having a current source tied to them. Assuming, without loss of generality, that nodes with a current source are numbered $1, \ldots, m$, we can write the RC-model for the power grid as:

$$\mathbf{C}\dot{\mathbf{v}}(t) + \mathbf{G}\mathbf{v}(t) = \begin{bmatrix} \mathbf{i}(t) \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{I} \\ \mathbf{0} \end{bmatrix} \mathbf{i}(t) = \mathbf{H}\mathbf{i}(t). \qquad (1)$$

The system equation (1) can be formulated [2] so that n is specifically the number of non-C4 nodes, i.e., nodes that do not correspond to V_{dd} sites, and $\mathbf{v}(t)$ is the n-vector of time varying voltage drops (difference between V_{dd} and node voltages). $\mathbf{i}(t)$ is the m-vector of current loads, and $\mathbf{H}$ is an $(n \times m)$ matrix whose top $(m \times m)$ block is the identity matrix and bottom block 0. Assuming all capacitance is node-to-ground, $\mathbf{C}$ is the $(n \times n)$ diagonal capacitance matrix. $\mathbf{G}$ is the $(n \times n)$ conductance matrix and is a symmetric positive definite M-matrix [3]. For the purposes of power grid verification and optimization, DC analysis plays an important role in some practical design flows. The system equation for a DC grid model is analogous to (1), and given by $\mathbf{G}\mathbf{v} = \mathbf{H}\mathbf{i}$. We draw this parallel because the techniques proposed in this paper apply to both the DC and RC models.

We specify current constraints as linear inequalities [2], and we distinguish local and global constraints. A local constraint sets a bound on the maximum value of current drawn by a source. For example, if $i_j(t)$ does not exceed l_j, we write $0 \le i_j(t) \le l_j$. A similar inequality at every current source implies:

$$\mathbf{0} \le \mathbf{i}(t) \le \mathbf{I_L} \qquad (2)$$

where the j^{th} component of $\mathbf{I_L}$ is l_j. Global constraints are upper bounds on the sum of certain current subsets. They are user-supplied, and meant to systematically incorporate engineering

knowledge of circuit uncertainty. We express them as:

$$\sum_{j=0}^{m-1} u_{ij} i_j(t) \leq g_i, \qquad i = 0, \ldots, c-1, \tag{3}$$

where c is the number of global constraints. Let $\mathcal{G}_i$ be the i^{th} global constraint. u_{ij} is an indicator variable taking the value of 1 if i_j is included in $\mathcal{G}_i$, and 0 otherwise. Together, (2) and (3) define a feasibility region for circuit currents, denoted by $\mathcal{I}_\mathcal{F}$. We want to compute the vector of maximum node voltage drops on the grid as currents vary inside $\mathcal{I}_\mathcal{F}$. The exact solution to this problem is prohibitive and involves solving one LP over $\mathbb{R}^n$, for every node on the grid, i.e., n LPs in total. This work proposes a different approach, which, short of finding the vector of exact maximum voltage drops at every node, computes an efficient upper bound thereon, in both accuracy and speed.

2.2 Vector of Upper Bounds

As a matter of notation, we mention that all inequalities in this paper, when applied on vectors and matrices, and all min/max operators applied on vectors, are component-wise.

Let $\mathbf{A} = \mathbf{G} + \mathbf{C}/\Delta t$. Time-discretizing (1) yields:

$$\mathbf{v}(t) = \mathbf{A^{-1}Hi}(t) + \mathbf{A^{-1}}(\mathbf{C}/\Delta t)\mathbf{v}(t - \Delta t). \tag{4}$$

$\mathbf{A}$ can be shown to be an M-matrix [2], so that$\mathbf{A^{-1}} \geq 0$, therefore:

$$\mathbf{v}(t) \leq \max_{\mathbf{i}(t)\in\mathcal{I}_\mathcal{F}} \mathbf{A^{-1}Hi}(t) + \mathbf{A^{-1}}(\mathbf{C}/\Delta t)\mathbf{v_{ub}}(t - \Delta t). \tag{5}$$

$\max \mathbf{A^{-1}Hi}(t)$ is a vector of component-wise maxima: if $\mathbf{a_j}$ is the j^{th} row of $\mathbf{A^{-1}}$, then the j^{th} component of $\max \mathbf{A^{-1}Hi}(t)$ is $\max \mathbf{a_j Hi}(t)$, $\mathbf{i}(t) \in \mathcal{I}_\mathcal{F}$. $\mathbf{v_{ub}}(t - \Delta t)$ is a vector of upper bounds on the maximum voltage drops at $(t - \Delta t)$.

Although the RC model features dynamic currents and voltages, the description of local and global constraints is static, i.e., $\mathbf{I_L}$ in (2) and g_i in (3) do not depend on time and $\mathcal{I}_\mathcal{F}$ is the same for each time step. Therefore, $\max \mathbf{A^{-1}Hi}(t)$, $\mathbf{i}(t) \in \mathcal{I}_\mathcal{F}$, is independent of t, and we will denote it by $\mathbf{V_a}$. We have, from (5):

$$\mathbf{v_{ub}}(t) = \mathbf{V_a} + \mathbf{A^{-1}}(\mathbf{C}/\Delta t)\mathbf{v_{ub}}(t - \Delta t) \tag{6}$$

is an upper bound on $\mathbf{v}(t)$. Denote by $\mathbf{v}(0)$ the vector of voltage drops on the grid at $t = 0$, i.e., the grid's initial condition. This leads to $\mathbf{v_{ub}}(0) = \mathbf{v}(0)$ and $\mathbf{v_{ub}}(\Delta t) = \mathbf{V_a} + \mathbf{A^{-1}}(\mathbf{C}/\Delta t)\mathbf{v}(0)$. Since $\mathbf{A^{-1}} \geq \mathbf{0}$ and $\mathbf{B} = \mathbf{A^{-1}C}/\Delta t \geq \mathbf{0}$, writing (6) at time steps $\Delta t, \ldots, k\Delta t$ yields:

$$\mathbf{v_{ub}}(k\Delta t) \quad = \quad (\mathbf{I} + \mathbf{B} + \ldots + \mathbf{B^{k-1}})\mathbf{V_a} + \mathbf{B^k}\mathbf{v}(0), \tag{7}$$

where $\mathbf{I}$ denotes the identity matrix. The convergence of $\mathbf{v_{ub}}$, as $k \to \infty$, depends on the convergence of a) the matrix series $\sum_{k=0}^{\infty} \mathbf{B^k}$ and b) the matrix sequence $\mathbf{B^k}$, as $k \to \infty$.

