An Efficient Dual Algorithm for Vectorless Power Grid Verification under Linear Current Constraints

Xuanxing Xiong and Jia Wang
Electrical and Computer Engineering Department
Illinois Institute of Technology, Chicago, IL 60616, USA

ABSTRACT

Vectorless power grid verification makes it possible to evaluate worst-case voltage drops without enumerating possible current waveforms. Under linear current constraints, the vectorless power grid verification problem can be formulated and solved as a linear programming (LP) problem. However, previous approaches suffer from long runtime due to the large problem size. In this paper, we design the DualVD algorithm that efficiently computes the worst-case voltage drops in an RC power grid. Our algorithm combines a novel dual approach to solve the LP problem, and a preconditioned conjugate gradient power grid analyzer. Our dual approach exploits the structure of the problem to simplify its dual problem into a convex problem, which is then solved by the cutting-plane method. Experimental results show that our algorithm is extremely efficient – it takes less than an hour to complete the verification of a power grid with more than $50K$ nodes and it takes less than 1 second to verify one node in a power grid with more than $500K$ nodes.

Categories and Subject Descriptors:
B.7.2 [Integrated Circuits]: Design Aids
General Terms: Algorithms, Verification
Keywords: Power grid, voltage drop, linear programming

1. INTRODUCTION

The power grid of integrated circuits (IC) must provide sufficient voltage at each gate in order to guarantee the correct operation of the circuit. It is indispensable to verify that the power supply noise at each node is acceptable when designing ICs, and this process is typically called *power grid verification*. The power supply noise is mainly due to IR drop and Ldi/dt drop, which are exaggerated with the development of deep sub-micron technologies because of the increasing wire resistance and clock frequency. In addition, the decreasing circuit supply voltage and the transistor threshold voltage make the circuit more vulnerable to power supply noises. Therefore, it is desired that the power grid verification techniques can provide accurate estimation of the noise at each node in a timely manner.

Conventionally, the power grid is verified by simulating the circuit to evaluate the noise at each node. Simulation based techniques need specific circuit implementations to provide detailed input current waveform, and can only be employed after the circuit design is done. However, in practice, early power grid verification is preferred for grid modification. Moreover, simulation based techniques suffer from the fundamental limitation of being either pessimistic or computationally prohibitive to cover all possible input current waveforms.

DAC 2010, June 13-18, 2010, Anaheim, California, USA.

To address such requirements, vectorless power grid verification has been proposed [1, 2, 3, 4] to evaluate the worst-case power supply noise without explicitly enumerating all input current waveforms. In [1, 2, 3], feasible input current waveforms are characterized into linear current constraints to cover all possible input current waveform, such that the maximum voltage drops in an RC power grid can be obtained after formulating and solving linear programming (LP) problems. In [4], the assumptions are further refined by introducing integer variables to model the uncertain working modes of circuit blocks, and integer linear programming (ILP) problems are formulated and solved. Unfortunately, solving all of these problems is time-consuming or even prohibitive for large grids. [3] takes advantage of the grid locality to generate a reduced-size LP problem for each node, and accepts a user specified over estimation margin to control the precision of solutions. This approach sacrifices the accuracy in order to achieve speed-up by reducing the number of variables in the LP problems, so it is not applicable when high precision is desired. Moreover, the speed-up is limited since to generate the reduced-size LP problems takes extra runtime.

In this paper, we design the DualVD algorithm to solve the vectorless power grid verification problem under linear current constraints for RC power grids. We propose a novel dual approach to solve the associated LP problem efficiently by exploiting its special structure. We simplify the dual problem of the LP problem by eliminating most decision variables and constraints. We show the simplified dual problem is a convex problem with a small number of decision variables and simple form of constraints. We then solve the convex problem by the cutting-plane method and provide means to speed-up the evaluation of the objective function and its subgradient. This dual approach is then combined with a preconditioned conjugate gradient power grid analyzer based on [5]. Experimental results show that the proposed dual approach is much faster than solving the LP problems directly, and our DualVD algorithm is much more efficient when compared with the method in [3].

The rest of the paper is organized as following. The problem formulation is introduced in Section 2. The algorithmic idea is developed in Section 3. The technical details of the proposed algorithm is presented in Section 4. After experimental results are shown in Section 5, we conclude the paper in Section 6.

2. PROBLEM FORMULATION

The vectorless power grid verification problem under linear current constraints was proposed in [1] and studied in [2, 3]. This problem considers an RC power grid consisting of resistors, capacitors, VDD pads, and currents drawn by gates. Let $\mathbf{v}(t)$ be the vector of the voltage drops of the nodes that are not VDD pads, $\mathbf{i}(t)$ be the vector of all the current sources attached to the aforementioned non-VDD nodes, G be the conductance matrix, and C be the matrix introduced by capacitors. According to [1], the following

system of equations must hold.

$$G\mathbf{v}(t) + C\dot{\mathbf{v}}(t) = \mathbf{i}(t). \quad (1)$$

Note that if we assume that there are n non-VDD nodes, then both $\mathbf{v}(t)$ and $\mathbf{i}(t)$ are $n \times 1$ vectors, G is an $n \times n$ symmetric M-matrix, and C is an $n \times n$ diagonal matrix.

Clearly, the amount of voltage drops depends on the waveform of current excitations. As it is pessimistic to assume a peak current for each current source and computationally prohibitive to enumerate all possible current waveforms, current constraints are introduced to capture the infinite many current excitations in an optimization framework. There are two kinds of current constraints: *local constraints* and *global constraints*. The local constraints define an upper-bound for every current source,

$$0 \le \mathbf{i}(t) \le \mathbf{I}_L, \forall t,$$

where $\mathbf{I}_L \ge 0$ is an $n \times 1$ upper-bound vector. The global constraints define upper bounds for groups of current sources,

$$U\mathbf{i}(t) \le \mathbf{I}_G, \forall t.$$

Here we assume that there are m current source groups and m is much smaller than n, U is an $m \times n$ 0/1 matrix indicating the assignments of current sources to groups, and $\mathbf{I}_G \ge 0$ is an $m \times 1$ upper-bound vector.

As summarized in [3], vectorless power grid verification can be performed under two power grid analysis models, i.e. the DC analysis model and the transient analysis model with a time-step h, and the key problem can be formulated as the following *maxVD-LCC (maximum voltage drop under linear current constraints)* problem.

