Incremental Power Network Analysis Using Backward Random Walks

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

***Abstract*—The process of power network analysis during VLSI chip design is inherently iterative. It is very common for the designer to make many small perturbations to an otherwise complete design, to enhance the design or fix design violations. Considering the size of the modern chips, updating the solution for the changed network can be a computationally intensive task. In this paper 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 updated for the significantly smaller region only. The proposed algorithm is capable of handling consecutive perturbations without any degradation. The experimental results show speedups of up to 13.7× as compared to a complete solution.**

I. Introduction

Many problems in VLSI design and in other fields involve the solution of a system of linear equations where the left hand side has the form of diagonally dominant matrix with positive diagonal and nonpositive off-diagonal entries. Examples include supply network analysis, temperature analysis under the resistive-electrical duality, and quadratic placement. This paper addresses the problem of incremental power grid analysis; incremental versions of the other problems may also be solved in similar ways.

The equations that represent the power grid are given by:

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

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. 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 [1]–[4].

We focus on the problem of incremental analysis of the power network for the steady-state case. 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 [5].

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 [6]. However, this may require a large number of partitions, involving large amounts of computation. Other approaches employ iterative solvers [7] using the unperturbed solution as an initial guess, sensitivity methods [8], and the fictitious domain method [9]. 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. An iterative solver is also described in [10], but operates on smaller, denser systems.

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

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 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. Fig. 1 illustrates the dimension of LHS matrix corresponding to the full network and the much smaller subsystem corresponding to the RoI of a perturbation.

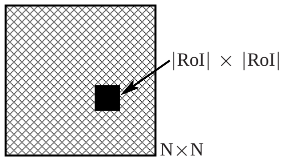


Fig. 1. Reduction in system size using RoIs.

Identifying the RoI is nontrivial. The work of [11] uses the bookkeeping information of the random walk solver of [1] (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 [12], 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 [13]. 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 solutions from that domain cannot be adapted easily.

In this work, we develop the idea of backward random walks 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. Experimental results show speedups of up to 13.7× compared to the Hybrid Solver of [14] on IBM power grid benchmarks of [15]. The key feature of the backward random walk approach is the ability to find some columns of G^{-1} efficiently, individually, without finding the entire G^{-1}.

The paper is organized as follows. The backward random walk method is discussed in Section II. Next, Section III discusses the proposed incremental solver, after which the relation between the backward random walks with LU decomposition of the LHS is stated in Section IV. Finally, Section V presents the experimental results.

II. Background

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

The solution to Eq. (1) is $\mathbf{V} = G^{-1}\mathbf{E}$, and can be written as:

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

where $(G^{-1})_{ij}$ is the $(i,j)^{\text{th}}$ element of G^{-1}, v_i and e_i are the i^{th} elements of $\mathbf{V}$ and $\mathbf{E}$, respectively, and N is the dimension of G.

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. Fig. 2 shows voltages of the nodes in the network due to just one of the current loads. As this figure suggests, there is a close relation between finding the RoI and Eq. (2).

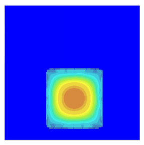


Fig. 2. Contribution of a single current load on the power network solution.

B. The Backward Random Walk Game

In this section, we qualitatively review the similarities and differences of backward random walks of Section II.A and the forward random walks described in [1]. The forward game captures the effect of all of the RHS elements (i.e., the current loads in power network analysis context) on the solution of a single node, while the backward game captures the effect of a single source on the solution of the entire system, as indicated by Eq. (2). For the case where there are few nonzero elements on the RHS, only a few corresponding columns of G^{-1} must be computed, and backward random walks are extremely appealing. As we will show, this arises during incremental analysis, i.e., in finding the effect of a small perturbation on the solution of the entire system.

The forward game is suitable for the case that the full solution of a subset of nodes in the network is desired (e.g., voltages of a subset of nodes in the power grid), while the backward game is good at finding the effect of individual sources on the entire system (e.g., the effect of a current load on the node voltages of the entire grid).

The key difference is that the forward game is constructed based on the rows of G, while the backward game is constructed based on the columns of G. Therefore, the forward game finds the rows of the matrix inverse, G^{-1}, and the backward game finds the columns of this matrix. In mathematical terms a backward game is based on Eq. (2) while the forward game is based on:

$$v_i = \begin{bmatrix} (G^{-1})_{i1} & (G^{-1})_{i2} & \ldots & (G^{-1})_{iN} \end{bmatrix} \mathbf{E} \quad (3)$$

where $i = 1, \ldots, N$ and the parameters are as defined before.