The series $\sum_{k=0}^{\infty} \mathbf{B^k}$ is known to converge [3] if and only if $\rho(\mathbf{B}) < 1$, where $\rho(\mathbf{B})$ is the magnitude of the largest eigenvalue of $\mathbf{B}$, under which condition the series limit is $(\mathbf{I} - \mathbf{B})^{-1}$. The condition that $\rho(\mathbf{B}) < 1$ is also necessary and sufficient for the convergence of the sequence $\mathbf{B^k}$ to 0 [3]. We will now prove that $\rho(\mathbf{B}) < 1$, and find a limit value for $\mathbf{v_{ub}}(k\Delta t)$, independent of the initial condition $\mathbf{v}(0)$, by making use of the fact that $\mathbf{B} \geq 0$ and of the following theorem [3]:

Theorem 1. *Let $\mathbf{B}$ be a nonnegative matrix. Then $\rho(\mathbf{B}) < 1$ if and only if $\mathbf{I} - \mathbf{B}$ is nonsingular and $(\mathbf{I} - \mathbf{B})^{-1}$ is nonnegative.*

A key convergence result is captured in the following claim:

Corollary 1. *$\mathbf{v_{ub}}(k\Delta t)$ converges to $\mathbf{V_u} = (\mathbf{I} + \mathbf{G^{-1}C}/\Delta t)\mathbf{V_a}$, as $k \to \infty$, for all $\Delta t > 0$.*

PROOF. $\mathbf{B} = \mathbf{A}^{-1}(\mathbf{A} - \mathbf{G}) = \mathbf{I} - \mathbf{A^{-1}G}$, $\mathbf{I} - \mathbf{B} = \mathbf{A^{-1}G}$, and $(\mathbf{I} - \mathbf{B})^{-1} = \mathbf{G^{-1}A} = \mathbf{I} + \mathbf{G^{-1}C}/\Delta t$. $\mathbf{A}$ and $\mathbf{G}$ being positive definite, $\mathbf{det}(\mathbf{A^{-1}G}) = \mathbf{det}(\mathbf{G})/\mathbf{det}(\mathbf{A}) \neq 0$ so $\mathbf{I} - \mathbf{B}$ is nonsingular. Since $\mathbf{G}$ is an M-matrix, $\mathbf{G^{-1}} \geq 0$. Given that $\mathbf{C} \geq 0$, we immediately have that $(\mathbf{I} - \mathbf{B})^{-1} \geq \mathbf{0}$. Therefore, by theorem 1, $\rho(\mathbf{B}) < 1$. This implies that $\mathbf{B^k}$ converges to 0 and that $\sum_{k=0}^{\infty} \mathbf{B^k}$ converges to $(\mathbf{I}-\mathbf{B})^{-1} = \mathbf{I}+\mathbf{G^{-1}C}/\Delta t$, as $k \to \infty$, yielding that $\mathbf{v_{ub}}(k\Delta t)$ converges to $(\mathbf{I} + \mathbf{G^{-1}C}/\Delta t)\mathbf{V_a}$. □

Computation of this limit is easy, given $\mathbf{V_a}$: scale the j^{th} component of $\mathbf{V_a}$ by $c_j/\Delta t$ (c_j is the j^{th} component of $\mathbf{C}$), yielding $\mathbf{V_s} = \mathbf{CV_a}/\Delta t$, then solve for $\mathbf{V_b}$ such that $\mathbf{GV_b} = \mathbf{V_s}$ (one standard linear system solve). The upper bound is the sum $\mathbf{V_u} = \mathbf{V_a} + \mathbf{V_b}$. Therefore, $\mathbf{V_u}$ can be easily deduced from $\mathbf{V_a}$, given a standard factorization of $\mathbf{G}$.

The problem of finding a vector of upper bounds on the voltage drop maxima given an RC grid model is thus reduced to finding $\mathbf{V_a} = \max \mathbf{A^{-1}Hi}$, $\mathbf{i} \in \mathcal{I}_\mathcal{F}$. The problem is similar when a DC grid model is used: find $\max \mathbf{G^{-1}Hi}$, $\mathbf{i} \in \mathcal{I}_\mathcal{F}$, with the exception that the DC solution yields the exact vector of voltage drop maxima, not an upper bound thereon.

We digress to mention that, when transient analysis is used, $\mathbf{V_a}$ depends on the choice of the time step Δt: since $\mathbf{V_a}$ is the upper bound on the voltage drops at $t = \Delta t$, the smaller the time step, the smaller $\mathbf{V_a}$. In fact, although the upper bound converges for any $\Delta t > 0$, choosing too small a time step leads to overestimating the maximum voltage drops in a way that could be easily avoided. To see why, observe that in the above derivation of $\mathbf{V_u}$, we are implicitly assuming that currents may change arbitrarily in their feasibility region $\mathcal{I}_\mathcal{F}$ within Δt time. This would allow, for example, that currents switch from 0 to their maximum possible values, i.e., their local constraints, or the other way around, within a single time step. This is clearly not the case for arbitrarily small Δt. With this in mind, we note that problems with a larger time step are subsets of problems with a smaller time step, by virtue of the fact that maximizations need to be performed at a greater number of time points when smaller time steps are used. This results in the observation that the smaller Δt, the larger $\mathbf{V_u}$. Therefore, the upper bound resulting from unrealistically small Δt is too pessimistic an estimate. On the other hand, the time step needs to be small enough to capture the transition times on the grid voltages. Therefore, design expertise needs to guide the choice of Δt by striking a balance between the dynamics of the current loads and voltage responses on the grid, in order to avoid pessimism in the computation of $\mathbf{V_u}$. An alternative, which could obliviate this question altogether, is the use of dynamic current constraints, i.e., making $\mathcal{I}_\mathcal{F}$ time-dependent. This may be more difficult in practice, both from the user's standpoint (supplying dynamic constraints) as well as the tool's (dealing with them). However, dynamic constraints may afford greater accuracy in voltage drop estimation. Either way, dealing with dynamic current constraints is part of our ongoing research, and this papers considers only static constraints.

Going back to the estimation of $\mathbf{V_u}$, observe that, for a given Δt, if $\mathbf{V_{a,ub}} \geq \mathbf{V_a}$, then $(\mathbf{I} + \mathbf{G^{-1}C}/\Delta t)\mathbf{V_{a,ub}} \geq \mathbf{V_u}$, i.e., a conservative estimate of $\mathbf{V_a}$ leads to a conservative estimate of $\mathbf{V_u}$. In what follows, we propose an approach to efficiently compute a conservative estimate of $\mathbf{V_a} = \max \mathbf{A^{-1}Hi}$, where $\mathbf{A}$ is understood to mean $\mathbf{G}$ in case of DC analysis.

3. COMPUTATIONAL STRATEGY

3.1 Finding $\mathbf{V_a}$ at The Vertices of $\mathcal{I}_\mathcal{F}$

Every local and global current constraint corresponds to a hyperplane, more precisely, to a half-space of a hyperplane, in $\mathbb{R}^m$. The feasibility region $\mathcal{I}_\mathcal{F}$ is thus a convex polytope [4] formed by the intersection of all these half-spaces. A global constraint $\mathcal{G}_i$, of the form $\sum u_{ij} i_j \leq g_i$, as in (3), defines a hyperplane $\mathcal{H}_{\mathcal{G}_i} : \sum u_{ij} i_j = g_i$, and two half-spaces $\mathcal{H}^-_{\mathcal{G}_i} : \sum u_{ij} i_j \leq g_i$ (belongs to $\mathcal{G}_i$), and $\mathcal{H}^+_{\mathcal{G}_i} : \sum u_{ij} i_j > g_i$ (outside of $\mathcal{G}_i$). Note that, from (2), two hyperplanes are associated with every local constraint: one for the lower bound of 0, and one for the upper bound. If we have c global constraints, then $\mathcal{I}_\mathcal{F}$ consists of the intersection of $(2m+c)$ half-spaces. Of interest are the vertices of $\mathcal{I}_\mathcal{F}$. A detailed discussion of hyperplanes, polytopes, and vertices is beyond the scope of this paper, and we refer the reader to [4] for a comprehensive discussion. It suffices to say that a vertex of $\mathcal{I}_\mathcal{F}$ is formed by the intersection of any m of the $(2m + c)$ hyperplanes, provided this intersection exists and belongs to $\mathcal{I}_\mathcal{F}$.