PROBLEM 1 (MAXVD-LCC). *Assume there are n non-VDD nodes and m global current constraints. Let $A = G$ if the DC analysis model is of concern or $A = G + \frac{C}{h}$ if the transient analysis model is of concern. Let $\mathbf{I}_L \ge 0$ and $\mathbf{I}_G \ge 0$ be the upper-bound vectors for local and global current constraints respectively. Let U be a 0/1 matrix. Suppose that $\mathbf{v}$ and $\mathbf{i}$ are the vectors of decision variables and let v_l be the l'th component of $\mathbf{v}$ for $1 \le l \le n$. Solve the following optimization problem for every $1 \le l \le n$,*

$$\textit{Maximize } v_l \quad \textit{s.t.} \quad A\mathbf{v} = \mathbf{i}, 0 \le \mathbf{i} \le \mathbf{I}_L, U\mathbf{i} \le \mathbf{I}_G. \quad (2)$$

3. ALGORITHM OVERVIEW

For every $1 \le l \le n$, the optimization problem in Eq.(2) is a linear programming (LP) problem since both the objective function and the constraints are linear functions of the decision variables. As there are n optimization problems and n is usually large for any practical power grid, such LP problems have to be solved very efficiently.

Intuitively, instead of sending the whole Eq.(2) to an LP solver, one would decompose the optimization problem first utilizing the fact that A is an M-matrix and thus is invertible. Let $\mathbf{e}_l$ be an $n \times 1$ vector of all 0's except the l'th component being 1. Let $\mathbf{c}_l \triangleq A^{-1}\mathbf{e}_l$, which can be computed by solving $A\mathbf{x} = \mathbf{e}_l$ using any power grid analysis algorithm. Because $A\mathbf{v} = \mathbf{i}$ and A is symmetric, we have $v_l = \mathbf{c}_l{}^T\mathbf{i}$. Then the optimization problem becomes

$$\text{Maximize } \mathbf{c}_l{}^T\mathbf{i} \text{ s.t. } 0 \le \mathbf{i} \le \mathbf{I}_L, U\mathbf{i} \le \mathbf{I}_G, \quad (3)$$

which is also an LP problem but with roughly half of the decision variables and constraints in comparison to Eq.(2).

The above intuitive idea was partially explored in [3]. As the optimization problem in Eq.(3) still involves a large number of decision variables and constraints for any practical power grid, it was proposed by [3] to compute an approximated $\mathbf{c}_l$ with only a few non-zero components. Then most decision variables and local constraints corresponding to the zero element in the approximated $\mathbf{c}_l$ can be dropped from Eq.(3) and thus it can be efficiently solved by any LP solver. However, this approach has two drawbacks. First, the number of non-zero elements depends on the accuracy of the approximation. If a higher level of accuracy is required, the approximated $\mathbf{c}_l$ will contain more non-zero components and the optimization problem in Eq.(3) is harder to solve. Second, as indicated by our preliminary experiments, the computation of the approximated $\mathbf{c}_l$ does consume a significant amount of running time in comparison to obtaining the exact $\mathbf{c}_l$ using efficient power grid analysis algorithms.

Therefore, it is of great interest to solve the optimization problem in Eq.(3) efficiently with the exact $\mathbf{c}_l$, such that the advances in power grid analysis can be leveraged. Clearly, to achieve such a goal, one must be able to exploit the structures in the problem. Our major contribution in this paper is an efficient dual approach to solve Eq.(3) exploiting its special structures. We exploit the special structure of the local constraints to simplify the dual problem Eq.(3) by eliminating most decision variables and constraints. We show the simplified dual problem is a convex problem with m decision variables and m constraints requiring the decision variables being non-negative. We propose to solve the convex problem by the cutting-plane method and to speed-up the evaluation of the objective function and its subgradient by exploiting the special structure of the global constraints. We design the DualVD algorithm to solve the *maxVD-LCC* problem efficiently combining a preconditioned conjugate gradient power grid analyzer based on [5] to obtain $\mathbf{c}_l$, and our dual approach to solve Eq.(3). The details are presented in the next section.

4. PROPOSED APPROACH

4.1 Simplified Dual Problem

Recall that m is much smaller than n. Most constraints in Eq.(3) are simply the bounds on the decision variables. To exploit such special structure in the primal problem, a typical approach is to investigate the corresponding dual problem. Let $\boldsymbol{\beta}$ be the vector of Lagrangian multipliers corresponding to the constraints $\mathbf{i} \le \mathbf{I}_L$ and $\boldsymbol{\gamma}$ be the vector of Lagrangian multipliers corresponding to the constraints $U\mathbf{i} \le \mathbf{I}_G$. The dual problem of Eq.(3) can be formulated as follows.

$$\text{Minimize } \mathbf{I}_L^T\boldsymbol{\beta} + \mathbf{I}_G^T\boldsymbol{\gamma} \text{ s.t. } \boldsymbol{\beta} + U^T\boldsymbol{\gamma} \ge \mathbf{c}_l, \boldsymbol{\beta} \ge 0, \boldsymbol{\gamma} \ge 0. \quad (4)$$

As Eq.(3) has a feasible solution $\mathbf{i} = 0$, the optimal value of Eq.(3) and Eq.(4) are the same due to the strong duality of LPs. Therefore, one can solve Eq.(4) in order to obtain the maximum voltage drop.

Let $\boldsymbol{\gamma} \ge 0$ be an arbitrary $m \times 1$ vector. For $1 \le j \le n$, denote the j'th column of U by $\mathbf{u}_j$ and the j'th component of $\mathbf{c}_l$ by $c_{l,j}$. Let the vector $\boldsymbol{\beta}^* = (\beta_1^*, \beta_2^*, \ldots, \beta_n^*)^T$ where

$$\beta_j^* \triangleq \max(0, c_{l,j} - \mathbf{u}_j^T\boldsymbol{\gamma}), \forall 1 \le j \le n.$$