As described later in Section II.C, the construction of the 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 the analogy is that the walker has to pay for each motel he visits and gets an award as he reaches a home. In contrast, the analogy for the backward game is the walker has a sum of money (similar to home award in forward game) that he has to distribute among the motels he visits based on how frequently he visits them. These analogies are then translated to the rows and columns of G^{-1} for the forward and backward games, respectively.

C. Constructing the Backward Walk Game

We now describe how the backward random walk game is constructed from Eq. (1), and how the columns of G^{-1} are computed using this approach. This game is based on a network of roads with motels or homes at its intersections. The walker starts from some intersection, j, 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 $i = 1, \ldots, \text{degree}(j)$, and $\text{degree}(j)$ 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 on the path is called *walk length*. The goal of this game is to enumerate the visits to each motel during a walk.

Without loss of generality it can be assumed that for the network,

$$\sum_{j=1, j\neq i}^{N} |g_{ij}| = g_{ii}, \ \forall i \quad (4)$$

Note that this is always the case for supply network equations if the connections to the ground node are included in the equation. In general, this can be assured by adding a dummy variable whose value equals the supply node voltage (V_{dd} or zero) to the $\mathbf{V}$ vector and a new column to the LHS of the Eq. (1), G. For the ground network, for each row of the G we have:

$$\begin{aligned} \sum_{j=1}^{N} g_{ij} v_j &= \sum_{j=1}^{N} g_{ij} v_j + g_{i(N+1)} \times 0, \ \forall i \\ g_{i(N+1)} &= g_{ii} - \sum_{j \neq i} |g_{ij}| \end{aligned} \quad (5)$$

where $\sum_{j=1}^{N+1} g_{ij} = 0$ (for the supply network, if this transform is applied, the excitation vector $\mathbf{E}$ must be appropriately adjusted). The insertion of the dummy nodes is essential to obtain valid road probability distributions discussed in Theorem 1.

Theorem 1 *For each row of a system of linear equations defined by Eq.* (1)*, a valid probability mass distribution can be defined as:*

$$p_{ji} = \begin{cases} -g_{ij}/g_{ii} & j \neq i \\ 0 & j = i \end{cases} \quad (6)$$

Proof: Based on the assumption of Eq. (4) and since all off-diagonals of G are less than or equal to zero, for $i, j = 1, \ldots, N$, we have:

$$0 \leq p_{ji} \leq 1$$
$$\sum_{i=1}^{N} p_{ji} = 1$$

Note that in Eq. (6), $p_{ji} = 0$ for many of the i's due to the sparsity of G. Overloading subscript i to show only the nonzeros, we have:

$$\sum_{i=1}^{\text{degree}(j)} p_{ji} = 1, \ j = 1, \ldots, N \quad (7)$$

where degree(j) denotes the number of nonzeros in the j^{th} equation. Writing p_{ji}'s in matrix form, P, we get:

$$G = (I - P)^T D \quad (8)$$

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

A random walk game can be constructed from Eq. (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 Eq. (6). Note that the vector $\mathbf{V}$ consists of both unknowns as well as known variables that are added, as shown in Eq. (4). The unknown and known variables of vector $\mathbf{V}$, are mapped to motels and homes, respectively.

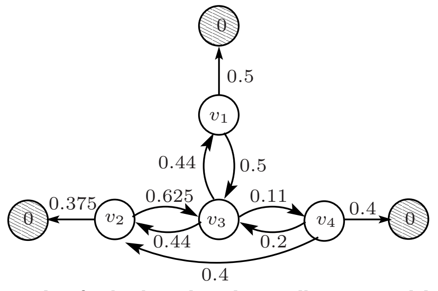


Fig. 3. An example of a *backward* random walk game modeling Eq. (9).

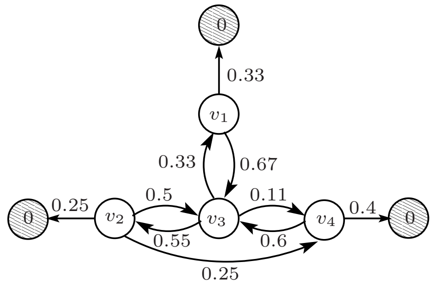


Fig. 4. An example of a *forward* random walk game modeling Eq. (9).