Notice that the local constraints alone form a hypercube in $\mathbb{R}^m$, with 2^m vertices at $[0/l_0, 0/l_1, \ldots, 0/l_{m-1}]^T$. The j^{th} local

constraint, therefore, can be either "turned ON" in a vertex if its corresponding entry is l_j, or "turned OFF" if 0. Denoting the local constraint hypercube by $\mathcal{K}$, we can write $\mathcal{I}_\mathcal{F}$ as

$$\mathcal{I}_\mathcal{F} = \mathcal{K} \cap \mathcal{H}^-_{\mathcal{G}_0} \cap \mathcal{H}^-_{\mathcal{G}_1} \ldots \cap \mathcal{H}^-_{\mathcal{G}_{c-1}}. \quad (8)$$

Let $\mathbf{V_a}(j)$ be $\mathbf{V_a}$'s j^{th} component. Since $\mathbf{V_a}(j) = \max \mathbf{a_j Hi}$, $\mathbf{i} \in \mathcal{I}_\mathcal{F}$, and because $\mathcal{I}_\mathcal{F}$ is linearly constrained, $\mathbf{V_a}(j)$ is the solution of a linear program, therefore must occur at a vertex of $\mathcal{I}_\mathcal{F}$ [4]. This is true $\forall j = 0, \ldots, (n-1)$. We express this as:

$$\mathbf{V_a} = \max_{\mathbf{i} \in \mathcal{V}(\mathcal{I}_\mathcal{F})} \mathbf{A^{-1} Hi} \quad (9)$$

where $\mathcal{V}(\mathcal{I}_\mathcal{F})$ denotes the set of vertices of $\mathcal{I}_\mathcal{F}$. Therefore, $\mathbf{V_a}$ can be computed in two steps: 1) Solve $\mathbf{A^{-1} Hi}$ at all vertices of $\mathcal{I}_\mathcal{F}$ and let $\mathcal{S}$ be the set of solution vectors 2) $\mathbf{V_a}(j)$ is the maximum value of the j^{th} component of all vectors in $\mathcal{S}$.

Alone, however, this procedure is insufficient, since the number of vertices in $\mathcal{I}_\mathcal{F}$ may be too large [4]. In what follows, we refer to solving $\mathbf{A^{-1} Hi}$ at a particular vertex $\mathbf{i}$ as *visiting* vertex $\mathbf{i}$.

Our overall strategy is to enlarge $\mathcal{I}_\mathcal{F}$ in a way that requires visiting a user-limited number of vertices in order to compute a conservative estimate of $\mathbf{V_a}$ while limiting the computational cost. The first such enlargement of $\mathcal{I}_\mathcal{F}$ is by forming the intersection of $\mathcal{K}$ with each global constraint separately.

3.2 Applying One Global Constraint at A Time

Noting that $\mathcal{I}_\mathcal{F} \subset \mathcal{K} \cap \mathcal{H}^-_{\mathcal{G}_i}$, $\forall i$, we can write

$$\mathbf{V_a} \leq \min_{i=0,\ldots,c-1} \max_{\mathbf{i} \in \mathcal{V}(\mathcal{K} \cap \mathcal{H}^-_{\mathcal{G}_i})} \mathbf{A^{-1} Hi} \quad (10)$$

where the minimum and maximum are component-wise. This "decoupling" of global constraints reduces our problem to maximizing $\mathbf{A^{-1} Hi}$ over the polytope $\mathcal{K} \cap \mathcal{H}^-_{\mathcal{G}_i}$, formed by the intersection of a single hyperplane with a hypercube. This polytope includes $\mathcal{I}_\mathcal{F}$, so maximizing a function over it overestimates the maximum of that function over $\mathcal{I}_\mathcal{F}$, with the advantage that its vertices are fewer and easier to compute than those of $\mathcal{I}_\mathcal{F}$. The accuracy trade-off is to the extent of overlap among global constraints: if no two global constraints share the same current source, then (10) is an equality. In practice, this is not a problem at the design planning stage since currents represent high-level blocks and global constraints are user-supplied, one could formulate them with little or no overlap, but this is not a requirement of our work – global constraints *may* overlap.

3.3 Vertex Dominance

Not all vertices of $\mathcal{K} \cap \mathcal{H}^-_{\mathcal{G}_i}$ need to be visited: this section shows that it is enough to only visit the vertices which belong to $\mathcal{H}_{\mathcal{G}_i}$, or a subset thereof.

Because $\mathbf{A^{-1}} \geq 0$, we can write

$$\mathbf{i_1} \geq \mathbf{i_2} \Rightarrow \mathbf{A^{-1} Hi_1} \geq \mathbf{A^{-1} Hi_2}. \quad (11)$$

In other words, increasing any coordinate (component) of $\mathbf{i}$ is guaranteed to not decrease any component of $\mathbf{A^{-1} Hi}$.

We say that $\mathbf{i_1}$ *dominates* $\mathbf{i_2}$ if $\mathbf{i_1} \geq \mathbf{i_2}$. Given (11), when looking for $\max \mathbf{A^{-1} Hi}$, it is sufficient to visit $\mathbf{i_1}$, and there is no need to visit $\mathbf{i_2}$. Define the *dimension* of $\mathcal{G}_\rangle$, denoted by $\dim(\mathcal{G}_i)$, as the number of current sources included in $\mathcal{G}_i$. To a vertex $\mathbf{i} \in \mathbb{R}^m$, we associate a rank $r(\mathbf{i}) = \sum_{j=0}^{m-1} \mathbf{i}(j)$, where $\mathbf{i}(j)$ is the j^{th} coordinate of $\mathbf{i}$, i.e., $\mathbf{i} = [\mathbf{i}(0) \ldots \mathbf{i}(m-1)]^T$.

Proposition 1. *If $dim(\mathcal{G}_i) = m$, then*

$$\max_{\mathbf{i} \in \mathcal{V}(\mathcal{K} \cap \mathcal{H}^-_{\mathcal{G}_i})} \mathbf{A^{-1} Hi} = \max_{\mathbf{i} \in \mathcal{V}(\mathcal{K} \cap \mathcal{H}_{\mathcal{G}_i})} \mathbf{A^{-1} Hi} \quad (12)$$

Proof. For clarity, we draw on Fig. 1(a) for reference throughout this proof. For $\mathcal{G}_i$ to be meaningful, $\mathcal{H}_{\mathcal{G}_i}$ must intersect $\mathcal{K}$, i.e., $g_i < \sum_{i=0}^{m-1} l_i$. The vertex with all constraints ON ($\mathbf{i_3}$) must

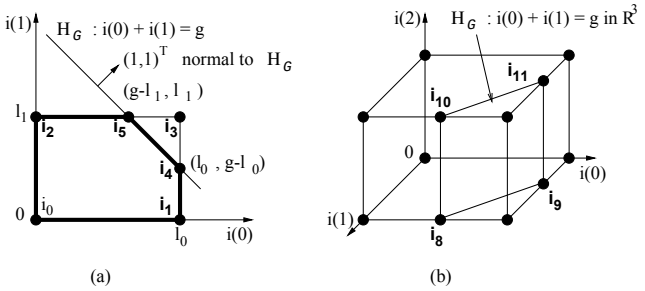

Figure 1: (a) $\mathbf{i_4}$ and $\mathbf{i_5}$ dominate $\mathcal{V}(\mathcal{K} \cap \mathcal{H}^-_\mathcal{G})$, (b) $\mathcal{V}^0_\mathcal{G} = \{\mathbf{i_8}, \mathbf{i_9}\}$, $\mathcal{V}^1_\mathcal{G} = \{\mathbf{i_{10}}, \mathbf{i_{11}}\}$. $\mathcal{V}^1_\mathcal{G}$ dominates $\mathcal{V}(\mathcal{K} \cap \mathcal{H}^-_\mathcal{G})$.

therefore be in $\mathcal{K} \cap \mathcal{H}^+_{\mathcal{G}_i}$, so any vertex of $(\mathcal{K} \cap \mathcal{H}^-_{\mathcal{G}_i}) \setminus \mathcal{H}_{\mathcal{G}_i}$ must have at least one local constraint OFF ($\mathbf{i_0}$, $\mathbf{i_1}$, and $\mathbf{i_2}$).

