Incremental Analysis of Power Grids using Backward Random Walks

Baktash Boghrati and Sachin S. Sapatnekar, University of Minnesota, Minneapolis, MN

Power grid design and analysis is a critical part of modern VLSI chip design and demands tools for accurate modeling and efficient analysis. The process of power grid design is inherently iterative, during which numerous small changes are made to an initial design, either to enhance the design or to fix design constraint violations. Due to the large sizes of power grids in modern chips, updating the solution for these perturbations can be a computationally intensive task. In this work, we first introduce an accurate modeling methodology for power grids that, contrary to conventional models, can result in asymmetrical equations. Next, we propose an efficient and accurate incremental solver that utilizes the backward random walks to identify the region of influence of the perturbation. The solution of the network is then updated for this significantly smaller region only. The proposed algorithm is capable of handling both symmetrical and asymmetrical power grid equations. Moreover, it can handle consecutive perturbations without any degradation in the quality of the solution. Experimental results show speedups of up to 13$\times$ for our incremental solver, as compared to a full re-solve of the power grid.

Categories and Subject Descriptors: B.8.2 [**Performance and Reliability**]: Performance Analysis and Design Aids

General Terms: Digital Circuits, Architectures, Reliability, Low Power

Additional Key Words and Phrases: Random walks, power grid, incremental analysis.

ACM Reference Format:
Boghrati, B. and Sapatnekar, S. S. 2013. Incremental Backward Random Walk Solver for Various Power Grid Models. ACM Trans. Des. Autom. Electron. Syst. V, N, Article A (January YYYY), 28 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

The problem of verifying the integrity of a power grid is a critical step in the design of integrated circuits. In the steady state (similar equations can be used for transient analysis with time-stepping), this verification corresponds to the solution of a set of equations to determine the voltages in the grid, and ensuring that they are within a certain permissible range. This system of equations can be written as:

$$G\mathbf{V} = \mathbf{E}, \tag{1}$$

where $G \in \Re^{N\times N}$ is the left hand-side (LHS) matrix, modeling the conductances, $\mathbf{V} \in \Re^{N}$ is the vector of unknown node voltages, and $\mathbf{E} \in \Re^{N}$ is the right hand side (RHS) vector, modeling the current loads. Matrix G is sparse and diagonally dominant ($\sum_{i\neq j} |g_{ij}| \leq g_{ii}, \forall i$), and all off-diagonals of G are less than or equal to zero. In fact, several problems in VLSI design and in other fields involve the solution of a system of similar linear equations where the left hand side has the form of a diagonally dominant matrix with positive diagonal and nonpositive off-diagonal entries, such as thermal analysis under the resistive-electrical duality, and quadratic placement. Mainstream methods for solving such systems include direct methods such as LU/Cholesky factorization and iterative methods. Lately, there has been an upsurge of interest in the use of random walk-based solvers for solving systems with diagonally dominant LHS matrices, and these solvers are competitive with conventional solvers [Qian et al. 2003; Qian and Sapatnekar 2005; Guo et al. 2004; Miyakawa et al. 2011; Wang 2012].

This paper addresses a specific problem in the area of steady-state power grid analysis where an incremental change is made in an already-solved network and the new solution is to be determined. Power grid design is a highly iterative process in which the designer makes small perturbations to a complete initial design to fine-tune the design or fix violations in the noise specifications. Examples of perturbations to a power network include changes to the wire conductances (e.g., when the length or thickness of the wires change), power pad placement, or current loads. For a small perturbation the solution of the perturbed system is *close* to the initial solution, meaning that the change in the solution of most of the nodes of the network is insignificant, and the small subset of nodes that are physically or electrically close to the perturbation are most likely to be affected [Singh and Sapatnekar 2004; Chiprout 2004]. Examples of small perturbations are: change in power grid structure locally, change in current sources, and change in pad positions.

The notion of closeness of the solution of the perturbed system to the initial solution suggests that for finding the perturbed solution, solving for the entire system from scratch is quite wasteful. To efficiently perform this task, one should leverage the property of "closeness" and utilize the unperturbed solution. However, this is not a trivial task since it is unclear *a priori* which nodes change significantly and which do not. One possible method would be to partition the network, create macromodels for each partition, and solve the problem hierarchically, as in [Zhao et al. 2000]. However, this may require a large number of partitions, involving large amounts of computation. Moreover, the size of the optimal partition may depend on the magnitude of the perturbation. Other approaches employ iterative solvers [Golub and van Loan 1996] using the unperturbed solution as an initial guess, sensitivity methods [Pillage et al. 1994] and the fictitious domain method [Fu et al. 2007]. These suffer from the fact that they require the full system to be solved again, for the entire chip, and do not fully utilize the "closeness" properties above. In particular, the fictitious domain method [Fu et al. 2007] defines a macromodel for the perturbed region and a "coating" region around it. This solver iteratively computes the port currents of this macromodel using the current solution of system at the boundary of the macromodel and then applies the updated port currents to the RHS of the system. Finally, it solves the entire system with the updated RHS. An iterative solver is also described in [Ye et al. 2008], but operates on smaller, denser systems.

An alternative approach is to identify the set of nodes that are significantly affected by the perturbation, called the *region of influence* (RoI). The RoI is the set of all of the nodes in the network for which the voltage changes by more than a given threshold under the applied perturbation changes. If the RoI could be found, one could, in principle, solve for a drastically smaller system of equations by keeping the solution of the nodes out of RoI at their initial values, and recomputing only the solutions within the RoI. Figure 1 schematically illustrates the dimension of LHS matrix corresponding to the full network and the much smaller subsystem corresponding to the RoI of a perturbation.

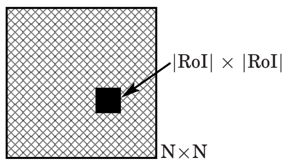

$|\text{RoI}| \times |\text{RoI}|$
$\text{N} \times \text{N}$

Fig. 1. Reduction in system size using RoIs.

Identifying the RoI is nontrivial. The work of [Boghrati and Sapatnekar 2010] uses the bookkeeping information of the random walk solver of [Qian et al. 2003] (which we will refer to as *forward random walks*), obtained as part of a preprocessing phase, to identify the RoI for any perturbation. This approach has two drawbacks: (1) it uses approximations that neglect second-order terms, leading to some errors, and (2) it neglects the fact that the bookkeeping information changes due to these perturbations, making this method increasingly inaccurate as more consecutive perturbations are made.

The work in this paper finds the RoI accurately and efficiently using the notion of *backward random walks*. The basic concept was outlined almost 50 years ago in [Hammersley and Handscomb 1964], but with no regard to computational efficiency. The backward random walk method is also known as shooting method in computer graphics, and has been used in the context of radiosity and illumination problems to determine the reflections of a light source on the environment [Castro et al. 2008]. The shooting method in graphics shares the idea of our approach, finding the affected region due to a source by running random walks from a source, but the problem structure is sufficiently different that the resemblance stops at a very superficial level, and solutions from that domain cannot be adapted easily.

In this work, we develop the theory of backward random walks and show how a solution for the incremental analysis problem can be obtained by applying this method. We employ the backward random walk method to (1) make the approach computationally practical, (2) show a theoretical relation to LU factorization, and (3) apply it to incremental analysis. Our incremental solver can capture any number of consecutive perturbations on the LHS and the RHS of the network of fixed size, without degradation in the accuracy, and hence overcomes the drawbacks of the forward random walks method. Experimental results show speedups of up to 13$\times$ compared to the Hybrid Solver of [Qian et al. 2005] on IBM power grid benchmarks of [Nassif 2008]. The key feature of the backward random walk approach is the ability to find some columns of G^{-1} efficiently, individually, and without incurring the large computational overhead of determining the entire G^{-1} matrix. Therefore, as opposed to the fictitious domain method [Fu et al. 2007] the RoI can be found directly and without any computation for the nodes that are not affected.

Further, we show the relationship between forward and backward random walks. In particular, forward random walks effectively construct G^{-1} by the row, while we show that backward walks construct G^{-1} by the column. Since G^{-1} for a symmetric G is symmetric, this implies that for a symmetric system, the forward and backward walks are equivalent, and our backward random walks can be substituted by the already-known theory of forward random walks. However, we also demonstrate that when a power grid modeled appropriately, the resulting system of equations results in voltage-controlled current sources that can result in an asymmetric G matrix. This is where the theory of backward walks developed in this paper has particular applicability, since we will show that incremental analysis requires specific columns of G^{-1} to be reconstructed.

The paper is organized as follows. A procedure for more accurately modeling power grids is discussed in Section 2, followed by a discussion of the backward random walk method in Section 3. Next, Section 4 discusses the proposed incremental solver, after which the relation between the backward random walks with LU decomposition of the LHS is presented in Section 5. Finally, Section 6 presents the experimental results, and the paper ends with a conclusion in Section 7.

2. POWER GRID MODELING

2.1. Power Grid Abstraction

Figure 2 shows the layout of a portion of a 2D power grid layout where several cells are connected to a V_{dd} line. A *detailed extracted circuit* for the power grid of this figure is depicted in Figure 3. In this figure, the resistors model the wire resistances and the triangles symbolize various cells connected to the power grid. The capacitance and the inductance of the network are not shown since we are focusing on the steady-state analysis of the grid. Similar to [Nassif 2008], it is assumed that all cell connections are at the lowest metal level and that the tapping points only lie in these layers. At any intermediate metal level, the connections are only made at vias.

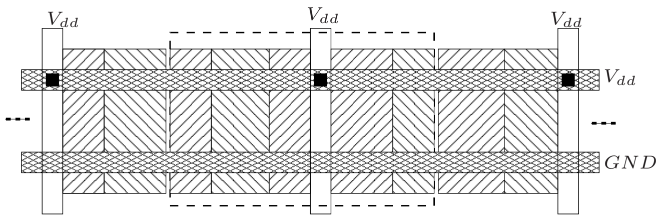


Fig. 2. An example of the layout of a portion of a power grid.

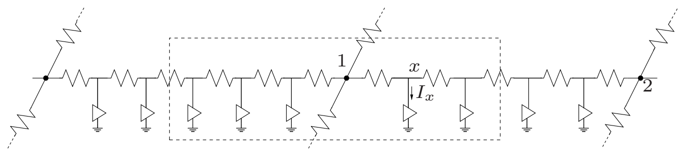


Fig. 3. Schematic of the detailed extracted power grid of Figure 2.

For a full chip power grid analysis, detailed extraction results in large systems of equations that are memory-intensive and computationally prohibitive, since every node with a current source must be retained. A simplifying assumption that is conventionally used in power grid extraction for steady-state analysis is to lump all the cells at the power line intersections, i.e., to assume the cells are connected to the power grid only at the power line intersections, which results in over all much smaller power grid to analyze, as illustrated in Figure 4. As pointed out in [Nassif 2008], this approximation has to be essential because it is impractical to model individual connections between the gates and power grid. The lumped approach of Figure 4 has several drawbacks:

— It overlooks the dependence of the current drawn by each cell on the voltage of their power supply [Nassif 2008].
— It introduces a discretization error by placing the lumped current source potentially far away from the original locations of the cells.