Then obviously $(\boldsymbol{\beta}^*, \boldsymbol{\gamma})$ is a feasible solution of Eq.(4). Further more, for any feasible solution $(\boldsymbol{\beta}, \boldsymbol{\gamma})$ of Eq.(4), it must be true that $\boldsymbol{\beta} \ge \boldsymbol{\beta}^*$. Since $\mathbf{I}_L \ge 0$, it implies

$$\mathbf{I}_L^T\boldsymbol{\beta} + \mathbf{I}_G^T\boldsymbol{\gamma} \ge \mathbf{I}_L^T\boldsymbol{\beta}^* + \mathbf{I}_G^T\boldsymbol{\gamma},$$

which means that for Eq.(4), the objective value corresponding to a feasible solution $(\boldsymbol{\beta}, \boldsymbol{\gamma})$ is always no smaller than that of $(\boldsymbol{\beta}^*, \boldsymbol{\gamma})$. Denote the j'th component of $\mathbf{I}_L$ by $I_{L,j}$. We

propose to simplify Eq.(4) by eliminating $\boldsymbol{\beta}$ using $\boldsymbol{\beta}^*$ and a new optimization problem is thus formulated as follows,

$$\text{Minimize } D(\boldsymbol{\gamma}) \text{ s.t. } \boldsymbol{\gamma} \geq 0, \tag{5}$$

where the objective function $D(\boldsymbol{\gamma})$ is defined as

$$D(\boldsymbol{\gamma}) \triangleq \mathbf{I}_G^T \boldsymbol{\gamma} + \sum_{j=1}^{n} I_{L,j} \max(0, c_{l,j} - \mathbf{u}_j^T \boldsymbol{\gamma}).$$

According to the above discussions, the optimization problems in Eq.(3), (4), and (5) have the following property.

THEOREM 1. *The optimization problems in Eq.(3), (4), and (5) have optimal solutions and their optimal values are the same.*

Though the three optimization problems have the same optimal value, the benefit of the formulation in Eq.(5), in comparison to the other two, is that the number of decision variables and constraints is much smaller because of the fact that $m \ll n$, and that the constraints are extremely simple. The concern is about the objective function $D(\boldsymbol{\gamma})$, which seems to be much more complicated than those of the other two because of the max operations involved. However, as we shall prove in the next subsection that $D(\boldsymbol{\gamma})$ is convex with respect to $\boldsymbol{\gamma}$, Eq.(5) can be solved by efficient convex programming techniques.

4.2 Convexity of Simplified Dual Problem

Consider a particular $m \times 1$ vector $\boldsymbol{\gamma} \geq 0$. For $1 \leq j \leq n$, define

$$E_j(\boldsymbol{\gamma}) \triangleq \begin{cases} 1, & \text{if } c_{l,j} \geq \mathbf{u}_j^T \boldsymbol{\gamma}, \\ 0, & \text{otherwise.} \end{cases}$$

Then clearly

$$\max(0, c_{l,j} - \mathbf{u}_j^T \boldsymbol{\gamma}) = E_j(\boldsymbol{\gamma})(c_{l,j} - \mathbf{u}_j^T \boldsymbol{\gamma}). \tag{6}$$

Moreover, let $\boldsymbol{\gamma}'$ be an arbitrary $m \times 1$ non-negative vector. As $E_j(\boldsymbol{\gamma})$ is either 0 or 1, it must be true that,

$$\max(0, c_{l,j} - \mathbf{u}_j^T \boldsymbol{\gamma}') \geq E_j(\boldsymbol{\gamma})(c_{l,j} - \mathbf{u}_j^T \boldsymbol{\gamma}'). \tag{7}$$

Recall that $I_{L,j} \geq 0$. According to Eq.(6) and (7), we have

$$\begin{aligned} & D(\boldsymbol{\gamma}') - D(\boldsymbol{\gamma}) \\ = & \sum_{j=1}^{n} I_{L,j} \big(\max(0, c_{l,j} - \mathbf{u}_j^T \boldsymbol{\gamma}') - \max(0, c_{l,j} - \mathbf{u}_j^T \boldsymbol{\gamma}) \big) + \mathbf{I}_G^T (\boldsymbol{\gamma}' - \boldsymbol{\gamma}) \\ \geq & \sum_{j=1}^{n} I_{L,j} \big(E_j(\boldsymbol{\gamma})(c_{l,j} - \mathbf{u}_j^T \boldsymbol{\gamma}') - E_j(\boldsymbol{\gamma})(c_{l,j} - \mathbf{u}_j^T \boldsymbol{\gamma}) \big) + \mathbf{I}_G^T (\boldsymbol{\gamma}' - \boldsymbol{\gamma}) \\ = & (\mathbf{I}_G - \sum_{j=1}^{n} I_{L,j} E_j(\boldsymbol{\gamma}) \mathbf{u}_j)^T (\boldsymbol{\gamma}' - \boldsymbol{\gamma}) \end{aligned}$$

Therefore, the following lemma must hold.

LEMMA 1. *$D(\boldsymbol{\gamma})$ is a convex function of $\boldsymbol{\gamma}$. Further more, let $g(\boldsymbol{\gamma})$ be an $m \times 1$ vector defined as*

$$g(\boldsymbol{\gamma}) \triangleq \mathbf{I}_G - \sum_{j=1}^{n} I_{L,j} E_j(\boldsymbol{\gamma}) \mathbf{u}_j.$$

Then $g(\boldsymbol{\gamma})$ is a subgradient of $D(\boldsymbol{\gamma})$.

4.3 Solving Simplified Dual Problem

Lemma 1 implies that the optimization problem in Eq.(5) is a convex programming problem. Since the subgradient of the objective function can be computed accordingly, we propose to solve Eq.(5) by Kelley's cutting-plane method [6].

To apply the cutting-place method, we shall first limit the search of the optimal solution of Eq.(5) to a bounded set. According to Theorem 1, Eq.(5) has an optimal solution. Assume that $\boldsymbol{\gamma}^* = (\gamma_1^*, \gamma_2^*, \ldots, \gamma_m^*)^T$ is such an optimal solution. Define $c_{\max} \triangleq \max(0, c_{l,1}, c_{l,2}, \ldots, c_{l,n})$. If there exists some k such that $1 \leq k \leq m$ and $\gamma_k^* > c_{\max}$, then consider the vector $\boldsymbol{\gamma}^-$ that is obtained from $\boldsymbol{\gamma}^*$ by replacing γ_k^* with $c_{\max}$. Obviously, $\boldsymbol{\gamma}^* \geq \boldsymbol{\gamma}^- \geq 0$. Moreover, since each component of $\mathbf{u}_j$ is either 0 or 1, we have

$$\max(0, c_{l,j} - \mathbf{u}_j^T \boldsymbol{\gamma}^*) = \max(0, c_{l,j} - \mathbf{u}_j^T \boldsymbol{\gamma}^-), \forall 1 \leq j \leq n.$$

Recall that $\mathbf{I}_G \geq 0$. We thus have $D(\boldsymbol{\gamma}^*) \geq D(\boldsymbol{\gamma}^-)$, and then it must be true that $D(\boldsymbol{\gamma}^*) = D(\boldsymbol{\gamma}^-)$ since $\boldsymbol{\gamma}^*$ is an optimal solution. Therefore, one can solve Eq.(5) by only examining the feasible solutions in a bounded set as stated in the following lemma.