Define, over $\mathcal{V}(\mathcal{K} \cap \mathcal{H}^-_\mathcal{G})$, the property $\mathcal{P}$: "if any constraint turned OFF in the vertex were turned ON, the resulting vertex would move to $\mathcal{H}^+_{\mathcal{G}_i}$" (e.g., $\mathbf{i_0}$ does not satisfy $\mathcal{P}$, $\mathbf{i_1}$ and $\mathbf{i_2}$ do). Clearly, any vertex of $\mathcal{K} \cap \mathcal{H}^-_\mathcal{G}$ which does not satisfy $\mathcal{P}$ is dominated by another vertex of the same polytope which does. Let $\mathbf{i}$ be a vertex in $(\mathcal{K} \cap \mathcal{H}^-_{\mathcal{G}_i}) \setminus \mathcal{H}_{\mathcal{G}_i}$ satisfying $\mathcal{P}$. Replacing any of the constraints turned off in $\mathbf{i}$ by $g_i - r(\mathbf{i})$ yields a point $\mathbf{i}^* \in \mathcal{H}_{\mathcal{G}_i}$ that dominates $\mathbf{i}$, since $\mathbf{i}^* \geq \mathbf{i}$ ($\mathbf{i_4}$ dominates $\mathbf{i_1}$, $\mathbf{i_5}$ dominates $\mathbf{i_2}$). Say $\mathbf{i}^*$ was obtained by replacing $\mathbf{i}(j)$ with $g_i - r(\mathbf{i})$. To see that $\mathbf{i}^* \in \mathcal{V}(\mathcal{K} \cap \mathcal{H}^-_{\mathcal{G}_i})$, note that $\mathbf{i} \in \mathcal{V}(\mathcal{K})$, thus lies at the intersection of m local constraints, including the 0 hyperplane for the j^{th} local constraint. In $\mathcal{K} \cap \mathcal{H}^-_{\mathcal{G}_i}$, $\mathbf{i}^*$ lies at the intersection of the same m constraints as $\mathbf{i}$, with the only exception of $\mathcal{H}_{\mathcal{G}_i}$ instead of the j^{th} local constraint. Thus, $\mathbf{i}^* \in \mathcal{V}(\mathcal{K} \cap \mathcal{H}^-_{\mathcal{G}_i})$. Therefore, every vertex of $\mathcal{K} \cap \mathcal{H}^-_{\mathcal{G}_i}$ is dominated by another vertex of that same polytope which also happens to be a vertex of $\mathcal{K} \cap \mathcal{H}_{\mathcal{G}_i}$. □

Since $\dim(\mathcal{G}_i) = m$, all vertices of $\mathcal{K} \cap \mathcal{H}_{\mathcal{G}_i}$ share the same rank g_i, so none dominates another (e.g. $\mathbf{i_4}$ and $\mathbf{i_5}$ in Fig. 1(a)). Proposition 1 implies that all vertices of $\mathcal{V}(\mathcal{K} \cap \mathcal{H}_{\mathcal{G}_i})$ must be visited.

We now turn to the case where $\dim(\mathcal{G}_i) = m\prime < m$. We distinguish two subsets of $\mathcal{V}(\mathcal{K} \cap \mathcal{H}_{\mathcal{G}_i})$: $\mathcal{V}^0(\mathcal{K} \cap \mathcal{H}_{\mathcal{G}_i})$ and $\mathcal{V}^1_{\mathcal{G}_i}(\mathcal{K} \cap \mathcal{H}_{\mathcal{G}_i})$, which comprise vertices where all currents not included in $\mathcal{G}_i$ are OFF and ON, respectively (see Fig. 1b). We can make a stronger claim than proposition 1:

Proposition 2. *If $dim(\mathcal{G}_i) = m\prime < m$, then*

$$\max_{\mathbf{i} \in \mathcal{V}(\mathcal{K} \cap \mathcal{H}^-_{\mathcal{G}_i})} \mathbf{A^{-1} Hi} = \max_{\mathbf{i} \in \mathcal{V}^1(\mathcal{K} \cap \mathcal{H}_{\mathcal{G}_i})} \mathbf{A^{-1} Hi} \quad (13)$$

Proof. Similar to the proof of proposition 1, and noting that $\mathcal{V}^1(\mathcal{K} \cap \mathcal{H}_{\mathcal{G}_i})$ dominates $\mathcal{V}(\mathcal{K} \cap \mathcal{H}_{\mathcal{G}_i})$. □

Thus, when $\dim(\mathcal{G}_i) < m$, we only need to visit $\mathcal{V}^1(\mathcal{K} \cap \mathcal{H}_{\mathcal{G}_i})$. Notice that vertices in $\mathcal{V}^0(\mathcal{K} \cap \mathcal{H}_{\mathcal{G}_i})$ can be thought of as being in $\mathbb{R}^{m\prime}$, since $(m - m\prime)$ of their coordinates are 0. Thus, computing the vertices of $\mathcal{V}^1(\mathcal{K} \cap \mathcal{H}_{\mathcal{G}_i})$ follows easily from $\mathcal{V}^0(\mathcal{K} \cap \mathcal{H}_{\mathcal{G}_i})$: compute the vertices of $\mathcal{V}^0(\mathcal{K} \cap \mathcal{H}_{\mathcal{G}_i})$ in $\mathbb{R}^{m\prime}$, then for each such vertex, set every coordinate corresponding to a current source not included in $\mathcal{G}_i$ to its local constraint value, yielding the vertices of $\mathcal{V}^1(\mathcal{K} \cap \mathcal{H}_{\mathcal{G}_i})$ in $\mathbb{R}^m$. In the rest of this paper, we may refer to m as the dimension of a global constraint and discuss the computation of vertices of $\mathcal{K} \cap \mathcal{H}_{\mathcal{G}_i}$ in $\mathbb{R}^m$, with the understanding that, if $\dim(\mathcal{G}_i) = m\prime < m$, we first compute the vertices of $\mathcal{V}^0(\mathcal{K} \cap \mathcal{H}_{\mathcal{G}_i})$ in $\mathbb{R}^{m\prime}$, then deduce those of $\mathcal{V}^1(\mathcal{K} \cap \mathcal{H}_{\mathcal{G}_i})$ in $\mathbb{R}^m$.