To account for the first-order dependence of cell current to the voltage of the power supply, each cell can be modeled as a Voltage Controlled Current Source (VCCS) with $I_x = g_x V_x + I_x^0$ for a cell at intermediate node x. Note that the VCCS here is fairly trivial, and is easily modeled by placing a conductance g_x in parallel with each current source. However, the second issue conventionally requires either an exact analysis of a much larger system (containing numerous nodes, corresponding to all cell tapping points on the power grid as shown in Figure 3), or an approximate analysis of a lumped system with discretization error.

In Section 2.2, we propose an approach for avoiding the discretization error in moving to a coarse extracted grid that eliminates all nodes but those at the intersections of grid wires, as in the simplified extracted grid. Using the VCCS model makes the task

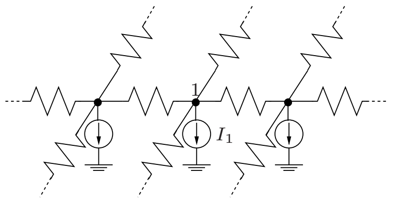


Fig. 4. Schematic of the simplified extracted power grid of Figure 2.

of going from detailed extracted power grid to a coarse extracted grid more difficult due to dependence of the current loads on intermediate nodes (i.e., nodes not on the power grid intersections) that are available only in the detailed extracted circuit. We employ circuit transformations to move the VCCS elements for each cell to VCCS elements at the intersections of the power grid. In effect, the circuit in Figure 3 is transformed to that in Figure 4, with the difference that the current sources are replaced by VCCS elements.

As we will see, these new VCCS elements are more complex than the simple current source/resistor elements that model individual cells. These elements result in a set of power grid equations that are diagonally dominant with negative off-diagonals, but *asymmetric*. Intuitively, this asymmetry arises because the voltage of each intermediate node is closer in value to voltage of the intersection node it is physically closer to. Therefore, when the intermediate nodes are eliminated, the VCCS may be a stronger function of one adjacent grid intersection node rather than another one, which, in terms of the nodal analysis stamp of a VCCS [Pillage et al. 1994], introduces asymmetry in the LHS of Equation (1).

2.2. Modeling the Power Grid using VCCS Elements

In this section, we first discuss the details of obtaining the coarser extracted power grid circuit from the detailed extracted power grid circuit, modeling each cell as a VCCS. Next, we discuss a method for obtaining such a model from a standard set of power grid benchmarks [Nassif 2008].

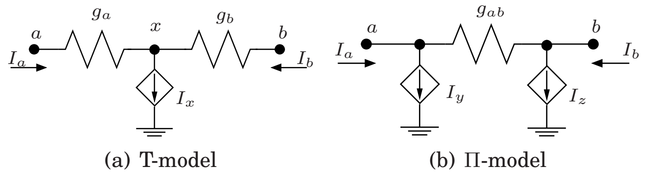


Fig. 5. Converting a T-model to a Π-model.

LEMMA 2.1. *Consider a circuit with two external ports at nodes a and b, represented either as a T-model shown in the circuit of Figure 5(a), or as a Π-model circuit of Figure 5(b), where the equations for the VCCS elements, I_x, I_y, and I_z, are*

$$\begin{aligned} I_x &= I_x^0 + g_{xa}v_a + g_{xb}v_b + g_x v_x \\ I_y &= I_y^0 + g_{ya}v_a + g_{yb}v_b \\ I_z &= I_z^0 + g_{za}v_a + g_{zb}v_b \\ g_{ab} &= \frac{g_a g_b}{g_a + g_b} \end{aligned}$$

The two models are electrically equivalent if the following relationships hold:

$$
\begin{aligned}
g_{ya} &= g_a\left(1 - \frac{g_a - g_{xa}}{\hat{g}_x}\right) - g_{ab}\\
g_{yb} &= g_{ab} - g_a\left(\frac{g_b - g_{xb}}{\hat{g}_x}\right)\\
I_y^0 &= \frac{g_a}{\hat{g}_x} I_x^0\\
g_{za} &= g_{ab} - g_b\left(\frac{g_a - g_{xa}}{\hat{g}_x}\right)\\
g_{zb} &= g_b\left(1 - \frac{g_b - g_{xb}}{\hat{g}_x}\right) - g_{ab}\\
I_z^0 &= \frac{g_b}{\hat{g}_x} I_x^0
\end{aligned}
\tag{2}
$$

where $\hat{g}_x = g_a + g_b + g_x$.

Proof: See Appendix A □

To obtain the coarsened power grid from the detailed extracted grid, Lemma 2.1 is progressively applied on the intermediate nodes of the detailed extracted grid to replace the T-models with Π-models. In each wire segment in the original circuit, shown in Figure 3, each gate at node x is represented by a conductance g_x in parallel with a constant current source; therefore, $g_{xa} = g_{xb} = 0$. The transformation in the lemma is then applied to a T-model in Figure 6(a), marked with a dashed square around it, to create the corresponding Π-model shown in Figure 6(b). Next, the adjacent VCCS elements are summed up to create new T-models, shown in Figure 6(c) that are successively reduced. Finally, each wire segment between intersections is reduced to the Π-model shown in Figure 6(d), and the final circuit has the look of the lumped circuits used in [Nassif 2008] with all sources at the power grid intersections, except that the sources at each node are not independent current sources, but VCCS elements that introduce asymmetry into the G matrix.

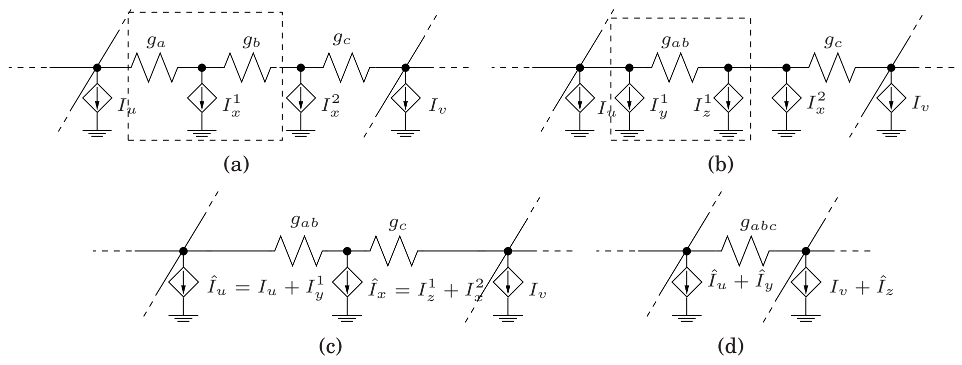


Fig. 6. Steps for eliminating an intermediate nodes in the detailed extracted grid where $g_{abc} = g_{ab}g_c/(g_{ab} + g_c)$.

Note that the above process is exact and makes no approximations, implying that the impact of the voltage of the power line at each cell is accounted for accurately.

Therefore, the dependence of the load of each cell on its supply voltage is correctly captured, and the discretization error mentioned in Section 2.1 is avoided.

Table I. Quantifying the inaccuracy in voltage estimation using lumped current source models.

benchmark	Current source model error			
	Norm. to V_{dd}		Norm. to max voltage drop	
	Mean	Max	Mean	Max
ibmpg1	6.14%	28.38%	14.96%	69.11%
ibmpg2	5.75%	14.51%	32.26%	81.48%
ibmpg4	0.05%	0.14%	8.98%	25.43%
ibmpg5	0.12%	0.52%	3.49%	14.63%
ibmpg6	1.07%	3.72%	12.91%	44.85%
Average	**2.63%**	**9.45%**	**14.52%**	**47.10%**

Table I quantifies the inaccuracy that is inherent in the conventional lumped model of Figure 4. For the detailed extracted circuits corresponding to power grid benchmarks of [Nassif 2008], which are created using the procedure explained in Appendix B, we quantify the error introduced by the lumped approximation. The average and maximum modeling errors shown in the table are normalized to the value of V_{dd} and also normalized to value of maximum voltage drop in the power grid. The error as normalized to V_{dd} is large for the first two benchmarks and smaller for the others. This can be attributed to the fact that the worst-case drops on these circuits are larger. An alternative, and possibly better, metric for measuring the error is to compare the error as a fraction of the maximum voltage drop over all nodes in the circuit[1]. Under this metric, it can be seen that the error is quite significant for all circuits, motivating the need for using our VCCS model over the existing lumped model.

The higher accuracy of the VCCS model comes at the expense of an *asymmetrical* LHS. The hybrid solver of [Qian and Sapatnekar 2008] and forward solver of [Boghrati and Sapatnekar 2010] are not capable of solving a system with asymmetrical LHS and it is harder for direct and iterative methods to solve. A direct solver for an asymmetrical system computes the LU factors of the LHS which requires $2n^3/3$ floating point operations rather than Cholesky factorization for symmetrical LHS that requires $n^3/3$ floating point operations. For instance, computing the LU factors of the VCCS model of *ibmpg1* takes $4.3\times$ more than computing the Cholesky factors of the symmetrical model of this circuit. To solve an asymmetrical system iteratively, Preconditioned Biconjugate Gradient Stabilized (BiCGSTAB) method can be used which requires more memory and its convergence is inferior to the Preconditioned Conjugate Gradient (PCG) method which is applicable only to symmetrical systems [Saad 2003]. In this paper, we propose a backward random walk solver that can solve asymmetrical equations as efficiently as symmetric equations.

3. FORWARD AND BACKWARD RANDOM WALKS

3.1. Forward Random Walks

The forward random walk game [Qian et al. 2003] is a construction for solving Equation (1) as $\mathbf{V} = G^{-1}\mathbf{E}$, that is based on the *rows* of LHS matrix, G. The method proceeds by solving the voltage at any specific node i as

$$v_i = \left[\, (G^{-1})_{i1} \;\; (G^{-1})_{i2} \;\; \ldots \;\; (G^{-1})_{iN} \,\right] \mathbf{E} \tag{3}$$

where $(G^{-1})_{ij}$ is the $(i,j)^{\text{th}}$ element of G^{-1}, and N is the dimension of G.

[1]We choose not to show normalization of the error at a node with respect to the voltage drop at the node since it exaggerates small errors at nodes with small voltage drops, thus providing largely meaningless data.

The forward random walk game is based on a network of roads with motels or homes at its intersections. To compute the voltage at node i, the walker starts with zero money in his pocket from the motel at i, pays for the cost of the motel (on credit), and takes one of the roads at the intersection randomly, according to some known probability distribution p_{ij}, to get to an adjacent intersection, j, where $j \in \{1, \ldots, \text{degree}(i)\}$, and degree$(i)$ denotes the number of roads at intersection i. This process of paying for the motel at the current node and randomly picking the next node is repeated until the walker reaches one of the specially designated home nodes in the circuit, where a reward is collected. For suitably chosen probabilities, motel costs, and rewards, the number of visits, including the revisits, to each motel normalized to the total number of walks started from node i, can be shown [Qian et al. 2005] to be the corresponding elements of the i^{th} row of the G^{-1}. Moreover, using this information the voltage of node i can be computed simply by multiplying the computed row of G^{-1} to the RHS vector which is equivalent to the average sum of money in walker's pocket at the end walk, averaged over a large number of walks [Qian et al. 2005].