D. Example of Backward and Forward Game Construction

Consider the following diagonally dominant matrix:

$$G = \begin{bmatrix} 1.5 & 0 & -1 & 0 \\ 0 & 2 & -1 & -0.5 \\ -0.75 & -1.25 & 2.25 & -0.25 \\ 0 & 0 & -0.25 & 1.25 \end{bmatrix} \quad (9)$$

To construct the backward game corresponding to this matrix, the *columns* of G should be normalized to its diagonals, such that it can be written in the form of Eq. (8), a row-wise probability matrix (corresponding to the transpose of the normalized G). Fig. 3 shows the random walk game, with the probability matrix:

$$P_{backward} = \left[\begin{array}{cccc|c} 0 & 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0.625 & 0 & 0.375 \\ 0.44 & 0.44 & 0 & 0.11 & 0 \\ 0 & 0.4 & 0.2 & 0 & 0.2 \end{array}\right] \quad (10)$$

Note that in this equation the last column corresponds to the ground node, to generate all probabilities as in Eq. (4).

Similarly the forward random walk game is constructed based on probability matrix of Eq. (11) which is computed by normalizing to *rows* of matrix G [16]. Fig. 4 shows the corresponding game.

$$P_{forward} = \left[\begin{array}{cccc|c} 0 & 0 & 0.67 & 0 & 0.33 \\ 0 & 0 & 0.5 & 0.25 & 0.5 \\ 0.33 & 0.55 & 0 & 0.11 & 0 \\ 0 & 0 & 0.6 & 0 & 0.4 \end{array}\right] \quad (11)$$

E. 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:

$$z_{ij} = \frac{\text{Number of visits to motel } i \text{ in } M \text{ walks from } j}{M} \quad (12)$$

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 $M \to \infty$.

The backward random walk game, as described in Section II.A, can solve Eq. (1) using the probability matrix P of Eq. (6). The solution is given by:

$$\mathbf{V} \quad = \quad D^{-1}\left(I-P\right)^{-T}\mathbf{E} = D^{-1}Z\mathbf{E}$$

where D is the matrix of the diagonals of G: the multiplication by D^{-1} scales the inverse of the normalized LHS (i.e., $(I-P)^T$) back to the inverse of the original LHS, G. We define:

$$\begin{aligned} \mathbf{X} &= D\mathbf{V} = \left(I-P\right)^{-T}\mathbf{E} = Z\mathbf{E} \\ Z &= \left[z_{ij}\right]_{N\times N} \end{aligned} \quad (13)$$

where the z_{ij}'s are defined by Eq. (12) [12].

Thus, the j^{th} column of G^{-1} can be found by:

$$(G^{-1})_{ij} = \frac{z_{ij}}{g_{ii}},\ i = 1, \ldots, N \quad (14)$$

As before, $(G^{-1})_{ij}$ is the element in (row i, column j) of G^{-1}.

Table I shows the second column of the exact G^{-1}, denoted by $(G^{-1})_{*2}$, of the example of Fig. 3, and the average relative error, in percent, of the estimation using backward random walks method, where 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. 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 I
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%

The number of walks for a desired accuracy can be determined dynamically in a manner similar to that derived in [16]. It can be shown that for a given relative error of δ, the number of walks needed for the relative error of random walk method to be less than or equal to δ, with confidence of α (e.g., 99%), is given by:

$$\frac{(\sigma/\mu)^2}{M} < \left(\frac{\delta}{Q^{-1}\left(\frac{1-\alpha}{2}\right)}\right)^2 \quad (15)$$

where Q^{-1} is the inverse of $Q(x) = 1/\sqrt{2\pi}\int_x^\infty e^{-u^2/2}du$ and μ and σ are the mean and standard deviation of the walk length.

III. 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 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:

$$(G + \Delta G)(\mathbf{V} + \Delta\mathbf{V}) = \mathbf{E} + \Delta\mathbf{E} \quad (16)$$

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:

$$\begin{aligned}(G+\Delta G)\Delta \mathbf{V} &= \Delta \mathbf{E}-\Delta G\mathbf{V}\\ G_{\text{eff}}\Delta \mathbf{V} &= \Delta \mathbf{E}_{\text{eff}}\end{aligned} \quad (17)$$

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 [11], where the second order term, $\Delta G\Delta \mathbf{V}$, is ignored, Eq. (17) *captures any perturbation to the system without any approximation.* Moreover, it does not make any assumptions regarding the nature of the perturbation as long as the number of nodes of the network is fixed. Note further that since the perturbed system models a power grid, it 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 Eq. (17) using backward random walks.
Step 2: Find the RoI using the computed solution.
Step 3: Refine the solution for the nodes within RoI.
We now describe each of these steps in greater detail.