3.4 Global Constraint Relaxation

Based on the above, our problem reduces to visiting the vertices at the intersection of a global constraint hyperplane with the local constraint hypercube. However, the number of these vertices may

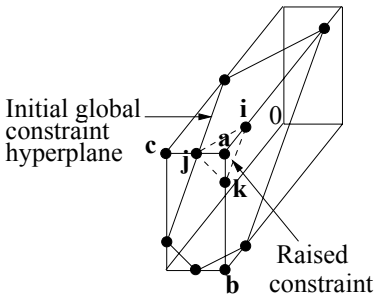

Figure 2: Before the raise, a, b, and c are left out of $\mathcal{G}$ and $\mathcal{K} \cap \mathcal{H}_{\mathcal{G}}$ includes five vertices. The raised hyperplane leaves only a out, and its three induced vertices i, j, and k, dominate $\mathcal{V}(\mathcal{K} \cap \mathcal{H}_{\mathcal{G}})$.

be too large and their computation expensive [4]. We seek to find a small, user-controlled number of current points that dominate $\mathcal{V}(\mathcal{K} \cap \mathcal{H}_{\mathcal{G}_i})$, such that visiting these currents could be done as quickly as the user can afford, while the estimate of $\max \mathbf{A^{-1}Hi}$ from these currents would be conservative.

The basic idea is to enlarge the polytope $\mathcal{K} \cap \mathcal{H}^-_{\mathcal{G}_i}$ by suitably shifting the hyperplane $\mathcal{H}_{\mathcal{G}_i}$ in such a way that the shifted hyperplane $\mathcal{H}^*_{\mathcal{G}_i}$ satisfies $\mathcal{H}^-_{\mathcal{G}_i} \subset \mathcal{H}^{*-}_{\mathcal{G}_i}$. Then, the vertices of $\mathcal{K} \cap \mathcal{H}^*_{\mathcal{G}_i}$ would dominate those of $\mathcal{K} \cap \mathcal{H}_{\mathcal{G}_i}$. We refer to this process as *relaxing the global constraint*. The question becomes how to relax $\mathcal{G}_i$ in such a way that the vertices of $\mathcal{K} \cap \mathcal{H}^*_{\mathcal{G}_i}$ are limited in number and easy to compute.

We say that a constraint $\mathcal{G}_i$ (or its hyperplane $\mathcal{H}_{\mathcal{G}_i}$) *leaves out* a vertex $\mathbf{i}$ of $\mathcal{K}$ if $\mathbf{i} \in \mathcal{H}^+_{\mathcal{G}_i}$, otherwise, we say that the vertex is *in the constraint*. The key lies in the fact that we can control the number of vertices of $\mathcal{K} \cap \mathcal{H}_{\mathcal{G}_i}$ by controlling the number of vertices of $\mathcal{K}$ left out of $\mathcal{G}_i$, based on the following theorem [4]:

Theorem 2. *Consider a polytope $\mathcal{D}$ and a hyperplane $\mathcal{H}$. A point v is a vertex of $\mathcal{D} \cap \mathcal{H}$ if and only if v is either a vertex of $\mathcal{D}$ lying in $\mathcal{H}$ or a point at which an edge (u, w) of $\mathcal{D}$ intersects $\mathcal{H}$, such that $u \in \mathcal{H}^-$ and $w \in \mathcal{H}^+$.*

Since m edges are incident to every vertex of $\mathcal{K}$, theorem 2 implies that, if $\mathcal{H}^{*+}_{\mathcal{G}_i}$ includes k vertices of $\mathcal{K}$, then the number of vertices of $\mathcal{K} \cap \mathcal{H}^*_{\mathcal{G}_i}$ cannot exceed km.

One way to guarantee that $\mathcal{H}^-_{\mathcal{G}_i} \subset \mathcal{H}^{*-}_{\mathcal{G}_i}$ is to increase g_i to g_i^*, which amounts to *raising* $\mathcal{H}_{\mathcal{G}_i}$ parallel to itself, just enough for the raised constraint $\mathcal{G}_i^*$ or its hyperplane $\mathcal{H}^*_{\mathcal{G}_i}$ to leave out at most k vertices of $\mathcal{K}$. Then the number of vertices of $\mathcal{K} \cap \mathcal{H}^*_{\mathcal{G}_i}$ cannot exceed km. See Fig. 2. A key feature is that the user controls the maximum computational cost with the choice of k. The problem is to find the minimum g_i^* necessary and to compute $\mathcal{V}(\mathcal{K} \cap \mathcal{H}^*_{\mathcal{G}_i})$. We refer to this relaxation as a *constraint raise*.

4. TOP-LEVEL ALGORITHM

Recall that visiting a vertex $\mathbf{i}$ means solving the linear system $\mathbf{Av} = \mathbf{Hi}$. Given a factorization of $\mathbf{A}$, this would require a forward/backward substitution at each vertex visited. Although the number of vertices is user-controlled through the choice of k, it can greatly surpass the number m of current sources. If a forward/backward substitution, which computes the n-vector of power grid voltage drops and is $O(n^2)$, were applied for each vertex visited, visiting all vertices would become too expensive. This, however, is unnecessary and the number of forward/backward substitutions does not need to exceed m: If we choose a basis of vectors in $\mathbb{R}^m$, i.e., m linearly independent m-vectors, we only need a forward/backward substitution at each of the bases (m in total). The solution at any other point in $\mathbb{R}^m$, particularly at a vertex, can be constructed from the solution of the bases, by virtue of system linearity, with the more efficient *daxpy* operation[3], which, for vectors $\mathbf{x}$ and $\mathbf{y}$ and a scalar α, computes $\alpha\mathbf{x}+\mathbf{y}$ in $O(n)$. We naturally chose the bases $\mathbf{e_j}$, $j = 0, \ldots, (m-1)$, for which the j^{th} component is 1 and all others 0.

Algorithm 1 describes the overall solution. $\mathcal{V}_i^{*+}$ is the set of current vertices left out of $\mathcal{G}_i^*$, and of cardinality k. As per section 3.3, $\mathcal{V}_i^{*+}$ need only contain vertices where local constraints not included in $\mathcal{G}_i$ are ON. $\mathcal{I}_i$ is the set of indices of current sources included in $\mathcal{G}_i$ and l_j is the j^{th} local constraint.

Algorithm 1 returns a vector $\mathbf{V_{ub}}$ of upper bounds on the maximum voltage drop on all the grid nodes.

1: Compute $\mathbf{b_j} = \mathbf{A^{-1}He_j}$, $j = 0, \ldots, m-1$
2: **for** $i = 0, \ldots, c-1$
3: Raise $\mathcal{H}_{\mathcal{G}_i}$ to form $\mathcal{H}^*_{\mathcal{G}_i}$ and $\mathcal{V}_i^{*+}$
4: Set $\mathbf{V_i^*} = \mathbf{0}$, $\widetilde{\mathbf{V}}_\mathbf{i} = \sum_{j \notin \mathcal{I}_i} l_j \mathbf{b_j}$, $\tilde{l}_i = \sum_{j \notin \mathcal{I}_i} l_j$
5: **for all** $\mathbf{i}^+ \in \mathcal{V}_i^{*+}$
6: $\mathbf{V}_{\mathbf{i}^+} = \sum_{j \in \mathcal{I}_i} \mathbf{i}^+(j)\mathbf{b_j}$
7: **for all** $j \in \mathcal{I}_i$
8: **if** $r(\mathbf{i}^+) - \mathbf{i}^+(j) - \widetilde{\mathbf{V}}_\mathbf{i} \leq g_i^*$ **then**
9: $\mathbf{V}^* = \widetilde{\mathbf{V}}_\mathbf{i} + \mathbf{V}_{\mathbf{i}^+} - (r(\mathbf{i}^+) - \tilde{l}_i - g_i^*)\mathbf{b_j}$
10: $\mathbf{V_i^*} = \max(\mathbf{V_i^*}, \mathbf{V}^*)$
11: $\mathbf{V_a} \leq \min_{i=0,\ldots,c-1} \mathbf{V_i^*} = \mathbf{V_{a,ub}}$
12: RC: $\mathbf{V_{ub}} = (\mathbf{I} + \mathbf{G^{-1}C}/\Delta t)\mathbf{V_{a,ub}}$, DC: $\mathbf{V_{ub}} = \mathbf{V_{a,ub}}$