3.2. Motivation for Computing G^{-1} by the Column

In effect, as shown in [Qian and Sapatnekar 2005; 2008], the forward random walk method computes G^{-1} by the row for the row(s) of interest, corresponding to the nodes where the voltages are to be evaluated. Reconsidering Equation (1), its solution $\mathbf{V} = G^{-1}\mathbf{E}$, can also be written in another form as:

$$\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{bmatrix} = \begin{bmatrix} (G^{-1})_{11} \\ (G^{-1})_{21} \\ \vdots \\ (G^{-1})_{N1} \end{bmatrix} e_1 + \cdots + \begin{bmatrix} (G^{-1})_{1N} \\ (G^{-1})_{2N} \\ \vdots \\ (G^{-1})_{NN} \end{bmatrix} e_N \tag{4}$$

where v_i and e_i are the i^{th} elements of $\mathbf{V}$ and $\mathbf{E}$, respectively.

Each element, $\left[(G^{-1})_{1i}, (G^{-1})_{2i}, \ldots, (G^{-1})_{Ni}\right]^T e_i$, of the summation is the contribution of the i^{th} element of the RHS, $\mathbf{E}$, on the solution vector $\mathbf{V}$. The full solution simply is the superposition of the contributions of all e_i. In the context of ground network analysis, this is the contribution of each current load on the node voltages of the network. In other words, each product within the sum represents the influence of e_i, and the nodes with near-zero values fall outside the RoI of e_i.

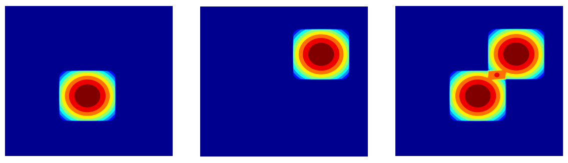


Fig. 7. Contribution of individual current loads on the power network solution.

On-chip power grids with C4 pins distributed over the area of the chip are well known to satisfy the property of locality, whereby the voltage drop impact of an individual current source is predominantly felt at nodes that lie in the region near the source, and not in faraway areas. This is illustrated in the schematic contour plot in Figure 7(a), where a current load is applied at a certain node and its influence on the voltage drop (effectively, computing $\mathbf{V} = G^{-1}\mathbf{E}$ by the column corresponding to this single excitation) is seen to show the property of locality. If another excitation is applied due to another current load, as shown in Figure 7(b), another contour plot is obtained, and if both are applied together (Figure 7(c)), then the result is a superposition of the contours, consistent with Equation (4).

While this observation is of limited utility in the complete solution of $G\mathbf{V} = \mathbf{E}$ for a realistic power grid, where the number of current loads is extremely large, it has particular application to the problem of incremental analysis. As we will see in Section 4, the impact of small perturbations to the power grid can be modeled by a small number of current loads. Therefore it is reasonable to attempt to compute the matrix inverse by the column, for the columns corresponding to these current loads, in order to generate a meaningful solution in a modest amount of time. For such a case, the nodes affected that constitute the RoI can be computed using a method similar to that illustrated in Figure 7. Therefore, there is a close relation between finding the RoI and Equation (4). As we will show soon, the use of backward random walks permits this computation in an easy manner.

3.3. The Backward Random Walk Game

In this section, we qualitatively explain the backward random walk method in contrast with the forward random walk approach. The key difference is that the forward game is constructed based on the *rows* of the LHS matrix, G, inverting one row at a time, and captures the effect of all of the RHS elements on the solution of a single node; while the backward game is constructed based on the *columns* of G, inverting one column at a time, and captures the effect of a single source on the solution of the entire system. Therefore, the forward game finds the matrix inverse, G^{-1}, by the row, and the backward game finds this matrix inverse by the column.

By Equation (4), for the case where there are few nonzero elements on the RHS (e.g., during incremental analysis), only a few corresponding columns of G^{-1} must be computed, and backward random walks are extremely appealing. The forward game, on the other hand, is suitable for the case where the voltages of a small subset of nodes in the network are desired, as seen from Equation (3).

As described later in Section 3.4, the construction of the forward and backward games is the same except in the fact that the forward walk road probabilities are constructed based on the *rows* of G, while the road probabilities of the backward game are based on the *columns* of this matrix. In both games multiple walks are started from a *motel* of interest, ended when a *home* is reached, and the motel visits are recorded. The difference then is how this information is interpreted. In a forward game, as mentioned in Section 3.1, the analogy is that the walker has to pay for each motel he visits and gets an award as he reaches a home. In this process the average number of visits to motel nodes are translated into the rows of G^{-1} and the sum of money into node voltages, in other words, sum of the contribution of each motel (RHS element) on the walker's total money.

In contrast, the analogy for the backward game is that the walker has a sum of money that he has to distribute among the motels he visits based on how frequently he visits them. Then the average number of motel visits is translated to the columns of G^{-1} and the sum of money (the contribution of one RHS element on entire system) distributed on all the nodes in the game, into the contribution of one RHS entry on the voltage of all nodes.

For a symmetrical G, the forward and backward games turn out to be identical and the columns of G^{-1} can be obtained by transposing the corresponding rows of G^{-1} computed by forward random walks and vice versa. However, as mentioned in Section 2, the LHS of the power grid equations can be *asymmetrical* where the forward and backward games turn out to be different. Like the forward solver, the backward solver is also directly related to the LU decomposition of the LHS matrix, as discussed in Section 5). As in [Qian and Sapatnekar 2005; 2008], this can be used as a preconditioner in combination with an iterative solver, but this is not explored in detail within this paper.

3.4. Constructing the Backward Walk Game

We now describe how the backward random walk game is constructed from Equation (1), and how the columns of G^{-1} are computed using this approach. As in the forward walk game, this game is based on a network of roads with motels or homes at its intersections. To compute the j^{th} column of G^{-1}, the walker starts from the motel at j, pays for the motel, and takes one of the roads at the intersection randomly, according to some known probability distribution p_{ji}, to get to the adjacent intersection, i, where $i \in 1, \ldots, \text{degree}(j)$, and degree$(j)$ denotes the number of roads at intersection j. The walker continues until a home is reached. This completes one full *walk*. The number of motels visited, including the revisits, on the path is called *walk length*. The goal of this game is to enumerate the visits to each motel during a walk.

THEOREM 3.1. *The LHS matrix G of the power grid with asymmetric VCCS models satisfies the following properties:*

$$\sum_{i=1, i\neq j}^{N} |g_{ij}| \leq g_{jj},\ \forall j \tag{5}$$

$$\begin{cases} g_{ij} \ \leq \ 0 \ \ j \neq i \\ g_{ij} \ > \ 0 \ \ j = i \end{cases} \tag{6}$$

Proof: See Appendix C. □

For power grid LHS matrix G with these properties, Equation (5) can be modified as:

$$\begin{aligned} \sum_{i=1, i\neq j}^{N+1} |g_{ij}| \ &= \ g_{jj},\ \forall j \\ g_{j(N+1)} \ &= \ -\left(g_{jj} - \sum_{i=1, i\neq j}^{N} |g_{ij}| \right) \ \forall j \end{aligned} \tag{7}$$

The added conductances are accounted for in Equation (1) by adding dummy node $N+1$ with known solution of 0. The insertion of the dummy node is essential to obtain valid road probability distributions discussed in Theorem 3.2. In the rest of the paper, we assume that G notation is overloaded to include this added dummy node.

THEOREM 3.2. *For each column of a system of linear equations defined by Equation* (1)*, a valid probability mass distribution can be defined as:*

$$p_{ij} = \begin{cases} -g_{ji}/g_{jj} & j \in [1, N], j \neq i \\ 0 & j = i \end{cases} \tag{8}$$

Proof: For a probability mass function to be valid, all probabilities should lie between 0 and 1, and the sum of all probabilities should be 1. For the conductance matrix of a power grid, G, using Equations (6) and (7), for $\forall i, j$, we have:

$$\begin{aligned} 0 \leq p_{ij} \leq 1 \\ \sum_{i=1}^{N} p_{ij} = 1 \end{aligned}$$

□

Note that in Equation (8), $p_{ij} = 0$ for many of the j's due to the sparsity of G. Writing the p_{ij}'s in matrix form, P, where subscript i and j denote the row and column index of this matrix, we get:

$$G = (I - P)^T D \tag{9}$$

where D is the matrix of the diagonals of G, and I is the identity matrix. Notice that as mentioned in Section 3.3, the road probabilities on the rows of matrix P are computed using the columns of the LHS matrix, G, which distinguishes the backward game from the forward game.

A random walk game can be constructed from Equation (1) by modeling each element of the vector $\mathbf{V}$ as an intersection, nonzero elements of the LHS as roads, and road probabilities as in Equation (8). Note that the vector $\mathbf{V}$ consists of both unknowns as well as known variables. The unknown and known variables of vector $\mathbf{V}$, are mapped to motels and homes, respectively.

Note that in construction of the game, connectivities and transition probabilities, and therefore number of node visits, are determined by the LHS only.

3.5. An Example Illustrating the Forward and Backward Games

Consider the following diagonally dominant matrix:

$$G = \begin{bmatrix} 1.5 & 0 & -1 & 0 \\ 0 & 2 & -1 & -0.5 \\ -0.75 & -1.25 & 2.25 & -0.25 \\ 0 & 0 & -0.25 & 1.25 \end{bmatrix} \tag{10}$$

Since we are interested in illustrating the setup for the backward [forward] random walks required to create G^{-1} by the column [row] (rather than the solution vector $\mathbf{V}$), we do not show the RHS vector $\mathbf{E}$ in this example.

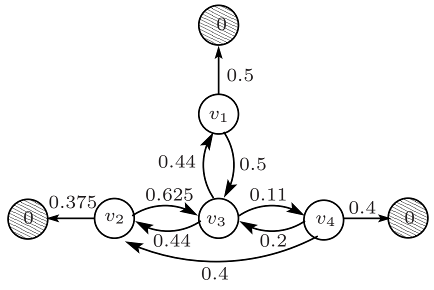


Fig. 8. An example of a *backward* random walk game modeling Equation (10).

To construct the backward game corresponding to this matrix, a dummy node is added to this equation such that the columns of G sum up to zero as described in Section 3.4. Next the *columns* of G is normalized to its diagonals, such that it can be written in the form of Equation (9), a row-wise probability matrix (corresponding to the transpose of the normalized G). Figure 8 shows the random walk game, with the probability matrix:

$$P_{backward} = \left[\begin{array}{cccc|c} 0 & 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0.625 & 0 & 0.375 \\ 0.44 & 0.44 & 0 & 0.11 & 0 \\ 0 & 0.4 & 0.2 & 0 & 0.4 \end{array}\right] \tag{11}$$

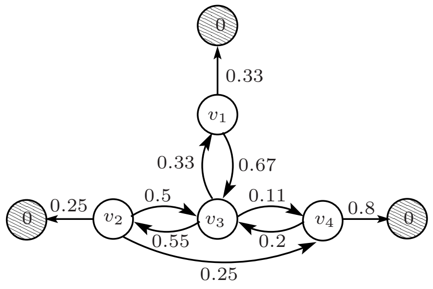


Fig. 9. An example of a *forward* random walk game modeling Equation (10).

Note that in this equation the last column corresponds to the ground node, to generate all probabilities as in Equation (7).