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. As a result, *only a few columns* of G_{eff}^{-1} must be computed, corresponding to nonzeros of $\Delta \mathbf{E}_{\text{eff}}$. Then, $\Delta \mathbf{V}$ is given by:

$$\Delta \mathbf{V}=\left[G_{\text{eff}}^{-1}\right]_{N\times\eta}\left[\Delta \mathbf{E}_{\text{eff}}\right]_{\eta\times 1} \quad (18)$$

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 II.E, 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 Eq. (18), 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 Eq. (17) according to RoI we have:

$$\begin{bmatrix} G_{\text{eff}}^{(in,in)} & G_{\text{eff}}^{(in,out)}\\ G_{\text{eff}}^{(out,in)} & G_{\text{eff}}^{(out,out)}\end{bmatrix}\begin{bmatrix}\Delta \mathbf{V}^{(in)}\\ \Delta \mathbf{V}^{(out)}\end{bmatrix}=\begin{bmatrix}\Delta \mathbf{E}_{\text{eff}}^{(in)}\\ \Delta \mathbf{E}_{\text{eff}}^{(out)}\end{bmatrix} \quad (19)$$

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:

$$G_{\text{eff}}^{(in,in)}\Delta \mathbf{V}_r^{(in)}=\Delta \mathbf{E}_{\text{eff}}^{(in)}-G_{\text{eff}}^{(in,out)}\Delta \mathbf{V}^{(out)} \quad (20)$$

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 Fig. 1. For this small system, any direct or iterative solver can be used; here, we use LAPACK [17]. 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 $\mathbf{V}$.

IV. 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 II.A with LU decomposition of the LHS matrix of Eq. (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 [16]. The work of [16] 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 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 II.A, 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 of the information he needs. 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 processed motels. Formally:

- q_{ij}: (# of visits to motel i in M walks from j)/M
- h_{ij}: (# of stops at home i in M walks from j)/M

Writing these in matrix format, we have:

$$Q=\left[q_{ij}\right]_{N\times N},\ H=\left[h_{ij}\right]_{N\times N} \quad (21)$$

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 II.E there is a direct relation between the backward random walks and G^{-1}, through Eq. (12). Similarly, there is a direct relation between H, Q and G^{-1}, stated in Theorem 2 below. The idea behind this relation is that the full visit records of Eq. (12) can be reconstructed from columns of H and Q.

Theorem 2 *Given the visit records, $\{H, Q\}$, G^{-1} can be found by:*

$$Z_j = Q_j + \sum_{i=1}^{N-1} h_{ij} Z_i, \; j = 1, \ldots, N \tag{22}$$

$$G^{-1} = D^{-1} Z \tag{23}$$

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: By definition, h_{ij} is the number of times that a walk, started from motel j, reaches home i. For the full visit record, denoted by Z_j, it is enough to add h_{ij} times the average visit number to all of the motels, given the walk was started from node i (i.e., Z_i), to the collected record, Q_j. Finally, G^{-1} is found from Z using Eq. (14).

Eq. (22) can be rewritten in matrix format as:

$$\begin{aligned} G^{-1} &= D^{-1}(I-H)^{-1}Q \\ G &= Q^{-1}(I-H)D = L_G U_G \end{aligned} \tag{24}$$

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 [16].

V. Experimental Results

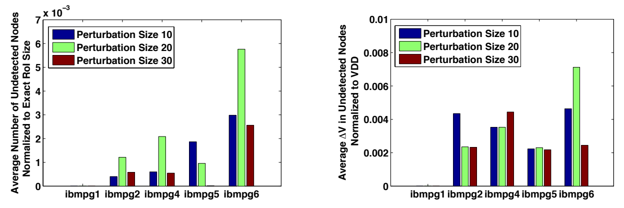
A. Performance of the Backward Random Walk Solver

Our incremental solver is tested and compared on a UNIX machine with 2.5GHz processor and 8GB RAM, and applied to the IBM power grid benchmarks of [15], where $V_{dd} = 1.8$V.

To demonstrate the accuracy of the RoI found by the proposed algorithm, we apply various randomly generated perturbations to the benchmarks and find the corresponding RoI using the proposed algorithm as well as an exact direct solver. The *exact RoI* 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 first measure of the quality of the RoI found by our approach is the number of *undetected nodes*: these are nodes that belong to the exact RoI, but are not listed in the RoI found by our method. The second measure is the amount of error in the solution of the undetected nodes.