LEMMA 2. *Let $\boldsymbol{\gamma}_{max}$ be an $m \times 1$ vector with all components being c_{max}. Then there is an optimal solution $\boldsymbol{\gamma}^*$ of Eq.(5) satisfying that $0 \leq \boldsymbol{\gamma}^* \leq \boldsymbol{\gamma}_{max}$.*

We then present the cutting-plane method to solve Eq.(5). For a finite set $\mathcal{S}$ of $m \times 1$ non-negative vectors, define the optimization problem CP($\mathcal{S}$) as follows,

$$\begin{aligned} \text{Minimize } & Y \quad \text{s.t.} \\ & Y \geq D(\boldsymbol{\gamma}') + (\boldsymbol{\gamma} - \boldsymbol{\gamma}')^T g(\boldsymbol{\gamma}'), \forall \boldsymbol{\gamma}' \in \mathcal{S}, \\ & 0 \leq \boldsymbol{\gamma} \leq \boldsymbol{\gamma}_{\max}, \end{aligned} \tag{8}$$

where the decision variables are Y and $\boldsymbol{\gamma}$. It is straightforward that CP($\mathcal{S}$) is an LP problem and has an optimal solution. Let $Y_{\mathcal{S}}$ and $\boldsymbol{\gamma}_{\mathcal{S}}$ be such optimal solution.

The cutting-plane method solves Eq.(5) by computing a series of sets $\mathcal{S}_0 \subset \mathcal{S}_1 \subset \mathcal{S}_2 \subset \cdots$ iteratively. Initially, $\mathcal{S}_0 = \{\boldsymbol{\gamma}_0\}$, where $\boldsymbol{\gamma}_0$ can be any feasible $m \times 1$ vector satisfying $0 \leq \boldsymbol{\gamma}_0 \leq \boldsymbol{\gamma}_{\max}$. We use $\boldsymbol{\gamma}_0 = 0$ in our implementation. For $p \geq 0$, the set $\mathcal{S}_{p+1}$ is computed as $\mathcal{S}_p \cup \{\boldsymbol{\gamma}_{\mathcal{S}_p}\}$ after solving the the LP problem CP($\mathcal{S}_p$). The iterations can be terminated when $\min_{\boldsymbol{\gamma} \in \mathcal{S}_{p+1}} D(\boldsymbol{\gamma})$ is close enough to the optimal value of Eq.(5), where the gap between these two can be computed according to the following lemma implied by the cutting-plane method.

LEMMA 3. *Let $\boldsymbol{\gamma}^*$ be an optimal solution of Eq.(5). Then*

$$Y_{\mathcal{S}_p} \leq D(\boldsymbol{\gamma}^*) \leq \min_{\boldsymbol{\gamma} \in \mathcal{S}_{p+1}} D(\boldsymbol{\gamma}), \forall p \geq 0.$$

The convergence of iterations is guaranteed by the following lemma.

LEMMA 4. *There exists a p^* such that $\boldsymbol{\gamma}^* = \boldsymbol{\gamma}_{\mathcal{S}_{p^*}}$.*

Due to the lack of space, we omit the details of the proof for Lemma 4, which is a special property of the cutting-plane method when applying to Eq.(5). We only mention that such property holds because of the fact that $D(\boldsymbol{\gamma})$ is the summation of a linear function of $\boldsymbol{\gamma}$ and a finite number of the maximum of 0 and a linear function of $\boldsymbol{\gamma}$.

In our implementation, a user-specified error-tolerance δ_{cp} is used to terminate the iterations when

$$\min_{\boldsymbol{\gamma} \in \mathcal{S}_{p+1}} D(\boldsymbol{\gamma}) - Y_{\mathcal{S}_p} \leq \delta_{\text{cp}}. \tag{9}$$

The vector $\boldsymbol{\gamma}_{\text{cp}}$, defined as $\boldsymbol{\gamma}_{\text{cp}} \triangleq \arg\min_{\boldsymbol{\gamma} \in \mathcal{S}_{p+1}} D(\boldsymbol{\gamma})$, is reported as the optimal solution of Eq.(5). Based on Lemma 3, the gap between $D(\boldsymbol{\gamma}_{\text{cp}})$ and $D(\boldsymbol{\gamma}^*)$ is stated in the following lemma.

LEMMA 5. *$D(\boldsymbol{\gamma}_{cp}) - \delta_{cp} \leq D(\boldsymbol{\gamma}^*) \leq D(\boldsymbol{\gamma}_{cp})$.*

Note that although multiple LP problems should be solved in the cutting-plane method for Eq.(5), since these problems are small in size, it takes much less running time than solving a single – but large LP problem as either Eq.(3) or Eq.(4)

4.4 Fast Evaluation of Objective Function and Its Subgradient

For the cutting-plane method introduced in the previous subsection, although the LP problems $\mathrm{CP}(\mathcal{S}_p)$ can be solved efficiently due to their small sizes, they have to be formulated first, and the computational complexity to formulate them is usually overlooked. To be more specific, for $p \geq 1$, to formulate $\mathrm{CP}(\mathcal{S}_p)$ requires to compute $D(\boldsymbol{\gamma}_{\mathcal{S}_{p-1}})$ and $g(\boldsymbol{\gamma}_{\mathcal{S}_{p-1}})$. Apparently, a straight-forward approach takes $O(nm)$ time to compute D and g according to their definitions. Therefore, as the size of the problem $\mathrm{CP}(\mathcal{S}_p)$ is much less than n, it can be more expensive to formulate $\mathrm{CP}(\mathcal{S}_p)$ than to solve it, which was also confirmed by our preliminary experiments. In this subsection, we will present a novel pre-processing step exploiting the special structure of the global constraints, i.e. the matrix U, such that D and g can be evaluated with a time complexity that is usually much smaller than $O(nm)$.