Similarly the forward random walk game is constructed based on probability matrix of Equation (12) which is computed by adding a dummy node to G such that its rows sum up to zero, and normalizing the *rows* of matrix G to its diagonals. Figure 9 shows the corresponding game.

$$P_{forward} = \left[\begin{array}{cccc|c} 0 & 0 & 0.67 & 0 & 0.33 \\ 0 & 0 & 0.5 & 0.25 & 0.25 \\ 0.33 & 0.55 & 0 & 0.11 & 0 \\ 0 & 0 & 0.2 & 0 & 0.8 \end{array}\right] \tag{12}$$

Having the random walk game constructed, the following section shows how it is used to compute the columns of the G^{-1} individually without computation of the entire G^{-1}.

3.6. Computing the Individual Columns of G^{-1}

In this section we show how the individual columns of G^{-1} can be computed using backward random walks.

Carrying out M walks from intersection j, one can find the conditional expected number of visits to each motel by:

$$\begin{aligned} z_{ij} &= \frac{\text{Number of visits to motel } i \text{ in } M \text{ walks from } j}{M} \\ Z &= [z_{ij}]_{N\times N} \end{aligned} \tag{13}$$

where z_{ij} is the expected number of visits to motel i when total of M walks initiated from motel j. This expected value becomes more exact as $M \to \infty$.

It is shown in [Hammersley and Handscomb 1964] that the backward random walk game constructed based on the of probability matrix P, as described in Section 3.2, gives the solution to:

$$\begin{aligned} (I-P)^T\mathbf{X} &= \mathbf{U} \\ \mathbf{X} &= Z\mathbf{U} \end{aligned} \tag{14}$$

where

$$Z = (I-P)^{-T} \tag{15}$$

Translating the notation of [Hammersley and Handscomb 1964] to the notation in our problem,

$$\begin{aligned} \mathbf{X} &= D\mathbf{V} \\ \mathbf{U} &= \mathbf{E} \end{aligned} \tag{16}$$

Substituting Equations (15) and (16) into Equation (14) we can see that the backward random walk approach, as formulated in [Hammersley and Handscomb 1964], provides the solution to Equation (1). Specifically,

$$\begin{aligned} D\mathbf{V} &= (I-P)^{-T}\mathbf{E} \\ \text{i.e., } \mathbf{V} &= D^{-1}(I-P)^{-T}\mathbf{E} = G^{-1}\mathbf{E} \end{aligned}$$

where the last equality follows from Equation (9).

Finally, from Equation (9) and (15), we can see that $G^{-1} = D^{-1}Z$. Therefore, the elements of G^{-1} can be written as:

$$(G^{-1})_{ij} = \frac{z_{ij}}{g_{jj}},\ i = 1,\ldots,N \tag{17}$$

For the example of Figure 8, let us consider the use of backward random walks to compute the second column of the exact G^{-1}. Table II shows the calculated value of this column, denoted by $(G^{-1})_{*2}$, and the average relative error, in percent, of the estimation. In order to avoid bias from a specific run of the random walk method, we show the average is over 100 runs of the random walk method.

This table suggests that as M increases, the random walk results become more accurate. As we will see later in Section 3.7 this table also suggests that the error of the random walk results halves for 4 times more walks. In other words the error of the random walk method is proportional to $1/\sqrt{M}$. Note that for even a small number of walks, the results are fairly accurate. Hence, if a rough but fast estimation of G^{-1} is desired, backward random walks is a suitable candidate.

Table II. Average relative error, in percent, for estimated G_2^{-1} for different walk numbers, M.

$(G^{-1})_{*2}$	$M = 25$	$M = 100$	$M = 400$	$M = 1600$
0.4115	26.7%	12.0%	5.8%	2.9%
0.8395	10.1%	4.9%	2.5%	1.2%
0.6173	21.3%	8.9%	4.6%	2.4%
0.1235	34.3%	18.9%	10.8%	5.1%

3.7. Stopping Criteria for the Backward Random Walks

In this section we show how the number of walks is determined dynamically as the random walks are proceeding to get a desired accuracy. Here the accuracy is defined as the relative error of the absolute sum of the solution, $\mathbf{V}$, from Equation (1) as defined by:

$$err = \frac{\sum_{i=1}^{N}(|v_i| - |v_i^*|)}{\sum_{i=1}^{N}|v_i^*|} \tag{18}$$

where v_i and v_i^* are the i^{th} element of the random walk solution and the exact solution respectively, and $|.|$ denotes the absolute value.

For M full walks starting from motel j, we take advantage of the fact that the total walk length is equal to the total number of visits to all of the motels. Mathematically, this is stated as:

$$\sum_{i=1}^{N} z_{ij} = \frac{1}{M}\sum_{s=1}^{M} w_{sj} \tag{19}$$

where z_{ij} is defined by Equation (13) and w_{sj} represents the length of the s^{th} walk starting from j.

Moreover, if W_j represents the random variable of the length of the walks starting from node j, we have:

$$\frac{1}{M}\sum_{s=1}^{M} w_{sj} \approx E[W_j] \tag{20}$$

where $E[W_j]$ is the expected value of W_j. This estimation becomes exact as the number of walks tends to infinity.

Theorem 3.3 shows relation of the relative error to the average walk length, which is calculated as the random walk game proceeds, and is used to estimate the relative error of the solution on the fly.

THEOREM 3.3. *The relative error of Equation* (18)*, corresponding to j^{th} motel, is given by:*

$$err = \frac{\frac{1}{M}\sum_{s=1}^{M} w_{sj} - E[W_j]}{E[W_j]} \tag{21}$$

Proof: Consider a backward walk starting from node j, designed to determine the contribution v_i^j of the j^{th} element of $\mathbf{E}$, denoted by e_j, on the voltage of some node i. We obtain:

$$v_i^j = (G^{-1})_{ij}e_j = \left(\frac{z_{ij}}{g_{jj}}\right)e_j \tag{22}$$

where the second equality follows from Equation (17). The sum of the absolute values of this error, over all nodes i, is then given by:

$$\sum_{i=1}^{N} |v_i^j| \;=\; \frac{|e_j|}{|g_{jj}|}\sum_{i=1}^{N} z_{ij} \tag{23}$$

Substituting $\sum_{i=1}^{N} z_{ij}$ from Equation (19) we have:

$$\sum_{i=1}^{N} |v_i^j| \;=\; \frac{|e_j|}{M|g_{jj}|}\sum_{s=1}^{M} w_{sj} \tag{24}$$

As $M \to \infty$, the solution of random walk method converges to the exact solution v_i^{j*}, and we have:

$$\sum_{i=1}^{N} |v_i^{j*}| \;=\; \frac{|e_j|}{|g_{jj}|}E[W_j] \tag{25}$$

where v_i^{j*} is the i^{th} element of the exact solution corresponding to j^{th} column. Substituting Equation (24) and (25) into Equation (18) completes the proof. □

Incremental solution of a system for a given perturbation, involves computation of several columns of G^{-1} corresponding to nonzeros of RHS. Theorem 3.3 is useful to compute each of these columns individually with a desired relative error corresponding to one nonzero element of RHS. It is easy to verify that the result of Theorem 3.3 holds for general case that there are multiple nonzeros in the RHS. Therefore the relative error of the updated solution corresponding to all the nonzero elements of the RHS is the same as the relative error of the individual columns. Note that the sum of several variables with the same relative error will have the same relative error.

An immediate result of Theorem 3.3 is the number of walks required to achieve a desired relative error of δ with a confidence of α. In other words to have:

$$prob\left[\left|\frac{1}{M}\sum_{s=1}^{M} w_{sj} - \mu\right| < \mu\delta\right] > \alpha \tag{26}$$

where $prob[.]$ is the probability function and μ is the expected value of the random variable of the average walk length, $\frac{1}{M}\sum_{s=1}^{M} w_{sj}$. The walk length average is the sample average of a series of identical and independent (I.I.D.) random variables, W_j, the random variable of the length of walks starting form j. Therefore according to the central limit theorem we have:

$$\frac{1}{M}\sum_{s=1}^{M} w_{sj} \sim \mathcal{N}(\mu, \sigma) \tag{27}$$

where $\mathcal{N}(\mu, \sigma)$ denotes the normal distribution with mean of $\mu = \mu_{W_j}/M$ and standard deviation of $\sigma = \sigma_{W_j}/\sqrt{M}$, where μ_{W_j} and σ_{W_j} are the mean and standard deviation of W_j.

Taking advantage of the normal distribution of Equation (27), Equation (26) will be satisfied if the following holds:

$$\frac{(\sigma/\mu)^2}{M} < \left(\frac{\delta}{Q^{-1}\left(\frac{1-\alpha}{2}\right)}\right)^2 \tag{28}$$

where Q^{-1} is the inverse of the Q-function, defined as $Q(x) = 1/\sqrt{2\pi}\int_x^\infty e^{-u^2/2}du$.

In Equation (28), note that $M \propto 1/\delta^2$. Therefore, in order to get two times more accurate solution, four times more walks is required. As a result, random walks are efficient for moderate accuracies but for higher accuracies this method could be expensive.

4. INCREMENTAL SOLVER

We now propose an efficient incremental solver based on backward random walks in the framework of power network analysis. Our incremental solver proceeds under the reasonable assumption that the solution to the unperturbed system of equations, $G\mathbf{V} = \mathbf{E}$, is known. When the design is perturbed, it may result in a change in either G or $\mathbf{E}$, resulting in the new relationship:

$$(G + \Delta G)(\mathbf{V} + \Delta\mathbf{V}) = \mathbf{E} + \Delta\mathbf{E} \tag{29}$$

where ΔG models the change in the LHS, $\Delta\mathbf{E}$ models the change in the RHS, and $\Delta\mathbf{V}$ is the change in the solution caused by the perturbation. This equation can be rewritten as:

$$\begin{aligned} (G + \Delta G)\Delta\mathbf{V} &= \Delta\mathbf{E} - \Delta G\mathbf{V} \\ G_{\text{eff}}\Delta\mathbf{V} &= \Delta\mathbf{E}_{\text{eff}} \end{aligned} \tag{30}$$

where $\Delta\mathbf{E}_{\text{eff}} = \Delta\mathbf{E} - \Delta G\mathbf{V}$ is the effective change in the RHS and $G_{\text{eff}} = G + \Delta G$ is the total perturbed LHS.

Note that in contrast with [Boghrati and Sapatnekar 2010], where the second order term, $\Delta G\Delta\mathbf{V}$, is ignored, Equation (30) captures any perturbation to the system without any approximation. Moreover, this method does not make any assumptions regarding the nature of the perturbation as long as the number of nodes of the network is fixed. Further, since the perturbed system models a power grid, the LHS is diagonally dominant, and all of the off-diagonals are less than or equal zero, therefore it can be solved using random walks.

The steps followed by the proposed incremental solver are:

Step 1. Solve Equation (30) using backward random walks.

Step 2. Find the RoI using the computed solution.

Step 3. Refine the solution for the nodes within RoI.