The benchmarks are perturbed by randomly choosing a node and modifying the conductances, g_{ij}'s, and the current loads, e_i's, of that node and a number of its neighboring nodes. The amount of change applied is called the *perturbation amount* and the number of nodes being perturbed is called the *perturbation size*.

For the range of perturbations in our experiments, the size of the RoI is no more than about 2% of the total circuit size, and depending on the change, is often less. 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.



(a) Normalized number of undetected nodes
(b) Average change in the voltage of undetected nodes

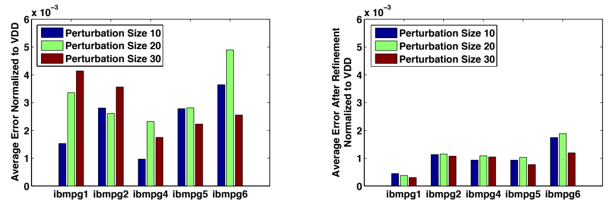
Fig. 5. Number of undetected nodes, normalized to the exact RoI size, and the average change in their voltage for various perturbation sizes (tolerance = 1%V_{dd}, perturbation amount = 20%, averaged over 10 perturbations).

Fig. 5(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%. The numbers shown in all plots are averaged over 10 different sets of perturbations. This figure indicates that the number of undetected nodes is equal to zero for benchmark $ibmpg1$, and for the rest of the benchmarks, this number is less than 0.7% the size of the corresponding exact RoI.

Then the next issue is the significance of these undetected nodes. The error caused by these nodes are the error of the estimated solution from random walks. The average of this error is plotted in Fig. 5(b). 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 are not detected, see an insignificant degradation in accuracy, and this is due to the fact that the estimated solution given by random walks is fairly 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 II.E, the differences remain less than the given tolerance (i.e., 1%V_{dd}), with a confidence of α. Due to this effect, in Fig. 5(b), the error for perturbation size of 20 was more than that for a perturbation size of 30.

In this paper, like other authors, 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 hass very few, resulting in very long walks and hence large runtimes. For such circuits, conventional solvers should be used instead of random walk solvers.



(a) Before refinement
(b) After refinement

Fig. 6. Absolute error of the solution normalized to V_{dd} and averaged over 10 perturbations, for nodes within the RoI *before* and *after* the refinement phase. (tolerance = 1% of V_{dd}, perturbation amount = 20%, averaged over 10 perturbations).

Next, we examine the accuracy of the solution within the RoI. Fig. 6(a) shows the average error of the nodes within the RoI normalized to V_{dd}, for different perturbation sizes. 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 0.5% 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 Figs. 6(a) and 6(b).

TABLE II
Runtime comparisons (tolerance = 1% of V_{dd}, perturbation region size = 30, perturbation amount = 20%)

	Equation Size (N)	Hybrid [2] (sec)	Refinement (sec)	Incremental Total (sec) [Speedup]
ibmpg1	30638	0.170	0.002	0.119[1.4×]
ibmpg2	127238	2.110	0.053	0.725[2.9×]
ibmpg4	953583	24.210	0.202	1.771[13.7×]
ibmpg5	1079310	15.500	0.136	1.975[7.8×]
ibmpg6	1670494	25.470	0.127	2.040[12.5×]
			Average Speedup:	7.7×

Table II compares the runtime of our proposed incremental solver with the Hybrid Solver of [2], 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 the size of the benchmarks N. 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 incremental solver for perturbation size of 30 and perturbation amount of 20% is significant: an average of $7.7\times$ and a maximum of $13.7\times$.

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.

B. Backward Solver on Asymmetrical LHS Matrices

Large power grids are typically abstracted on to grids where the loads are modeled as current sources at nodes on the grid. However, commonly, and even on standard benchmarks such as [15], the current loads are modeled as independent sources. This ignores the effect of the voltage drop at the cells that draw current, introducing inaccuracy in the solution. This modeling error can be corrected by use of voltage controlled current sources (VCCS). It can be shown that mapping current loads at low levels of the power grid to VCCS's connected to the grid results in an *asymmetrical* G matrix.