A general observation of the matrix U is that it has many duplicated columns, i.e., the $\mathbf{u}_j$'s only take a limited number of distinct values. This observation is due to the fact that when circuit designers specify the global constraints, they tend to specify the constraints according to the circuit hierarchy. Therefore, nodes in the same module would be included in the same set of global constraints, corresponding to the identical columns in U.

Let $\mathcal{U}$ be the set of the distinct columns of U, i.e., $\mathcal{U} \triangleq \bigcup_{j=1}^{n}\{\mathbf{u}_j\}$. Denote the elements of $\mathcal{U}$ as $\overline{\mathbf{u}}_1, \overline{\mathbf{u}}_2, \ldots$, and $\overline{\mathbf{u}}_{|\mathcal{U}|}$. The duplicated columns of U can be identified by introducing $|\mathcal{U}|$ index sets defined as $\mathcal{J}_k \triangleq \{j : \mathbf{u}_j = \overline{\mathbf{u}}_k\}, \forall 1 \leq k \leq |\mathcal{U}|$. Note that the sets $\mathcal{J}_k$ are a partition of $\{1, 2, \ldots, n\}$,

$$\mathcal{J}_k \cap \mathcal{J}_{k'} = \emptyset, \forall 1 \leq k \neq k' \leq |\mathcal{U}|, \text{ and } \bigcup_{k=1}^{|\mathcal{U}|} \mathcal{J}_k = \{1, 2, \ldots, n\}.$$

With a given $\mathbf{c}_l$, the elements in each set $\mathcal{J}_k$ can be sorted according to the corresponding components of $\mathbf{c}_l$. Therefore, without loss of generality, we can assume that

$$c_{l,j} \leq c_{l,j'}, \forall (j, j' \in \mathcal{J}_k) \wedge (j \leq j'), 1 \leq k \leq |\mathcal{U}|. \quad (10)$$

For a particular $\boldsymbol{\gamma}$ and $1 \leq k \leq |\mathcal{U}|$, Eq.(10) implies that $c_{l,j} - \mathbf{u}_j^T\boldsymbol{\gamma} = c_{l,j} - \overline{\mathbf{u}}_k^T\boldsymbol{\gamma}$ is non-decreasing for $j \in \mathcal{J}_k$. So, if the index j_k is defined as

$$j_k \triangleq \min\{j : c_{l,j} - \overline{\mathbf{u}}_k^T\boldsymbol{\gamma} \geq 0, j \in \mathcal{J}_k\}, \quad (11)$$

then

$$\begin{aligned} c_{l,j} - \mathbf{u}_j^T\boldsymbol{\gamma} &\geq 0, \qquad \forall (j \in \mathcal{J}_k) \wedge (j \geq j_k), \\ c_{l,j} - \mathbf{u}_j^T\boldsymbol{\gamma} &< 0, \qquad \forall (j \in \mathcal{J}_k) \wedge (j < j_k), \end{aligned}$$

which can be used to simplify the computation of $D(\boldsymbol{\gamma})$ as

$$\begin{aligned} D(\boldsymbol{\gamma}) &= \mathbf{I}_G^T\boldsymbol{\gamma} + \sum_{k=1}^{|\mathcal{U}|} \sum_{j \in \mathcal{J}_k} I_{L,j} \max(0, c_{l,j} - \mathbf{u}_j^T\boldsymbol{\gamma}) \\ &= \mathbf{I}_G^T\boldsymbol{\gamma} + \sum_{k=1}^{|\mathcal{U}|} \sum_{(j \in \mathcal{J}_k) \wedge (j \geq j_k)} I_{L,j}(c_{l,j} - \overline{\mathbf{u}}_k^T\boldsymbol{\gamma}) \\ &= \mathbf{I}_G^T\boldsymbol{\gamma} + \sum_{k=1}^{|\mathcal{U}|} \sum_{(j \in \mathcal{J}_k) \wedge (j \geq j_k)} I_{L,j} c_{l,j} - \sum_{k=1}^{|\mathcal{U}|} \overline{\mathbf{u}}_k^T\boldsymbol{\gamma} \sum_{(j \in \mathcal{J}_k) \wedge (j \geq j_k)} I_{L,j} \quad (12) \end{aligned}$$

Define $\mathrm{Sum}_{\mathrm{ILC}}(k, j)$ and $\mathrm{Sum}_{\mathrm{IL}}(k, j)$ as

$$\mathrm{Sum}_{\mathrm{ILC}}(k, j) \triangleq \sum_{(j' \in \mathcal{J}_k) \wedge (j' \geq j)} I_{L,j'} c_{l,j'}, \forall j \in \mathcal{J}_k, 1 \leq k \leq |\mathcal{U}|,$$

$$\mathrm{Sum}_{\mathrm{IL}}(k, j) \triangleq \sum_{(j' \in \mathcal{J}_k) \wedge (j' \geq j)} I_{L,j'}, \forall j \in \mathcal{J}_k, 1 \leq k \leq |\mathcal{U}|.$$

Eq.(12) becomes

$$D(\boldsymbol{\gamma}) = \mathbf{I}_G^T\boldsymbol{\gamma} + \sum_{k=1}^{|\mathcal{U}|} \big(\mathrm{Sum}_{\mathrm{ILC}}(k, j_k) - \mathrm{Sum}_{\mathrm{IL}}(k, j_k)\overline{\mathbf{u}}_k^T\boldsymbol{\gamma}\big). \quad (13)$$

Similarly, we have

$$g(\boldsymbol{\gamma}) = \mathbf{I}_G - \sum_{k=1}^{|\mathcal{U}|} \mathrm{Sum}_{\mathrm{IL}}(k, j_k)\overline{\mathbf{u}}_k. \quad (14)$$