Step 1: The first step of the incremental solution involves finding the columns of G_{eff}^{-1} corresponding to nonzeros of $\Delta\mathbf{E}_{\text{eff}}$. For a small perturbation, most of the LHS matrix, G, and the RHS vector, $\mathbf{E}$, will be unchanged and therefore most of the entries of $\Delta\mathbf{E}_{\text{eff}}$ are zero. And if there are η nonzeros in $\Delta\mathbf{E}_{\text{eff}}$, we have:

$$\eta \ll |\mathbf{RoI}| \ll N \tag{31}$$

where $|\mathrm{RoI}|$ and N are the size of RoI and Equation (30), respectively. As a result, *only a few columns* of G_{eff}^{-1} must be computed, corresponding to the η nonzeros of $\Delta\mathbf{E}_{\text{eff}}$. Then, $\Delta\mathbf{V}$ is given by:

$$\Delta\mathbf{V} = \left[G_{\text{eff}}^{-1}\right]_{N\times\eta} \left[\Delta\mathbf{E}_{\text{eff}}\right]_{\eta\times 1} \tag{32}$$

where $\left[\Delta\mathbf{E}_{\text{eff}}\right]_{\eta\times 1}$ denotes η nonzeros of $\Delta\mathbf{E}_{\text{eff}}$, and $\left[G_{\text{eff}}^{-1}\right]_{N\times\eta}$ denotes the columns of G_{eff}^{-1} corresponding to these nonzeros.

This step of the algorithm relies on the fact that random walk solver is capable of finding an estimate of the solution efficiently with moderate accuracy, just enough to identify the RoI that corresponds to the nodes that are significantly affected.

As discussed in Section 3.7, the accuracy of the random walk solution is proportional to the square root of the number of walks, and it is not efficient for very high accuracies. Therefore, this step merely feeds Step 2, which finds the RoI based on this solution, and a precise solution is found in Step 3.

In this work a relative tolerance of 30% is used for the random walk solver. This means that the relative error of $G_{\text{eff}}^{-1} \times (\Delta\mathbf{E}_{\text{eff}})_j$ is less than or equal to 30% with a confidence of $\alpha = 99\%$, where $(\Delta\mathbf{E}_{\text{eff}})_j$ denotes the vector of $\Delta\mathbf{E}_{\text{eff}}$ where all of its elements are set to zero except for the j^{th} one.

Step 2: The second step of the incremental solver compares the computed estimate of $\Delta\mathbf{V}$ given by Equation (32), to a user-defined tolerance, tol, to determine the RoI. Specifically, node j is said to lie within the RoI if $\Delta V_j > tol$.

In order to account for the potential errors in the estimated solution, a safety margin (i.e., $s < 1$) is used to get a pessimistic RoI where the criterion for putting node j in this pessimistic RoI is $\Delta V_j > s \times tol$. A pessimistic RoI contains all of the nodes with potentially substantial change and the computational expense is passed on to the exact solver that operates on the small RoI region, whose size is $\ll N$. In this work, the safety margin is chosen empirically to be $s = 1/3$.

Step 3: The last step of *refinement* involves the application of an exact solver. In this step, we leverage the initial solution of the network, $\mathbf{V}$, the estimated changes computed using random walks, $\Delta\mathbf{V}$, and the RoI. Reordering Equation (30) according to RoI we have:

$$\begin{bmatrix} G_{\text{eff}}^{(in,in)} & G_{\text{eff}}^{(in,out)} \\ G_{\text{eff}}^{(out,in)} & G_{\text{eff}}^{(out,out)} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{V}^{(in)} \\ \Delta\mathbf{V}^{(out)} \end{bmatrix} = \begin{bmatrix} \Delta\mathbf{E}_{\text{eff}}^{(in)} \\ \Delta\mathbf{E}_{\text{eff}}^{(out)} \end{bmatrix} \tag{33}$$

where the superscripts in and out denote the nodes inside and outside of the RoI, respectively.

Although $\Delta\mathbf{V}^{(out)}$ should be set to 0 from the definition of the RoI, in practice, we find that it is more appropriate to use the constant (but small) value of $\Delta\mathbf{V}^{(out)}$ from Step 1, which enables us to be less conservative with the RoI. Therefore, we solve the above equation for $\Delta\mathbf{V}^{(in)}$, keeping the $\Delta\mathbf{V}^{(out)}$ unchanged from Step 1, by solving:

$$G_{\text{eff}}^{(in,in)}\Delta\mathbf{V}_r^{(in)} = \Delta\mathbf{E}_{\text{eff}}^{(in)} - G_{\text{eff}}^{(in,out)}\Delta\mathbf{V}^{(out)} \tag{34}$$

where $\Delta\mathbf{V}_r^{(in)}$ is the refined solution of the nodes within RoI. The size of this equation, $|\text{RoI}|$, is significantly smaller than N as illustrated in Figure 1. For this small system, any direct or iterative solver can be used; here, we use LAPACK [Anderson et al. 1999]. The total solution is then computed by adding $\Delta\mathbf{V}_r^{(in)}$ for the nodes inside the RoI, and $\Delta\mathbf{V}^{(out)}$ for nodes out of RoI, to the initial solution $\mathbf{V}$.

5. BACKWARD WALKS AND LU DECOMPOSITION

Having developed the idea of backward random walks as an alternative to forward random walks, and having applied backward walks to efficiently solve the incremental analysis problem, we now explore another novel theoretical result. We study an interesting correspondence between the backward random walks of Section 3 with LU decomposition of the LHS matrix of Equation (1), G. This relation can be used to find a quick and moderately accurate LU factorization of G which can be used for a variety of applications, e.g., as a preconditioner for an iterative method similar to the work of [Qian and Sapatnekar 2008]. The work of [Qian and Sapatnekar 2008] showed the relation between the UL factors of the LHS and the forward random walks, and this is its counterpart for backward walks. This relationship is not specifically used by the incremental solver but is pointed out in this paper for completeness.

The basic idea behind this relation is the notion of *reusing* the walks. Taking a second look at the backward random walk game described in Section 3, it is easy to see that during a walk, when the walker arrives at an intersection that has been frequently visited in the past, further walks are not necessary: the walker already has all the necessary information. Therefore, for the purposes of the game, he can simply stop the walk and use the previous information, i.e., reuse the previous walks.

In practice, this notion is implemented simply by marking a processed motel as a home and keeping two separate visit records, one for motel visits, and one for the stops at the home nodes which are the previously processed motels. Formally, we define

$$\begin{aligned} q_{ij} &= \frac{\text{Number of visits to motel } i \text{ in } M \text{ walks from } j}{M} \\ h_{ij} &= \frac{\text{Number of stops at home } i \text{ in } M \text{ walks from } j}{M} \end{aligned}$$

Writing these in matrix format, we have:

$$Q = [q_{ij}]_{N\times N}\,,\ H = [h_{ij}]_{N\times N} \tag{35}$$

where Q and H contain the motel and home visit records, respectively. Note that the matrices Q and Z have similar definitions, but the difference is that the Q matrix incorporates the effect of stopping at home nodes that are defined during the process of solving the system. As discussed shortly, Q is a lower triangular matrix while Z is a full matrix. The G matrix can be constructed from either Z and D or from Q, H, and D.

Without loss of generality, assume that the motels are processed in natural order. Hence, when processing motel l, all of the motels $j < l$ are previously processed and marked as home. As a result, the indices of all motels that the walker visits on his way are greater than or equal to the starting motel index. Similarly, the indices of all of the home nodes that the walker stops at are strictly less than the starting motel index. Therefore, Q and H will be lower-triangular and strictly upper-triangular matrices, respectively.

As discussed in Section 3.6 there is a direct relation between the backward random walks and G^{-1}, through Equation (13). Similarly, there is a direct relation between H, Q and G^{-1}, stated in Theorem 5.1 below. The idea behind this relation is that the full visit records of Equation (13) can be reconstructed from columns of H and Q.

THEOREM 5.1. *Given the visit records, $\{H, Q\}$, G^{-1} can be found by:*

$$\begin{cases} Z_1 = Q_1 \\ Z_j = Q_j + \sum_{i=1}^{j-1} h_{ij} Z_i, \ j = 2, \dots, N \end{cases} \tag{36}$$

$$G^{-1} = D^{-1} Z \tag{37}$$

where Z_j and Q_j denote the j^{th} column of Z and Q, respectively, and D is the matrix of the diagonals of G.

Proof: The case of $j = 1$ is trivial since no processed motel exists for the walker to stop at.

For the case of $j > 1$, by definition, out of M walks started from node j, $M \times h_{ij}$ walks stop at node $i < j$. Playing by the rules of the original game, i.e., not stopping at node i, is equivalent to starting $M \times h_{ij}$ new walks from node i after stopping in the modified game. Doing so, walker visits node $k \in [1, N]$, $M \times h_{ij} \times z_{ki}$ times. Accounting for all the nodes that the walker stopped at during the M walks from node j, number of visits to node k missed due to stopping, $M \times \hat{z}_{ki}$, is:

$$M \times \hat{z}_{ki} = M \sum_{i=1}^{j-1} h_{ij} z_{ki} \tag{38}$$

Adding the number of visits to node $k \geq j$, the total number of visits to node k in M walks from node j, $M \times z_{kj}$, we have:

$$M q_{kj} + M \sum_{i=1}^{j-1} h_{ij} z_{ki} = M z_{kj} \tag{39}$$

substituting in Equation (17) and writing in vector format completes the proof. □

Equation (36) can be rewritten in matrix format as:

$$\begin{aligned} G^{-1} &= D^{-1}(I - H)^{-1} Q \\ G &= Q^{-1}(I - H) D = L_G U_G \end{aligned} \tag{40}$$

where $L_G = Q^{-1}$ and $U_G = (I - H)D$ are the lower-triangular and upper-triangular factors of G. For a symmetrical G, LDL and Cholesky factorization can be computed from D, H, and diagonals of Q, without actually computing Q^{-1}, similar to [Qian and Sapatnekar 2008].

6. EXPERIMENTAL RESULTS

To demonstrate the performance of the proposed backward random walk incremental solver, this solver is implemented in C++ and is compared against the Hybrid solver of [Qian and Sapatnekar 2005] and the forward random walk incremental solver of [Boghrati and Sapatnekar 2010] on the original IBM Power Grid (IBMPG) benchmarks [Nassif 2008]. We choose to compare against these two specific solvers for the following reasons. The forward random walk incremental solver, which uses forward random walks to identify the RoI, is similar to the incremental solver of this work and is a predecessor of this work. The hybrid solver is an efficient public domain solver that uses random walks to obtain a high quality preconditioner that is then used to solve the entire power grid equation with the Preconditioned Conjugate Gradient (PCG) method. This solver has been found to be superior to a number of other public domain solvers [Qian and Sapatnekar 2008].

Next, the effectiveness of our solver on asymmetrical equations is examined using VCCS model of IBMPG benchmarks described in Appendix B. The reason for showing results on the original benchmarks is because this enables a comparison with the hybrid solver and the forward random walk solver, which cannot handle asymmetry. Moreover, as we will see, the runtime of the backward walk solver on the original and modified PG benchmarks is very similar. The experiments are conducted on a UNIX machine with 8GB RAM and 2.5GHz processor, where V_{dd} is set to 1.8V, as in the benchmarks.