$\mathbf{V_i^*}$ is the vector of maximum voltage drops induced by $\mathcal{G}_i^*$. Let $\mathbf{i}^-$ be the j^{th} neighbor of a vertex $\mathbf{i}^+ \in \mathcal{K} \cap \mathcal{H}^+_{\mathcal{G}_i^*}$, obtained by changing $\mathbf{i}^+(j)$ to 0 if $\mathbf{i}^+(j) = l_j$. $\mathbf{i}^-$ may be in $\mathcal{G}_i^*$ *only if* $j \in \mathcal{I}_i$ (line 7), and is in $\mathcal{G}_i^*$ *if* the condition on line 8 is met. In this case, $\exists \mathbf{i}^* \in \mathcal{V}(\mathcal{K} \cap \mathcal{H}^*_{\mathcal{G}_i})$ along the edge from $\mathbf{i}^+$ to $\mathbf{i}^-$. $\mathbf{i}^*$ shares all the coordinates of $\mathbf{i}^+$, except $\mathbf{i}(j)$, and is such that the sum of its coordinates corresponding to currents included in $\mathcal{I}_i$ adds up to g_i^*. Therefore, $\mathbf{i}^*(j) = g_i^* - (r(\mathbf{i}^+) - \tilde{l}_i - \mathbf{i}^+(j))$. Letting $\mathbf{V}^*$ denote the voltage drop at $\mathbf{i}^*$, we have that $\mathbf{V}^* = \widetilde{\mathbf{V}}_\mathbf{i} + \mathbf{V}_{\mathbf{i}^+} - (\mathbf{i}^+(j) - \mathbf{i}^*(j))\mathbf{b_j}$, which reduces to the expression on line 9. Notice that daxpies are performed on lines 4 and 6 to compute $\widetilde{\mathbf{V}}_\mathbf{i}$ and $\mathbf{V}_{\mathbf{i}^+}$.

4.1 Complexity

In algorithm 1, we pay a one-time-cost $O(mn^2)$ to compute the $\mathbf{b_h}$s by m forward/backward substitutions (line 1). Forming the raise (line 3) can be done in $O(km)$ as we show in section 5, and the computation of $\widetilde{\mathbf{V}}_\mathbf{i}$ (line 4) is $O(mn)$. The innermost **for** loop (line 7) performs $O(n)$ operations on lines 9 and 10, and iterates at most m times, once for every neighbor of $\mathbf{i}^+$, adding up to $O(mn)$. Line 6 is also $O(mn)$. The **for** loop on line 5 iterates k times, and is thus $O(kmn)$. Therefore, the outer **for** loop (line 2) is $O(ckmn)$. Crucially, this is linear in all parameters, notably the grid size n. Contrast that with the cost of n LPs on $\mathbb{R}^n$. The algorithm of section 5 requires that the local constraints be sorted, which is a standard $O(m \log m)$ operation, that needs to be done only once in algorithm 1, and does not affect the overall complexity. Finally, algorithm 1 requires storage of the m n-vectors $\mathbf{b_j}$, which can create a memory bottleneck for large n. However, grid locality [5] enables the "sparsification" of each $\mathbf{b_j}$ by observing that $\mathbf{b_j}$ represents the power grid response to a single current source excitation, the bulk of which is confined to a relatively small neighborhood surrounding the location of the excitation. Therefore, only a small fraction of the n entries of each $\mathbf{b_h}$ effectively need to be stored.

5. MINIMUM RAISE COMPUTATION

For simplicity of presentation, we assume without loss of generality, that $\dim(\mathcal{G}) = m$, and that $\mathcal{K}$ and $\mathbf{i}$ are a hypercube and a vertex in $\mathbb{R}^m$. If $\dim(\mathcal{G}) = m' < m$, we would be implicitly working in $\mathbb{R}^{m'}$ (see the discussion at the end of section 3.3).

Observe that if $\mathcal{G}$ leaves out a vertex $\mathbf{i}$ of $\mathcal{K}$, then it must leave out all vertices of rank greater than $r(\mathbf{i})$. Clearly, the vertex $\mathbf{i_0} = [l_0\ l_1 \dots l_{m-1}]^T$, with all local constraints ON, is the rank-wise largest vertex of $\mathcal{K}$. Let us denote the vertex with the j^{th} largest rank, after $r(\mathbf{i_0})$, by $\mathbf{i_j}$. We say that j is the *order* of $\mathbf{i_j}$ (0 is the order of $\mathbf{i_0}$). If we knew the sequence $\mathcal{R} = \{\mathbf{i_1}, \dots, \mathbf{i_k}\}$, we could set g^* to $r(\mathbf{i_k})$, which would be the minimum value for which k vertices $(\mathbf{i_0}, \dots, \mathbf{i_{k-1}})$ are left out of the raised global constraint. For this to hold, no vertex outside of $\mathcal{R}$ can have a strictly larger rank than any vertex in $\mathcal{R}$. If, furthermore, $r(\mathbf{i_k}) = r(\mathbf{i_{k-1}})$, then in order to ensure that k is an upper bound on the number of vertices left out of $\mathcal{G}$, we must find the largest $k' < k$ for which $r(\mathbf{i}_{k'}) < r(\mathbf{i}_{k'-1})$, and let $g^* = r(\mathbf{i}_{k'})$.

Since $\mathbf{i}(j)$ is either 0 or l_j, we can uniquely identify $\mathbf{i}$ by the set of local constraints that are ON in $\mathbf{i}$. The problem becomes to find which constraints to turn ON to obtain k combinations with maximal total sum. This likens our problem to the Subset Sum Problem (SSP) family of knapsacks [6]: we can view a vertex $\mathbf{i}$ for which l_j is ON as a packing of the knapsack that includes item l_j, and we want to find the k packings with maximal total value. But contrary to a usual SSP, we specifically seek k packings of maximal value. It is possible to tackle this problem naively by applying iteratively some of the known SSP algorithms. However, the resemblance to SSP motivates the use of dynamic programming to tailor a specific algorithm for the problem at hand, as discussed in this section.

Our algorithm assumes the local constraints are sorted in ascending order, i.e., $l_i \leq l_j$ if $i < j$. We use an m-column table $\mathbf{T}$, each entry which represents a *unique* vertex of $\mathcal{K}$. Conceptually, entries of $\mathbf{T}$ fall into two groups: *non-empty entries* and *empty entries*. $\mathbf{T}$ is formed so that $\mathbf{i_1}, \dots, \mathbf{i_k}$ are represented by non-empty entries. A non-empty entry is *defined as an entry which is "bound to" a vertex* of $\mathcal{K}$ (for convenience, we will also say that the vertex is bound to the non-empty entry). When a vertex is bound to an entry, we say the entry is *filled*. There will be a 1-to-1 correspondence between a non-empty entry and a vertex of $\mathcal{K}$. In the description below, we rely extensively on this 1-to-1 correspondence, and refer to an entry of $\mathbf{T}$ and the vertex to which it is bound interchangeably, e.g., the rank of a non-empty entry refers to the rank of the vertex to which it is bound.