Based on the above discussion, $D(\boldsymbol{\gamma})$ and $g(\boldsymbol{\gamma})$ are computed in three steps as follows. First, when U is given in Problem 1, $\mathcal{U}$ and $\mathcal{J}_k$, $\forall 1 \leq k \leq |\mathcal{U}|$, are computed according to their definitions. Second, when $\mathbf{c}_l$ is obtained, $\mathcal{J}_k$, $\forall 1 \leq k \leq |\mathcal{U}|$, are sorted to make the assumption in Eq.(10) valid, and then $\mathrm{Sum}_{\mathrm{ILC}}(k, j)$ and $\mathrm{Sum}_{\mathrm{IL}}(k, j)$, $\forall j \in \mathcal{J}_k$ and $1 \leq k \leq |\mathcal{U}|$, are computed. Third, with a given $\boldsymbol{\gamma}$, the indices j_k, $\forall 1 \leq k \leq |\mathcal{U}|$, are obtained according to their definition by $|\mathcal{U}|$ binary searches since $\mathcal{J}_k$ is ordered according to Eq.(10), and then $D(\boldsymbol{\gamma})$ and $g(\boldsymbol{\gamma})$ are calculated according to Eq.(13) and (14). Therefore, for each iteration in the cutting-plane method, only the third step is necessary and the first two steps can be pre-computed. The time complexity of the third step is $O(m|\mathcal{U}| + \sum_{k=1}^{|\mathcal{U}|} \log |\mathcal{J}_k|)$. Recall that $\sum_{k=1}^{|\mathcal{U}|} |\mathcal{J}_k| = n$ as $\mathcal{J}_k$ is a partition of $\{1, 2, \ldots, n\}$. Since,

$$\sum_{k=1}^{|\mathcal{U}|} \log |\mathcal{J}_k| = \log \prod_{k=1}^{|\mathcal{U}|} |\mathcal{J}_k| \leq \log \Big(\frac{\sum_{k=1}^{|\mathcal{U}|} |\mathcal{J}_k|}{|\mathcal{U}|}\Big)^{|\mathcal{U}|} = |\mathcal{U}| \log \frac{n}{|\mathcal{U}|},$$

the time complexity of the third step simplifies to $O\big(|\mathcal{U}|(m + \log \frac{n}{|\mathcal{U}|})\big)$. Obviously, $|\mathcal{U}| \leq n$. Because $|\mathcal{U}| \log \frac{n}{|\mathcal{U}|}$ increases monotonically for $|\mathcal{U}| \leq n$, the time complexity of the third step is much smaller than the time complexity $O(nm)$ of the straight-forward computation when $|\mathcal{U}| \ll n$, or no worse than $O(nm)$ if $|\mathcal{U}|$ approaches n. In summary, the time and space complexity associated with the three steps are stated in the following lemma.

LEMMA 6. *The time complexity of the first step is $O(nm \log |\mathcal{U}|)$, of the second step is $O(n \log n)$, and of the third step is $O\big(|\mathcal{U}|(m + \log \frac{n}{|\mathcal{U}|})\big)$. Extra $O(n)$ storages are required to store the pre-computed values.*

4.5 Computing $\mathbf{c}_l$

The dual approach proposed in the previous subsections assumes that $\mathbf{c}_l$ is known. In this subsection, we will present a method to compute $\mathbf{c}_l$, $\forall 1 \leq l \leq n$.

Recall that $\mathbf{c}_l = A^{-1}\mathbf{e}_l$, and thus can be computed by solving $A\mathbf{x} = \mathbf{e}_l$. Therefore, to obtain all $\mathbf{c}_l$, $\forall 1 \leq l \leq n$, we need to solve a system of linear equations for n times with the same left-hand-side (LHS) matrix A but different right-hand-side (RHS) vectors. As A has the good properties of being a symmetric M-matrix, we propose to apply the preconditioned conjugate gradient (PCG) method [7, 8], which is an iterative method to solve a system of linear equations with symmetric LHS matrix, due to its fast convergence with a proper preconditioner and the straight-forward implementation. For the PCG method, the preconditioner M is a matrix with the same size as A such that $M \approx A$ and $M\mathbf{x} = \mathbf{r}$

can be easily solved for $\mathbf{x}$ given some RHS vector $\mathbf{r}$.

To obtain the preconditioner M, we apply the stochastic preconditioning technique proposed in [5]. This preconditioning technique was motivated by a random-walk power grid analyzer [9], and had been shown to have high quality, especially for large problem sizes, in comparison to conventional deterministic preconditioning techniques. Though it requires more running time to compute M according to [5] in comparison to conventional techniques, it is not a concern in our setting since the preconditioning time will be amortized when the maximum voltage drops are computed for all n nodes. In the stochastic preconditioning technique, we perform random walks on the power grid to generate a sparse lower triangular matrix L and a diagonal matrix Λ. The preconditioner $M \triangleq L^T \Lambda L$ satisfies that $M \approx A$, and the linear equations $M\mathbf{x} = \mathbf{r}$ can be solved by a backward substitution followed by a forward substitution.

Practically, the PCG method can be terminated when the norm of the residual is small enough. As the residual will not be 0, the computed $\mathbf{c}_l$ is different from the exact value and thus the optimal value of Eq.(5) based on the computed $\mathbf{c}_l$ is different from the exact maximum voltage drop. Therefore, it is necessary to analyze the error of the optimal value of Eq.(5) due to the non-zero residual such that we can convert from a user-specified error-tolerance of the optimal value of Eq.(5) to a proper termination condition of the PCG method. Details follow.

Let $\bar{\mathbf{c}}_l$ be the exact solution of $A\mathbf{x} = \mathbf{e}_l$ and $\mathbf{c}_l$ be the solution computed by the PCG method. According to Theorem 1, we can consider the optimization problem in Eq.(3) instead of Eq.(5) as their optimal values are the same. For the optimization problem in Eq.(3), let $\bar{\mathbf{i}}$ be an optimal solution and $\overline{v_l}$ be the optimal value for the exact $\bar{\mathbf{c}}_l$, and let $\mathbf{i}^*$ be an optimal solution and v_l^* be the optimal value for the computed $\mathbf{c}_l$. Let the residual vector $\mathbf{r} \triangleq \mathbf{e}_l - A\mathbf{c}_l$. It is obvious that

$$A^{-1}\mathbf{r} = \bar{\mathbf{c}}_l - \mathbf{c}_l, \overline{v_l} = \bar{\mathbf{c}}_l^T \bar{\mathbf{i}} \geq \bar{\mathbf{c}}_l^T \mathbf{i}^*, v_l^* = \mathbf{c}_l^T \mathbf{i}^* \geq \mathbf{c}_l^T \bar{\mathbf{i}}.$$