6.1. Performance of the Backward Random Walk Solver on Symmetric LHS Matrices

The performance of the proposed incremental solver is demonstrated based on several metrics indicating the quality of the found RoI compared to *exact RoI* and precision of the solution of the nodes within the RoI. The *exact RoI*, found using a direct solver, is the set of nodes for which the exact solution is perturbed by more than a specified tolerance, 1% of the V_{dd} here. The metrics used for evaluating the quality of the computed RoI are:

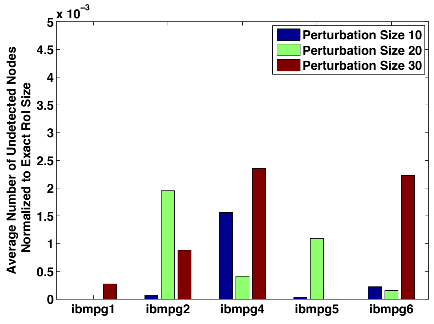
— The number of *undetected nodes*, i.e., the nodes that belong to the exact RoI, but are not listed in the RoI found by our method.
— *Error magnitude* in the solution of the undetected nodes.

A comparison is performed over several random perturbations of various *perturbation amounts*, i.e., the magnitudes of change relative to the solution of the node, and various *perturbation sizes*, i.e., the number of nodes perturbed. To generate a perturbed benchmark, one node is chosen randomly from the original benchmark and the conductances, g_{ij}'s, and the current loads, e_i's, of that node and a number of its neighboring nodes are modified. This approach models the locality property of a real perturbation.

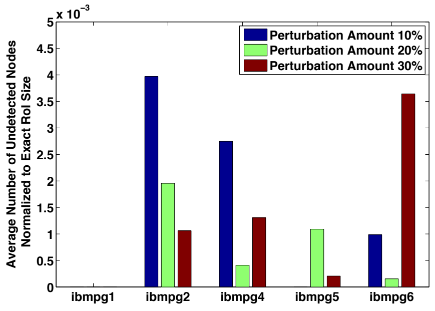
For the range of perturbations in our experiments, the maximum and the average size of the RoI is 17.1% and 2.9% of the total circuit size, respectively, which is significantly larger perturbations than the ones used in prior work [Boghrati and Sapatnekar 2010]. The empirically-chosen safety factor parameter in the algorithm is set to $1/3$ to compensate for approximation error and to generate a pessimistic RoI.

Figure 10(a) shows the number of undetected nodes, normalized to the exact RoI size, versus the size of the perturbed region, for perturbation amount of 20%, for various perturbation sizes. Similarly, the normalized number of undetected nodes versus the amount of perturbation, for perturbation size of 20, for various perturbation amounts, is shown in Figure 10(b). The numbers shown in all plots are averaged over 10 different sets of perturbations. These figures indicate that the number of undetected nodes is less than 0.5% the size of the corresponding exact RoI. Note that the backward solver has been able to identify all of the nodes within RoI for some of the perturbations.

The next issue is the significance of the undetected nodes. The errors caused by the undetected nodes are the errors of the estimated solution from random walks. The average of this error versus the perturbation size for a perturbation amount of 20% is plotted in Figure 11(a) and this error versus perturbation amount for a perturbation size of 20 is shown in Figure 11(b).

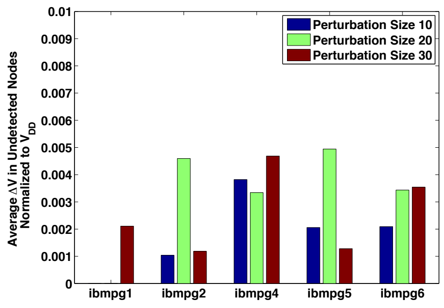


(a) Perturbation amount = 20%

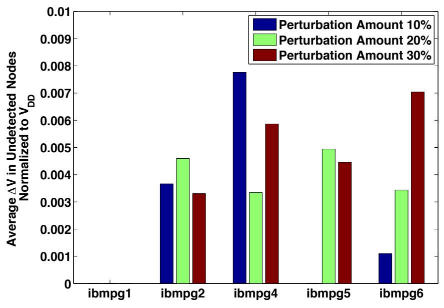


(b) Perturbation size = 20

Fig. 10. Number of undetected nodes, normalized to the exact RoI size (tolerance = $1\%V_{dd}$, averaged over 10 perturbations).



(a) Perturbation amount = 20%



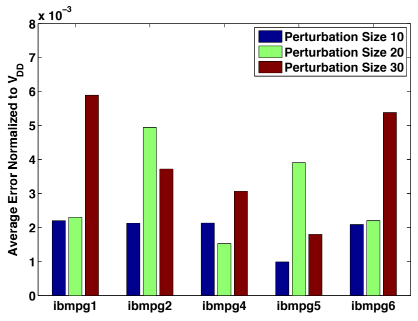
(b) Perturbation size = 20

Fig. 11. Average change in the voltage of undetected nodes (tolerance = $1\%V_{dd}$, averaged over 10 perturbations).

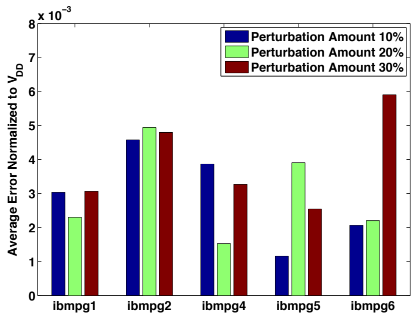
It is found that the average of this error is less than 1% of V_{dd}, indicating that even the nodes that lie within the RoI, but remain undetected, see an insignificant degradation in accuracy, and this is due to the fact that the estimated solution given by random walks is sufficiently accurate for the purposes of detecting the RoI.

Due to the stochastic nature of random walks, different runs result in slightly different estimated solutions and hence different computed RoIs. As discussed in Section 3.7, the differences remain less than the given tolerance (i.e., $1\%V_{dd}$), with a confidence of α. Due to this effect, in Figure 11(a) for instance, the error for perturbation size of 20 was more than that for a perturbation size of 30 for benchmark $ibmpg2$. For the similar reason, in Figure 11(b) perturbation amount of 20% resulted in less error than perturbation amount of 10% for $ibmpg4$. In Figure 10 for the same reason some counterintuitive behavior is observed.

In this paper we exclude the results of $ibmpg3$ because of the specific structure of the LHS of this benchmark (similar observations have been made by other authors). This benchmark models the power network of a circuit with very few external V_{dd} and ground connections, less than 0.1% of the nodes. As a result, the corresponding random walk game has very few home nodes, resulting in very long walks and hence large runtimes. For such circuits, conventional solvers should be used instead of random walk solvers. In general the number and distribution of the power supply connections affects the performance of the random walks. The best scenario for random walk solver is frequent and uniform distribution of the power supply connections.

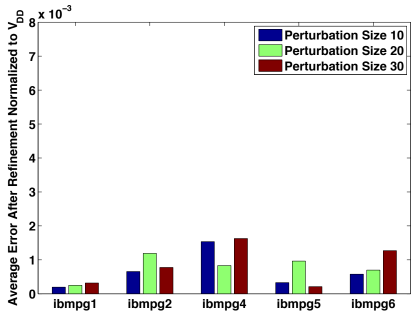


(a) Perturbation amount = 20%

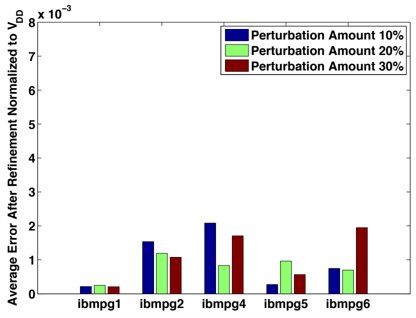


(b) Perturbation size = 20

Fig. 12. Absolute error of nodes within the RoI *before* the refinement phase, normalized to V_{dd} and averaged over 10 perturbations. (tolerance = 1% of V_{dd}).



(a) Perturbation amount = 20%



(b) Perturbation size = 20

Fig. 13. Absolute error of nodes within the RoI *after* the refinement phase, normalized to V_{dd} and averaged over 10 perturbations. (tolerance = 1% of V_{dd}).

Next, we examine the accuracy of the solution within the RoI. Figure 12 shows the average error of the nodes within the RoI normalized to V_{dd}, for different perturbation sizes and perturbation amounts. This figure suggests that although the relative tolerance of the random walk solver is set to 30%, the average error of the estimated solution is less than 0.7% of V_{dd}. Feeding the solution to the refinement stage, where we solve a much smaller system of size $|\text{RoI}| \times |\text{RoI}|$, the solution becomes more accurate; this can be seen by comparing the results in Figures 12 and 13.

Table III compares the runtime of our proposed incremental solver with the Hybrid Solver of [Qian and Sapatnekar 2005], which is an efficient public-domain iterative solver that uses a preconditioner based on random walks that has been shown to be faster than other comparable solvers, using identical solver tolerances. The first column shows N, the matrix dimension for each benchmark. The second column shows the runtime of the Hybrid Solver and the remaining columns are related to our incremental solver. It can be seen that the refinement phase is extremely fast and takes only a small fraction of the total runtime, and that the total speed up of the proposed

Table III. Runtimes (tolerance = 1% of V_{dd}, perturbation region size = 30, perturbation amount = 20%).

	Equation Size (N)	Hybrid (sec)	Random Walk (sec)	Refinement (sec)	Incremental Total (sec) [Speedup]
ibmpg1	30,638	0.17	0.111	0.001	0.112 [1.5×]
ibmpg2	127,238	2.14	0.743	0.058	0.801 [2.7×]
ibmpg4	953,583	26.37	1.673	0.274	1.947 [13.5 ×]
ibmpg5	1,079,310	18.74	2.108	0.122	2.230 [8.4×]
ibmpg6	1,670,494	37.23	2.898	0.200	3.098 [12.0 ×]
				Average Speedup:	7.6×

incremental solver for perturbation size of 30 and perturbation amount of 20% is significant: an average of 7.6× and a maximum of 13.5×.

Moreover, broadly speaking, as the system size increases the benefit of using the incremental solver becomes more significant. The intuitive reason for this is that for benchmarks of similar topology, a perturbation of the same size and amount results in a RoI of almost the same size, which requires almost the same amount of effort for the random walk solver to find the RoI and the exact solver to refine the solution. In fact, the speedup depends on the structure of the equation as well, i.e., its density, condition number, and the number of home nodes in its corresponding random walk game.

6.2. Backward Solver on Asymmetric LHS Matrices

As discussed in Section 2, the use of VCCS in modeling power grids makes it possible to account for the effect of supply voltage drop on the current load which results in a more accurate power grid model but equations with *asymmetrical* LHS. In this section we demonstrate that the backward solver is capable of handling these asymmetrical equations as efficient as the symmetrical equations while the forward solver of [Boghrati and Sapatnekar 2010] is capable of solving the symmetrical equations only.

We have created variations of the ibmpg benchmarks [Nassif 2008] using the procedure of Section 2.2 where the detailed extracted circuit of the benchmarks are obtained as described in Appendix B. These benchmarks are referred to as the **ibmpg'** and the circuit with each number represent the corresponding circuit with the same number in the original benchmark suite.