We say that a non-empty entry of $\mathbf{T}$ is *ordered* if the order of the vertex to which it is bound is known. We denote this order by $\Phi(T_{i,j})$, and we let $\Phi^{-1}(x)$ be the table entry bound to the vertex of order x. At any given point in the algorithm, the set of non-empty but unordered entries in $\mathbf{T}$ forms the *frontier* of $\mathbf{T}$, denoted $\mathcal{F}$. We denote by $\mathcal{F}^+$ the non-empty entries which are not on $\mathcal{F}$, i.e., the set of ordered vertices, and by $\mathcal{F}^-$ the remaining vertices of $\mathcal{K}$, which correspond to empty entries in $\mathbf{T}$. Our aim is to have k entries in $\mathcal{F}^+$. Let $T_{i,j}$ $(i, j \geq 0)$ denote the $(i,j)^{\text{th}}$ entry of $\mathbf{T}$, and $T_{:,j}$ be $\mathbf{T}$'s j^{th} column.

We impose the following key condition: $T_{i,j}$ can be bound to a vertex $\mathbf{i}$ of $\mathcal{K}$ if and only if $\mathbf{i}(j) = 0$ and $\mathbf{i}(j') = l_{j'}$, $\forall j' > j$. A vertex $\mathbf{i}$ which can be bound to any entry in $T_{:,j}$ is said to *belong to* $T_{:,j}$, denoted by $\mathbf{i} \in T_{:,j}$. Therefore, 2^j vertices belong to $T_{:,j}$. We say that $T_{:,j}$ is *complete* if its 2^j entries are in $\mathcal{F}^+$.

Recall that a vertex $\mathbf{i}$ is completely determined by the set of current constraints that are turned ON in $\mathbf{i}$. Denoting this set by $\mathcal{L}(\mathbf{i})$, we can write $\mathbf{i} = \cup_{j \in \mathcal{L}(\mathbf{i})}\{l_j\}$. Suppose $\mathbf{i_1} \in T_{:,j_1}$ and consider the *unique* vertex $\mathbf{i_2} \in T_{:,j_2}$, $j_2 < j_1$, such that $\mathbf{i_1} = \mathbf{i_2} \setminus \{l_{j_1}\}$. In this case, we say that $\mathbf{i_1}$ *is paired with* $\mathbf{i_2}$ and write $\mathbf{i_2} = \Pi(\mathbf{i_1})$. If $\mathbf{i_1}$ is bound to T_{i_1,j_1} and $\mathbf{i_2}$ is bound to T_{i_2,j_2}, we similarly write $T_{i_2,j_2} = \Pi(T_{i_1,j_1})$. Observe that a vertex is completely specified by the column to which it belongs and the vertex with which it is paired. It will be convenient, for notational purposes, to define T_0, a conceptual table entry of order 0, bound to $\mathbf{i_0}$: it does not have a row or column, but pairing an entry in $T_{:,j}$ with T_0 forms the vertex $\cup_{j' \neq j}\{l_{j'}\}$. Let $\mathbf{T}\text{–}to\text{–}\mathcal{V}(.)$ be an operator which takes a table entry and returns the vertex to which it is bound. It can be defined recursively as $\mathbf{T}\text{–}to\text{–}\mathcal{V}(T_{i,j}) = \mathbf{T}\text{–}to\text{–}\mathcal{V}(\Pi(T_{i,j})) \setminus \{l_j\}$.

Algorithm 2 describes the procedure. The key idea is that as we go south on any given column of $\mathbf{T}$, the ranks (orders) of the vertices bound to table entries are non-increasing (increasing).

Algorithm 2 minimum raise computation

```
1: Set F+ = F = ∅, F− = T
2: for j = 0, ..., m − 1
3:    Set Π(T_{0,j}) = T_0
4:    Set F = F ∪ T_{0,j}, F− = F− \ T_{0,j}
5:    Set needs_pairing(j) = false
6: while |F+| = t < k
7:    Let T_{i*,j*} be the leftmost entry of maximal rank in F
8:    Set i^(t+1) = i*, j^(t+1) = j*, i_{t+1} = T−to−V(T_{i*,j*})
9:    Set F+ = F+ ∪ T_{i*,j*}, F = F \ T_{i*,j*}
10:   if T_{:,j*} is not complete then
11:      if j* ≥ j^(t) and needs_pairing(j^(t)) == false then
12:         Set needs_pairing(j^(t)) = true
13:         Set needs_pairing_at_row(j^(t)) = i^(t)
14:      for j = j* + 1, ..., m − 1
15:         if needs_pairing(j) == true then
16:            Set i = needs_pairing_at_row(j)
17:            Set jump_to(T_{i,j}) = t + 1
18:            Set needs_pairing(j) = false
19:      Set jump_to(T_{i*,j*}) = t + 2
20:      Let T_{i1,j1} = Φ^{-1}(Φ(Π(T_{i*,j*})) + 1)
21:      if j1 ≥ j* then
22:         Set t' = jump_to(T_{i1,j1}), let T_{i1,j1} = T_{i^(t'),j^(t')}
23:      Fill T_{i*+1,j*} such that Π(T_{i*+1,j*}) = T_{i1,j1}
24:      Set F = F ∪ T_{i*+1,j*}, F− = F− \ T_{i*+1,j*}
25: return R = {i_1, ..., i_k}
```

Initialization occurs on lines 1–5, where row 0 in every column is initialized with the vertex of largest rank that belongs to that column, obtained by a pairing with T_0 (line 3). $\mathcal{F}$ is set to row 0 of $\mathbf{T}$ (line 4). The algorithm works by ensuring that an unordered vertex of maximal rank must exist on $\mathcal{F}$ at any given stage. Iteratively, the loop starting on line 6 picks the table entry T_{i^*,j^*} bound to such a vertex (line 7) and outputs the vertex $\mathbf{i_{t+1}}$ (line 8), then moves T_{i^*,j^*} from $\mathcal{F}$ to $\mathcal{F}^+$ (line 9). Lines 10–24 result in a *shift of the frontier* by 1 row south on $T_{:,j^*}$ (line 24), by binding a suitably chosen vertex to $T_{i^*+1,j}$ (line 23).

Key features of the algorithm are that $\mathcal{F}$ comprises the south-most non-empty and non-ordered entries in each non-complete column, and that $\mathcal{F}$ shifts in each iteration by a single entry south on the column $T_{:,j^*}$ to which the most-recently-ordered vertex, T_{i^*,j^*}, belongs (unless $T_{:,j^*}$ is complete). The vertex to be bound to T_{i^*+1,j^*} is selected so that it must be a vertex of maximal rank among the unbound vertices belonging to $T_{:,j^*}$. Lines 10–22 satisfy this purpose. We can summarize their basic idea as follows: Recall that an entry is always paired with another from a lower-indexed column. We first attempt to pair T_{i^*+1,j^*} with the vertex of order equal to the order of the vertex with which T_{i,j^*} is paired, plus 1 (line 20). a) If this vertex is bound to an entry in columns $0, \dots, (j^* - 1)$, then the pairing is successful and T_{i^*+1,j^*} is filled with this pairing (lines 20 and 23). b) Otherwise, T_{i^*+1,j^*} is to be paired with the entry in columns $0, \dots, (j^* - 1)$, bound to the vertex of smallest order with which no other entry in $T_{:,j^*}$ is paired. The entry with which T_{i^*+1,j^*} is to be paired is stored in the field *jump_to* (lines 21–22), which, alongside *needs_pairing* and *needs_pairing_at_row*, perform book-keeping to retrieve efficiently the entry with which to form the pairing in case b).