Therefore,

$$\overline{v_l} \geq \bar{\mathbf{c}}_l^T \mathbf{i}^* = \mathbf{c}_l^T \mathbf{i}^* + \mathbf{r}^T A^{-1} \mathbf{i}^* = v_l^* + \mathbf{r}^T A^{-1} \mathbf{i}^*, \quad (15)$$

and

$$\overline{v_l} = \mathbf{c}_l^T \bar{\mathbf{i}} + \mathbf{r}^T A^{-1} \bar{\mathbf{i}} \leq \mathbf{c}_l^T \mathbf{i}^* + \mathbf{r}^T A^{-1} \bar{\mathbf{i}} = v_l^* + \mathbf{r}^T A^{-1} \bar{\mathbf{i}}. \quad (16)$$

On the other hand, since A is an M-matrix, every element of A^{-1} is non-negative. So all the components of $A^{-1}\bar{\mathbf{i}}$ and $A^{-1}\mathbf{i}^*$ are non-negative since $\bar{\mathbf{i}} \geq 0$ and $\mathbf{i}^* \geq 0$. Moreover, let Δ be the maximum component of the solution of the linear equations $A\mathbf{x} = \mathbf{I}_L$. All the components of $A^{-1}\bar{\mathbf{i}}$ and $A^{-1}\mathbf{i}^*$ should be no larger than Δ. Let r_i be the i'th component of $\mathbf{r}$. According to Eq.(15) and (16), we have

$$v_l^* + \Delta \sum_{i=1}^{n} \min(r_i, 0) \leq \overline{v_l} \leq v_l^* + \Delta \sum_{i=1}^{n} \max(r_i, 0)$$

Therefore, if a user-specified error-tolerance δ_{pcg} is used to terminate the PCG method when

$$||\mathbf{r}||_1 \triangleq \sum_{i=1}^{n} |r_i| \leq \frac{\delta_{\text{pcg}}}{\Delta}, \quad (17)$$

then the following lemma holds.

LEMMA 7. *Define* $v_l^+ \triangleq \Delta \sum_{i=1}^{n} \max(r_i, 0)$ *as a correction of* v_l^* *based on the residual. We have*

$$v_l^* + v_l^+ - \delta_{pcg} \leq \overline{v_l} \leq v_l^* + v_l^+.$$

Algorithm DualVD

Inputs

A, $\mathbf{I}_L$, $\mathbf{I}_G$, U: as specified in Problem 1.

δ_{pcg}, δ_{cp}: user-specified error-tolerances.

Outputs

Maximum voltage drop at each node.

1 Compute the preconditioner M according to [5]
2 Solve $A\mathbf{x} = \mathbf{I}_L$ to obtain Δ
3 Compute $\mathcal{U}$ and $\mathcal{J}_k$, $\forall 1 \leq k \leq |\mathcal{U}|$
4 **For** $l = 1$ to n
5 Apply PCG method to compute $\mathbf{c}_l$ using the termination condition in Eq.(17)
6 Sort $\mathcal{J}_k$, $\forall 1 \leq k \leq |\mathcal{U}|$, according to Eq.(10)
7 Compute $\text{Sum}_{\text{ILC}}(k, j)$ and $\text{Sum}_{\text{IL}}(k, j)$, $\forall j \in \mathcal{J}_k$ and $1 \leq k \leq |\mathcal{U}|$
8 Apply cutting-plane method to solve Eq.(5) using the termination condition in Eq.(9)
9 Report the maximum voltage drop at node l to be $D(\boldsymbol{\gamma}_{\text{cp}}) + v_l^+$

Figure 1: The DualVD algorithm.

4.6 The DualVD Algorithm

Combining the PCG method and the proposed dual approach, we design the DualVD algorithm to solve the *maxVDLCC* problem as shown in Fig. 1. In this algorithm, the preprocessing steps on line 1 to 3 are used to generate necessary data for the PCG method and the cutting-plane method. Then the maximum voltage drop at each node is computed in the body of the loop on line 4. Note that on line 9, we do not report the optimal value of Eq.(5) as the maximum voltage drop at a node because it could be smaller than the exact maximum voltage drop due to the error introduced by the PCG method. We report a conservative voltage drop within the user-specified error-tolerances based on the optimal value and a correction from the residual of the PCG method.

The correctness of the DualVD algorithm is stated in the following theorem, which is implied by Lemma 5, Lemma 7, and the fact that $D(\boldsymbol{\gamma}^*) = v_l^*$ in these two lemma.

THEOREM 2. *For every* $1 \leq l \leq n$,

$$D(\boldsymbol{\gamma}_{cp}) + v_l^+ - \delta_{cp} - \delta_{pcg} \leq \overline{v_l} \leq D(\boldsymbol{\gamma}_{cp}) + v_l^+.$$

5. EXPERIMENTAL RESULTS

The DualVD algorithm is implemented in C++ and compiled with GCC version 4.2. The LP problems $\text{CP}(\mathcal{S})$ in the cutting-plane method are solved by MOSEK [10], a general optimizing engine. For comparison, we replace the proposed dual approach in the DualVD algorithm by MOSEK to solve Eq.(3) directly, and call this algorithm DirectVD hereafter. As MOSEK allows to choose between the simplex method and the interior point method to solve LP problems, we experiment with both options but only report the results using the simplex method as it is slightly faster. All the experiments are performed on a 64-bit Linux machine with 2.4GHz Intel Q6600 processor and 4GB memory. Note that although the processor has multiple cores, only one core is used for experiments.

We generate 7 power grid benchmarks randomly using a setting of 4 metal layers, 1.2V VDD, and various C4 bumps/ chip sizes/power consumptions. Both error tolerances δ_{cp} and δ_{pcg} are set to 10^{-4}, i.e. 0.1mV. Two global constraints settings are experimented: one with 4 global constraints and the other with 10. Under such experimental settings, we observe that the PCG method and the cutting-plane method converge in less than 15 and 40 iterations respectively.