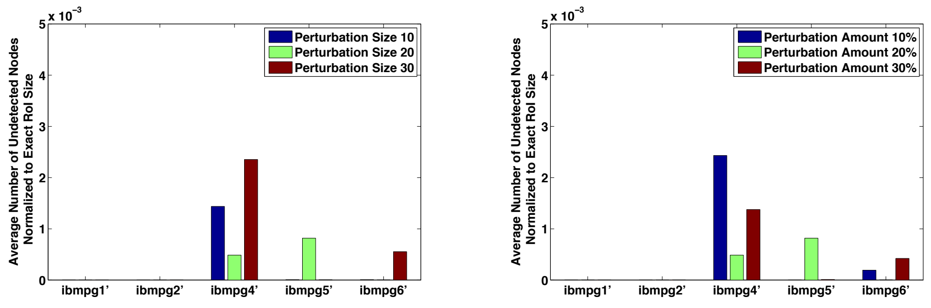
Figure 14 is analogous to Figure 10 and shows the average number of undetected nodes, normalized to the exact RoI size, versus the size of the perturbed region, and amount of perturbation. Comparing these figures, it can be seen that the number of undetected nodes is less than 0.5% the size of the corresponding exact RoI for both symmetrical and asymmetrical power grid models.

The performance of the backward solver in terms of error and runtime is shown in Table IV for symmetrical and asymmetrical power grid models. This table shows the absolute error of the solver after the refinement phase normalized to V_{dd} averaged over 10 random perturbations of region size of 30 and amount of 20% and their corresponding runtime. It is clear from this table that the backward solver performs equally well for both models.

6.3. Comparing Forward/Backward Solver Based Incremental Analysis

Finally in this section we compare the forward incremental solver of [Boghrati and Sapatnekar 2010] with the proposed backward solver of this work over the *ibmpg* benchmarks. The work of [Boghrati and Sapatnekar 2012] compares these solvers using the benchmarks of [Boghrati and Sapatnekar 2010].

Table V compares the forward and backward solver accuracy in terms of the quality of the computed RoI and error in the solution of the nodes within the RoI before and after refinement phase. The number of undetected nodes is normalized to the RoI



(a) Perturbation amount = 20% (b) Perturbation size = 20

Fig. 14. Number of undetected nodes, normalized to the exact RoI size (tolerance = $1\%V_{dd}$, averaged over 10 perturbations).

Table IV. Comparison of the error of the nodes within RoI *after* refinement phase and total runtime of backward solver on symmetrical and asymmetrical equations (Averaged over 10 random perturbations. perturbation region size = 30, perturbation amount = 20%, error is normalized to V_{dd}, tolerance = 1% of V_{dd}).

	Symmetrical model			Asymmetrical model	
	Error (Norm to V_{dd})	Runtime (sec)		Error (Norm to V_{dd})	Runtime (sec)
ibmpg1	3.18E-04	0.112	**ibmpg1'**	1.11E-04	0.111
ibmpg2	7.77E-04	0.801	**ibmpg2'**	8.62E-05	0.740
ibmpg4	1.63E-03	1.947	**ibmpg4'**	1.56E-03	1.673
ibmpg5	2.11E-04	2.230	**ibmpg5'**	1.50E-04	2.111
ibmpg6	1.27E-03	3.098	**ibmpg6'**	5.22E-04	2.897
Average	8.41E-04	1.638	**Average**	4.86E-04	1.506

Table V. Comparison of the accuracy of the Forward solver and Backward solver (Averaged over 10 random perturbations. perturbation amount = 5%, tolerance = 1% of V_{dd}).

	Forward Incremental Solver				Backward Incremental Solver			
Bench-mark/ Pert. size	Norm. Undet. Nodes	Norm. ΔV in Undet.	Error Norm. to V_{dd} Before Refined	After Refined	Norm. Undet. Nodes	Norm. ΔV in Undet.	Error Norm. to V_{dd} Before Refined [$\times$Less]	After Refined [$\times$Less]
ibmpg1/3	0.04	5.4E-03	1.1E-03	2.8E-04	0	0	2.1E-04[5.03$\times$]	1.5E-05[19.40$\times$]
ibmpg2/3	0.02	2.4E-03	1.5E-03	5.6E-04	0	0	2.2E-04[6.53$\times$]	1.4E-04[3.92$\times$]
ibmpg1/5	0.02	2.9E-03	1.3E-03	7.7E-05	0	0	1.0E-04[12.76$\times$]	9.1E-06[8.49$\times$]
ibmpg2/5	0.04	6.0E-03	5.4E-03	6.08E-03	0.001	2.25E-04	5.3E-04[10.17$\times$]	3.3E-04[18.33$\times$]
ibmpg1/7	0.06	1.2E-02	7.2E-03	2.5E-03	0	0	9.6E-04[7.52$\times$]	1.7E-04[14.19$\times$]
ibmpg2/7	0.10	1.5E-02	1.3E-02	6.7E-03	0.004	3.78E-04	1.0E-03[12.42$\times$]	3.1E-04[21.41$\times$]
Average	**0.05**	**7.4E-03**	**4.9E-03**	**2.7E-03**	**0.001**	**1.01E-04**	**5.1E-04[9.07$\times$]**	**1.6E-04[14.29$\times$]**

size, and ΔV is normalized to V_{dd}. As this table suggests, on average, the number of undetected nodes of the backward solver is about 50$\times$ less than that of the forward solver and also the error in the undetected nodes is about 70$\times$ less. Moreover this tables shows that the error in the solution of the nodes within RoI, computed by backward solver, before the refinement phase is up to 12$\times$ and on average 9$\times$, and after refinement phase up to 21$\times$ and on average 14$\times$ less than the forward solver.

In this section, comparisons are shown only for the *ibmpg1* and *ibmpg2* benchmarks since the forward solver implementation is unable to handle large benchmarks. Moreover, smaller perturbations are used here: due to recursive nature of the forward solver method, the forward solver cannot handle the large perturbations used in Section 6.1. The key difference between the backward and forward solver that makes the backward solver significantly more efficient is that for obtaining the RoI, the amount of effort of backward solver is proportional to the perturbation size, while this amount for forward

Table VI. Comparison of the runtime of the Forward solver and Backward solver (Averaged over 10 random perturbations. perturbation amount = 5%, tolerance = 1% of V_{dd}).

Bench-mark/ Pert. size	Forward Incremental Solver Runtime (sec)						Backward Incremental Solver Runtime (sec)				
	Book-keeping	RoI	Refine	1 run	10 runs	30 runs	RoI	Refine	1 run [Speedup]	10 runs [Speedup]	30 runs [Speedup]
ibmpg1/3	0.39	0.01	1.7E-04	0.40	0.47	0.62	0.02	2.6E-04	0.02[18.7×]	0.21[2.2×]	0.64[1.0×]
ibmpg2/3	2.48	0.06	1.2E-03	2.54	3.09	4.31	0.10	1.3E-03	0.10[24.6×]	1.03[3.0×]	3.10[1.4×]
ibmpg1/5	0.39	0.01	1.6E-04	0.40	0.49	0.70	0.03	1.9E-04	0.03[12.1×]	0.33[1.5×]	1.00[0.7×]
ibmpg2/5	2.48	0.07	1.1E-03	2.55	3.14	4.46	0.15	3.6E-03	0.15[16.9×]	1.51[2.1×]	4.52[1.0×]
ibmpg1/7	0.39	0.01	1.5E-04	0.40	0.49	0.70	0.03	3.4E-04	0.03[13.2×]	0.30[1.6×]	0.91[0.8×]
ibmpg2/7	2.48	0.07	1.1E-03	2.55	3.14	4.46	0.21	4.7E-03	0.21[12.1×]	2.11[1.5×]	6.32[0.7×]
								Average Speedup	**16.28**×	**1.98**×	**0.92**×

solver is proportional to RoI size, which is significantly larger than perturbation size (see Equation (31)).

Table VI shows the runtime breakdown for forward solver and the backward solver. The forward solver uses three steps in creating an incremental solution, *Bookkeeping*, *RoI computation*, and *Refinement*. The bookkeeping step is the initialization step and is performed only once for a series of consecutive perturbations to a circuit. Therefore, the overhead of this initialization step is amortized for many consecutive perturbations. Hence, in this table the runtime of the forward and backward solver is shown for different number of consecutive perturbations.

Table VI shows that the backward solver is up to 24×, and on average, 16× faster for a single run. For 10 runs (i.e., 10 consecutive perturbations) the backward solver is up to 3×, and on average, 2× faster. For 30 runs the runtime of the backward solver and forward solver are about the same. In general as the number of runs increase, the runtime benefit of backward solver compared to [Boghrati and Sapatnekar 2010] decreases. However, the solution from [Boghrati and Sapatnekar 2010] solver loses its accuracy as errors from consecutive perturbations accumulate (the bookkeeping information collected becomes less accurate); on the other hand, the backward incremental solver retains its accuracy regardless of the number of perturbations. In addition, as pointed out earlier, the backward solver is scalable to larger problems.

7. CONCLUSION

In this paper, we have developed a method for efficient incremental power grid analysis based on a computationally efficient backward random walk method. The backward walk approach is particularly useful in computing the impact of incremental changes since it can efficiently compute the inverse of any individual column of the LHS matrix. This approach is approximate and is used to determine a RoI around the area of the perturbation where the voltages are significantly impacted. Next, an exact direct solver may be employed to determine the precise solution by solving for voltages within the RoI while assuming that voltages outside the RoI are unchanged or at their approximate levels, as given by the random walk solver.

Acknowledgment

This work was supported in part by the NSF under award CCR-0634802.

A. PROOF OF LEMMA 2.1

For the circuits in Figure 5(a) and 5(b) to be equivalent, as seen from nodes a and b, I_a and I_b in both circuits must be equal for any V_a and V_b.

$$\begin{aligned}
I_x &= I_x^0 + g_{xa}v_a + g_{xb}v_b + g_x v_x \\
I_y &= I_y^0 + g_{ya}v_a + g_{yb}v_b \\
I_z &= I_z^0 + g_{za}v_a + g_{zb}v_b \\
g_{ab} &= \frac{g_a g_b}{g_a + g_b}
\end{aligned}$$

For the T-model of Figure 5(a), by KCL, the sum of all currents entering nodes x, a, and b are zero according to KCL. Incorporating the device equations and KVL, this leads to the relations:

$$\begin{aligned}
v_x &= \frac{v_a(g_a - g_{xa}) + v_b(g_b - g_{xb}) - I_x^0}{g_a + g_b + g_x} \\
I_a &= g_a(v_a - v_x) \\
I_b &= g_b(v_b - v_x)
\end{aligned}$$

where v_a and v_b are the voltage of nodes a and b respectively.