Due to space restrictions, we omit a detailed proof of correctness for algorithm 2. For complexity analysis, we note that the initialization (lines 1–5) is $O(m)$. For each iteration of the **while** loop (line 6), we have: 1) Line 7 is $O(m)$, since every column includes at most 1 entry in $\mathcal{F}$, 2) Line 8 is $O(m)$, since we output $\mathbf{i_1}, \dots, \mathbf{i_k}$ sequentially, a vertex is always output after that with which it is paired, so that the application of $\mathbf{T}\text{–}to\text{–}\mathcal{V}$ is $O(m)$, 3) The inner **for** loop (line 14) is $O(m)$, and 4) Everything else is $O(1)$. This results in an asymptotic runtime complexity $O(km)$.

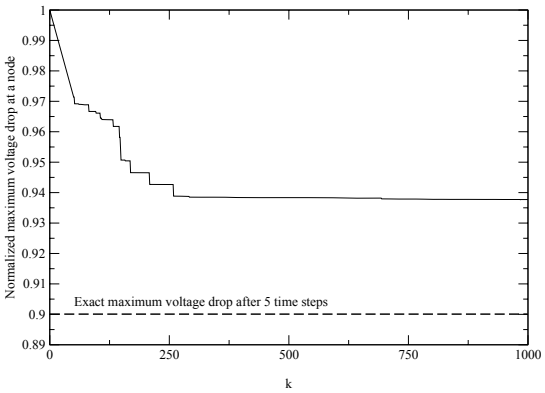

Figure 3: Estimation of the maximum voltage drop at a given node with increasing k.

Grid size (#nodes)	Analysis type	Runtime (sequential LPs)	Runtime (proposed method)	Average overestimation(%)
900	TR	2.5 min.	1 sec.	1.1
3,800	TR	45 min.	15 sec.	0.4
28,000	DC	16 sec./node (est. 5 days)	2.3 min.	0.25 (est.)
48,500	DC	20 sec./node (est. 11 days)	6.5 min.	0.30 (est.)
261,000	TR	–	1.1 hr.	–
590,000	TR	–	3.0 hr.	–

Table 1: Accuracy and speed comparisons of the proposed approach with sequential linear programs.

In terms of memory footprint, notice that we only need to store non-empty entries, k of which are ordered $(\mathbf{i_1}, \ldots, \mathbf{i_k})$ and up to m unordered (on $\mathcal{F}$). This results in space requirement $O(k+m)$.

6. EXPERIMENTAL RESULTS

To test our method, we wrote a C++ tool that generates power grids from user specifications, including grid dimensions, metal layers (M1 – M9), pitch and width per layer, and C4 and current source distribution. A global constraint is specified by the spatial region and metal layers it includes. All the reported results are on grids with seven global constraints covering the entirety of the grid area, and runtimes follow the application of all seven global constraints. Minimum spacing and sheet and via resistances were specified according to a 90 nm technology. All results were obtained on a Sun Fire X2200 M2 server, with two dual core, 64-bit, 2.6-GHz, AMD Opteron processors, and with 8 GB of memory.

Fig 3 shows, for a given node on a 650-node grid with 121 currents sources, the progress of voltage drop estimation with increasing k. The value at $k = 0$ corresponds to the voltage drop when all currents are set to their local constraints, i.e., when global constraints are ignored. All estimates of voltage drop are normalized to this value. The horizontal line corresponds to the value of the exact maximum voltage drop at the node after 5 time steps, computed from a linear program of size 650×5, as in [2], and amounts to 90% of the value when global constraints are ignored. We note that taking more time steps will only increase this number [2]. After $k = 1000$, our estimate is about 93.5%. Therefore, the overestimation due to not considering global constraints has been reduced by (at least) 65%.

Table 1 compares results of the proposed method with the execution of n LPs sequentially, one LP per node, for transient (TR) and DC analysis. The number of current sources ranged between 120 and 240, and k was set to 3000. For transient analysis, column 3 refers to the time it took to compute $\mathbf{V_u}$ (upper bounds on node voltage drops), which is essentially the cost of computing $\mathbf{V_a}$ by n LPs (see the corollary of theorem 1). Since the full vector $\mathbf{V_a}$ is needed for the computation of $\mathbf{V_u}$, solving all n LPs was necessary, thus limiting the size of grids on which this comparison could be made. Accuracy estimates were obtained on larger grids through DC analysis, where we could test our method on a node-by-node basis, without the need for all n LP solutions. For this purpose, we chose 20 random nodes on the grid, solved the 20 corresponding LPs, and estimated runtime (column 3) and accuracy (column 5) based on these 20 nodes, indicated by "*est*". Given the impractical runtimes of sequential LPs if the full-grid solution were needed, our method simply makes checking a power grid under uncertain, constraint-specified currents, a feasible and practical proposition. The last two rows of the table show larger grids, where even a single LP is too expensive to run, precluding accuracy estimation, but illustrating the ability of our technique to handle efficiently larger problems.

7. CONCLUSION

As power grid safety becomes increasingly important in modern integrated circuits, so does the need to start power grid verification early in the design cycle and incorporate circuit uncertainty at the early stages into useful power grid information. In this work, we adopt the framework of capturing circuit uncertainty via constraints on currents and maximizing node voltage drops over the constraint space. We propose a geometric approach to transform a problem whose solution requires as many linear programs as there are nodes, to another involving a user-limited number of solutions of a single linear system. The key is to determine, in the current space, and from the geometry of the optimization problem, a limited number of points at which to solve the power grid system, and to derive, from these solutions, conservative estimates of the power grid voltage drops.

This approach made the problem of prototyping the full power grid under uncertain currents practicable and scalable. Prior art simply couldn't handle large grids of more than a few hundred or thousand nodes, suffering too prohibitive runtimes for any larger systems. This work achieves runtime improvement manyfold, doing in seconds what required hours, and completing the prototyping of grids of about half a million nodes in a couple of hours. This comes at the expense of some accuracy loss. However, our approach is designed so that all non-accurate estimates are guaranteed conservative, and our results showed that inaccuracy was relatively small, rarely exceeding 1% on average.

Acknowledgment

The authors thank Dr. Ulrich Pferschy from the University of Graz, Austria, and Rami Beidas and Hratch Mangassarian from the University of Toronto for their helpful comments on the algorithm in section 5.

8. REFERENCES

[1] A. Dharchoudhury, R. Panda, D. Blaauw, R. Vaidyanathan, B. Tutuianu, and D. Bearden. Design and analysis of power distribution networks in PowerPC^{TM} microprocessors. In *ACM/IEEE Design Automation Conference*, pages 738–743, San Francisco, CA, June 15-19 1998.

[2] M. Nizam, F. N. Najm, and A. Devgan. Power grid voltage integrity verification. In *ACM/IEEE International Symposium on Low Power Electronics and Design*, pages 239–244, San Diego, CA, August 8-10 2005.

[3] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, PA, 2nd edition, 2003.

[4] R. Horst, P. P. Pardalos, and N. V. Thoai. *Introduction to Global Optimization*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2nd edition, 2000.

[5] E. Chiprout. Fast flip-chip power grid analysis via locality and grid shells. In *IEEE/ACM International Conference on Computer-Aided Design*, pages 485–488, San Jose, CA, November 7-11 2004.

[6] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack problems*. Springer-Verlag, Berlin, Germany, 2004.