Table 1: Runtime Comparison of DualVD and DirectVD

Power Grid Benchmarks	4 Global Constraints						10 Global Constraints					
	DirectVD		DualVD		Speedup		DirectVD		DualVD		Speedup	
Nodes	LP	Total	LP	Total	LP	Total	LP	Total	LP	Total	LP	Total
5875	2.51 m	2.99 m	9.76 s	38.63 s	15.4	4.6	2.66 m	3.15 m	22.53 s	51.45 s	7.1	3.7
22939	1.03 h	1.17 h	1.77 m	10.15 m	34.9	6.9	1.07 h	1.20 h	2.37 m	10.72 m	26.9	6.7
35668	2.99 h	3.37 h	3.95 m	26.36 m	45.4	7.7	3.14 h	3.51 h	5.12 m	27.39 m	36.8	7.7
51195	6.19 h	6.96 h	8.32 m	55.59 m	44.7	7.5	6.49 h	7.27 h	9.11 m	56.19 m	42.8	7.8
90643	17.66 h	20.24 h	25.36 m	3.01 h	41.8	6.7	16.64 h	19.22 h	26.86 m	3.02 h	37.2	6.4
141283	1.21 d	1.48 d	1.06 h	7.41 h	27.4	4.8	1.43 d	1.70 d	1.06 h	7.38 h	32.6	5.5
562363	12.52 d	17.06 d	19.11 h	5.36 d	15.7	3.2	14.71 d	19.34 d	17.39 h	5.33 d	20.3	3.6

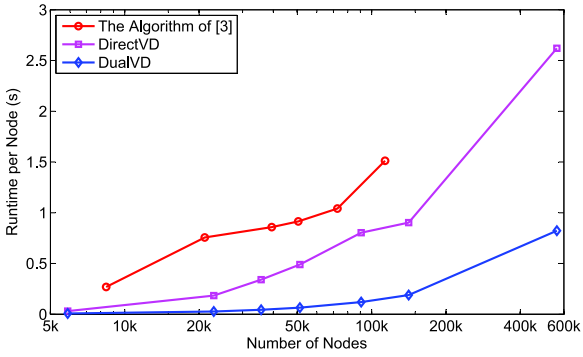

Figure 2: Comparison of runtime per node

The runtime comparison of the DualVD and the DirectVD algorithm is presented in Table 1. The voltage drops computed by the two algorithms are the same and thus are excluded from the table. The runtime can be generally partitioned into two parts: the runtime to compute $\mathbf{c}_l$ and the runtime to solve the LP problem Eq.(3) (or more precisely Eq.(5) for the DualVD algorithm). Both algorithms have the same runtime for the former as they share the same code to compute $\mathbf{c}_l$. Since the latter is the major contribution of the paper, we report the runtime for the latter under the columns "LP", e.g. for the DualVD algorithm, the runtime in column 4 and 10. Moreover, we report the total runtime under the columns "Total". The time units are abbreviated such that "s", "m", "h" and "d" denote "seconds", "minutes","hours", and "days" respectively. It can be seen that the proposed dual approach drastically reduces the runtime to solve the LP problem such that the DualVD algorithm can achieve significant speedup over the DirectVD algorithm. Note that for the test cases which take more than 10 hours, the reported runtime is an estimation from the runtime of 1000 nodes chosen randomly.

Our experimental results show the random-walk based preconditioner computation is fairly efficient, taking runtime ranging from a few seconds to a few minutes as the power grids become larger. However, it can be implied from Table 1 that to compute $\mathbf{c}_l$ by the PCG method may require more than 80% of runtime for large benchmarks using the DualVD algorithm. We view such observation as a chance to further speedup the DualVD algorithm by leveraging more advanced power grid analysis techniques. In addition, according to Eq.(17) and Theorem 2, the PCG runtime can be reduced by using a larger δ_{pcg}, which makes a trade-off between solution quality and runtime.

We compare our algorithm with the previous work [3] in Fig. 2 for the setting of 4 global constraints. Note that this is not a fair comparison since the source code and the benchmarks in [3] are not publicly available. We are able to obtain the type of processor (AMD Opteron 2218) in the linux machine used in [3] from the authors. We scale their runtime with 5mV accuracy based on SPEC CPU2006 results (`http://www.spec.org/`) to obtain their per node runtime. It is clear that our DualVD algorithm can achieve much higher accuracy (since $\delta_{cp} + \delta_{pcg} = 0.2\text{mV} \ll 5\text{mV}$) with much less runtime. Moreover, our DualVD algorithm exhibits a better scaling trend for large benchmarks as the runtime per node increases slowly as the number of nodes increases.

In summary, our DualVD algorithm accelerates the solution of the LP problems, reduces the runtime per node significantly, and makes vectorless verification of large power grids practical.

6. CONCLUSION

In this paper, we presented an efficient dual approach to solve the LP problem associated with the vectorless power grid verification problem. Our overall vectorless power grid verification algorithm DualVD combined the proposed dual approach with a random-walk based PCG power grid analyzer. Experimental results confirmed the efficiency of the proposed algorithm.

We expect to achieve additional speedup by leveraging more advanced power grid analysis techniques, and by parallel processing due to the fact that each node can be verified independently. It is also interesting to study whether the proposed algorithm can be extended to handle inductance in power grids [11].

7. REFERENCES

[1] D. Kouroussis and F. N. Najm. A static pattern-independent technique for power grid voltage integrity verification. In *DAC*, pages 99–104, 2003.

[2] I. A. Ferzli, F. N. Najm, and L. Kruse. A geometric approach for early power grid verification using current constraints. In *ICCAD*, pages 40–47, 2007.

[3] N. H. Abdul Ghani and F. N. Najm. Fast vectorless power grid verification using an approximate inverse technique. In *DAC*, pages 184–189, 2009.

[4] H. Qian, S. R. Nassif, and S. S. Sapatnekar. Early-stage power grid analysis for uncertain working modes. In *ISPD*, pages 132–137, 2004.

[5] H. Qian and S. S. Sapatnekar. Stochastic preconditioning for diagonally dominant matrices. *SIAM Journal on Scientific Computing*, 30(3):1178–1204, Mar. 2008.

[6] J. E. Kelley. The cutting-plane method for solving convex programs. *J. Soc. Indust. and Appl. Math.*, 8(4):703–712, Dec. 1960.

[7] G. H. Golub and C. F. Van Loan . *Matrix Computations*, 3rd ed.. The Johns Hopkins University Press, 1996.

[8] T.-H. Chen and C. C.-P. Chen. Efficient large-scale power grid analysis based on preconditioned Krylov-subspace iterative methods. In *DAC*, pages 559–562, 2001.

[9] H. Qian, S. R. Nassif, and S. S. Sapatnekar. Power grid analysis using random walks. *IEEE TCAD*, 24(8):1204–1224, Aug. 2005.

[10] *The MOSEK Optimization Software*. `http://www.mosek.com`.

[11] N. H. Abdul Ghani and F. N. Najm. Handling inductance in early power grid verification. In *ICCAD*, pages 127–134, 2006.