Eliminating v_x and defining $\hat{g}_x = g_a + g_b + g_x$ we get :

$$\begin{aligned}
I_a &= g_a \left[\left(1 - \frac{g_a - g_{xa}}{\hat{g}_x}\right) v_a - \left(\frac{g_b - g_{xb}}{\hat{g}_x}\right) v_b + \frac{I_x^0}{\hat{g}_x} \right] \\
I_b &= g_b \left[-\left(\frac{g_a - g_{xa}}{\hat{g}_x}\right) v_a + \left(1 - \frac{g_b - g_{xb}}{\hat{g}_x}\right) v_b + \frac{I_x^0}{\hat{g}_x} \right]
\end{aligned} \tag{41}$$

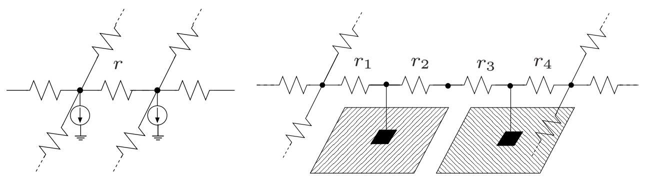
For the Π-model in Figure 5(b), similarly applying KCL, KVL, and the device equations at nodes a and b, we have:

$$\begin{aligned}
I_a &= (g_{ya} + g_{ab})v_a + (g_{yb} - g_{ab})v_b + I_y^0 \\
I_b &= (g_{za} - g_{ab})v_a + (g_{zb} + g_{ab})v_b + I_z^0
\end{aligned} \tag{42}$$

For the T-model and Π-model to be equivalent, Equation (41) and Equation (42) must be equivalent for any I_a, I_b, v_a, and v_b. Therefore, the corresponding coefficients of these equations must be equal. Equating these coefficients completes the proof.

B. MODIFYING EXISTING BENCHMARKS TO ADD MORE DETAIL

In this section, we describe how our test circuits showing a detailed extracted power grid were generated from an existing standard benchmark suite that is based on a lumped current source approximation [Nassif 2008].



(a) Extracted power grid (b) Detail extracted power grid

Fig. 15. Splitting a wire piece of extracted circuit to model it as a detailed extracted circuit, $r = r_1 + r_2 + r_3 + r_4$.

The format of the original benchmark circuits is similar to Figure 4, with lumped current sources at the intersections of long wire segments. Our modification that generates the detailed extracted circuit distributes the current sources at the power line intersections to specific locations on the power lines. To this end, the power line between any two nodes is first split into multiple pieces. As an example, Figure 15 shows a wire being split as if there were two cells laid out between these power grid nodes.

While it is possible to perform this split at very fine levels of granularity, there will be diminishing returns in accuracy beyond a point. In this work, our goal is to motivate the notion that the lumped approximation induces errors and to generate realistic power grids with asymmetric LHS matrices that could be exercised with the backward random walk solver. Therefore, for simplicity, we have assumed that there are two equally spaced cells connected to the grid between two adjacent nodes. Then, a portion of the current source at the power grid intersection is associated with each of the cells adjacent to it such that the total current is kept the same. Here the current load of the intersection current sources is distributed uniformly among the adjacent cells, effectively attempting to reverse-engineer the original process in which the cells were lumped together.

Once the cell currents are determined, each cell is modeled as a simple VCCS controlled by the voltage of the node connecting that cell to the power network, i.e., an independent current source with a parallel resistor. The parameters of this VCCS are then found with the aid of a *unit cell* for which these parameters are calibrated. A unit cell, as explained shortly, represents an average cell used in VLSI circuits. This unit cell is scaled such that its current draw at V_{dd} (or zero for GND network) is equal to the current draw of the cell it is replacing. The circuit model obtained with this procedure is the detailed extracted circuit of the power grid.

Next we look at the process of characterizing the unit cell. To this end, we determine:

— the statistics of cell usage in a typical set of circuits (here, we use the MCNC benchmark suite to obtain these statistics and a NANGate library [Nangate]).
— the current load model of each cell (an independent current source in parallel with a resistor), computed for the cell by measuring the variation in the current draw when V_{dd} is swept from 90% to 100% using small-signal SPICE simulations, and fitting a line on the I-V characteristic.

Given the cell usage statistics, the unit cell is obtained by computing the expected value of the current source and conductance of the load model of the cells, averaged over the distribution of cells.

C. PROOF OF THEOREM 3.1

Figure 16 shows two intersection nodes a and b of a detailed model of a power grid with K intermediate nodes (see nodes 1 and 2 of Figure 3 where $K = 5$). In this figure connected cells are modeled as an independent current source in parallel with a resistor. Note that the simplified extracted power grid model is a special case of this general model in which all the cell conductances are zero and the current sources are lumped together. The MNA stamp of the coarse asymmetric VCCS model of this circuit block is:

$$\begin{bmatrix} I_a \\ I_b \end{bmatrix} = \begin{bmatrix} g_{aa} & g_{ab} \\ g_{ba} & g_{bb} \end{bmatrix} \begin{bmatrix} v_a \\ v_b \end{bmatrix} + \begin{bmatrix} I_a^0 \\ I_b^0 \end{bmatrix} \quad (43)$$

Note that this stamp formulates the electrical characteristics of this circuit block as seen from nodes a and b. Lemma 2.1 utilized the same electrical characteristics to show two circuit blocks are equivalent.

The LHS of the Equation (1) is made of the superposition of the conductance matrix of this stamp for all the intersection nodes. Therefore, to show the proposed properties of this theorem, it is enough to show that this stamp has these properties, mathemat-

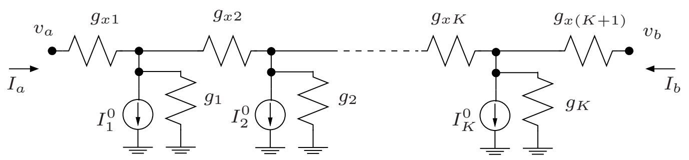


Fig. 16. Power grid model for two adjacent intersection nodes and the intermediate nodes in between

ically:

$$g_{ba} \leq 0 \tag{44}$$
$$g_{aa} > 0 \tag{45}$$
$$|g_{aa}| \geq |g_{ba}| \tag{46}$$
$$g_{ab} \leq 0 \tag{47}$$
$$g_{bb} > 0 \tag{48}$$
$$|g_{bb}| \geq |g_{ab}| \tag{49}$$

By linearity of this circuit, g_{aa} and g_{ba} can be written as:

$$\begin{aligned} g_{aa} &= I_a/v_a \\ g_{ba} &= I_b/v_b \\ I_k^0 &= 0,\ \forall k \in [a,b] \\ v_b &= 0 \end{aligned} \tag{50}$$

Figure 17 shows the equivalent circuit of Figure 16 in which $I_k^0 = 0,\ \forall k$ and $v_b = 0$. In this figure for $v_a > 0$, it is obvious that $I_a > 0$ and $I_b < 0$ therefore $g_{aa} > 0$ and $g_{ba} < 0$. Also if there was no resistive path between a and b, then $I_b = 0$. Hence Equations (44) and (45) hold.

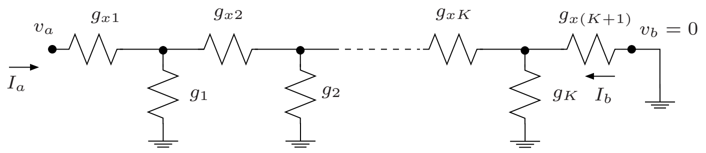


Fig. 17. Power grid model of 16 with $I_k^0 = 0\ \forall k$ and $v_b = 0$

To show Equation (46), based on Equation (50), it is enough to show that in the circuit of Figure 17, for a fixed v_a we have:

$$|I_a| \geq |I_b| \tag{51}$$

Based on KCL, the equality holds if $g_k = 0,\ \forall k$ and otherwise we have $|I_a| > |I_b|$. Also if there was no resistive path between a and b, $I_a = I_b = 0$ and as a result $g_{aa} = g_{ba} = 0$. Therefore Equation (46) holds.

Te same argument shows that Equations (47), (48), and (49) hold if we set $I_k^0 = 0,\ \forall k$ and $v_a = 0$.

Finally, since every node in the power grid is connected to at least one other node in the grid, there is one non-zero stamp associated with every node and therefore, all diagonal entries of the LHS are greater than zero. Hence, Equations (45) and (48) hold for every node in the grid. □

REFERENCES

ANDERSON, E., BAI, Z., BISCHOF, C., BLACKFORD, S., DEMMEL, J., DONGARRA, J., DU CROZ, J., GREENBAUM, A., HAMMARLING, S., MCKENNEY, A., AND SORENSEN, D. 1999. *LAPACK Users' Guide* Third Ed. Society for Industrial and Applied Mathematics, Philadelphia, PA.

BOGHRATI, B. AND SAPATNEKAR, S. 2010. Incremental solution of power grids using random walks. In *Proc. ASPDAC*. 757–762.

BOGHRATI, B. AND SAPATNEKAR, S. 2012. Incremental power network analysis using backward random walks. In *Proc. ASPDAC*. 41–46.

CASTRO, F., SBERT, M., AND HALTON, J. H. 2008. Efficient reuse of paths for random walk radiosity. *Computers & Graphics 32,* 1, 65–81.

CHIPROUT, E. 2004. Fast flip-chip power grid analysis via locality and grid shells. In *Proc. ICCAD*. 485–488.

FU, Y. ET AL. 2007. A novel technique for incremental analysis of on-chip power distribution networks. In *Proc. ICCAD*. 817–823.

GOLUB, G. H. AND VAN LOAN, C. F. 1996. *Matrix Computations* 3rd Ed. Johns Hopkins University Press, Baltimore, MD.

GUO, W., TAN, S., LUO, Z., AND HONG, X. 2004. Partial random walk for large linear network analysis. In *Proc. ISCAS*. Vol. 5.

HAMMERSLEY, J. M. AND HANDSCOMB, D. C. 1964. *Monte Carlo Methods*. Taylor & Francis.

MIYAKAWA, T., YAMANAGA, K., TSUTSUI, H., OCHI, H., AND SATO, T. 2011. Acceleration of random-walk-based linear circuit analysis using importance sampling. In *Proc. GLSVLSI*. 211–216.

NANGATE, I. NANGate. `http://www.si2.org/openeda.si2.org/projects/nangatelib`.

NASSIF, S. R. 2008. Power grid analysis benchmarks. In *Proc. ASPDAC*. 376–381.

PILLAGE, L. T., ROHRER, R. A., AND VISWESWARIAH, C. 1994. *Electronic Circuit and System Simulation Methods*. McGraw-Hill, New York, NY.

QIAN, H., NASSIF, S. R., AND SAPATNEKAR, S. S. 2003. Random walks in a supply network. In *Proc. DAC*. 93–98.

QIAN, H., NASSIF, S. R., AND SAPATNEKAR, S. S. 2005. Power grid analysis using random walks. *IEEE TCAD 24,* 8, 1204–1224.

QIAN, H. AND SAPATNEKAR, S. S. 2005. A hybrid linear equation solver and its application in quadratic placement. In *Proc. ICCAD*. 905–909.

QIAN, H. AND SAPATNEKAR, S. S. 2008. Stochastic preconditioning for diagonally dominant matrices. *SIAM Journal on Scientific Computing 30,* 3, 1178–1204.

SAAD, Y. 2003. *Iterative Methods for Sparse Linear Systems* Second Ed. Society for Industrial and Applied Mathematics, Philadelphia, PA.

SINGH, J. AND SAPATNEKAR, S. S. 2004. Topology optimization of structured power/ground networks. 116–123.

WANG, J. 2012. Deterministic random walk preconditioning for power grid analysis. In *Proc. ICCAD*.

YE, Y., ZHU, Z., AND PHILIPS, J. R. 2008. Generalized Krylov recycling methods for solution of multiple related linear equation systems in electromagnetic analysis. In *Proc. DAC*. 682–687.

ZHAO, M. ET AL. 2000. Hierarchical analysis of power distribution networks. In *Proc. DAC*. 150–155.