We have created variations of the **ibmpg** benchmarks [15] that capture this asymmetry effect while modeling the power grid more accurately. In particular, we show results on circuits **ibmpg1'** and **ibmpg2'**, which are represent the same grids as **ibmpg1** and **ibmpg2**, respectively, but with VCCS's instead of curent sources. Our first result in Table III demonstrates that the use of asymmetrical LHS matrices with VCCS's is essential for accuracy. To detect the model error precisely and compare the accuracy of the symmetrical and asymmetrical power grid abstractions, we use exact LU solvers here. For these circuits, ignoring the VCCS behavior can result in absolute errors of up to almost 30% of V_{dd}. On the other hand, using VCCS's makes the power grid model more realistic, resulting in a solution with better average and maximum errors.

TABLE III
ACCURACY: ASYMMETRICAL VS. SYMMETRICAL MODEL

Asymmetrical Model Error			Symmetrical Model Error		
	Normalized to V_{dd}			Normalized to V_{dd}	
	Average	Max		Average	Max
ibmpg1'	2.00%	8.55%	**ibmpg1**	6.16%	28.51%
ibmpg2'	0.05%	1.54%	**ibmpg2**	5.72%	14.47%

TABLE IV
COMPARISON: INCREMENTAL SOLUTION OF SYMMETRICAL AND ASYMMETRICAL POWER GRID LHS'S (TOLERANCE = 1% OF V_{dd}, PERTURBATION REGION SIZE = 30, PERTURBATION AMOUNT = 20%)

Asymmetrical Model			Symmetrical Model	
	Error (Norm. to V_{dd})	Runtime (Sec)		Runtime (Sec)
ibmpg1'	1.3e-4	0.106	**ibmpg1**	0.119
ibmpg2'	8.2e-5	0.774	**ibmpg2**	0.725

Next, we apply the backward random walk solver on these asymmetrical LHS's to solve the incremental analysis problem. Table IV shows together the results for both the asymmetrical and symmetrical cases; the numbers for the latter come from Table II. The total runtime of an incremental solution and the absolute error of the solution, relative to the exact solution of the VCCS based model, are also shown. Evidently, the backward solver can sove both the less accurate symmetrical and more accurate asymmetrical cases fast; it can be shown that the forward solver does not solve the asymmetric incremental problem efficiently.

C. Comparing Forward/Backward Solver Based Incremental Analysis

In order to compare the proposed approach with [11], the incremental solver of this work is applied to the same benchmarks with the same setup of [11]. The amount of perturbation is chosen uniformly in interval (0, 10%), with perturbation sizes of 10, 20, and 30.

The results indicate that the average absolute error of the solution for the nodes within the RoI, normalized to V_{dd}, is reduced $5\times$ to $12\times$ before refinement and up to $3\times$ after refinement. They both show about the same performance in number of undetected nodes. The amount of $\Delta\mathbf{V}$ in the undetected nodes in backward walks approach turns out to be up to $4\times$ more than that of [11] but yet below the threshold of 1% of V_{dd}.

The runtime benefit of the proposed backward walks method becomes more clear as the benchmark size increases. For the smallest benchmark, the backward walks runtime is slightly smaller than HybridSolver [2]. It is $3\times$ less than [11] for a single run. For 10 consecutive runs (i.e., for 10 consecutive perturbations), the runtime of the backward walks method is only 30% more than [11]. For the largest benchmark, the backward solver is $3.6\times$ faster than HybridSolver. In addition, the backward solver is $9.4\times$ faster than [11] for a single run and $1.5\times$ faster for 10 runs.

Note that as the number of runs increase, the runtime benefit of backward solver compared to [11] decreases. However, the solution from [11] solver looses its accuracy as errors from consecutive perturbations accumulate; on the other hand, the backward incremental solver retains its accuracy regardless of the number of perturbations.

REFERENCES

[1] H. Qian, S. R. Nassif, and S. S. Sapatnekar. Random walks in a supply network. *Proc. DAC*, pp. 93–98, 2003.
[2] H. Qian and S. S. Sapatnekar. A hybrid linear equation solver and its application in quadratic placement. *Proc. ICCAD*, pp. 905–909, 2005.
[3] W. Guo, S.X.D. Tan, Z. Luo, and X. Hong. Partial random walk for large linear network analysis. *Proc. ISCAS*, volume 5, 2004.
[4] T. Miyakawa, K. Yamanaga, H. Tsutsui, H. Ochi, and T. Sato. Acceleration of random-walk-based linear circuit analysis using importance sampling. *Proc. GLSVLSI*, page 211–216, 2011.
[5] E. Chiprout. Fast flip-chip power grid analysis via locality and grid shells. *Proc. ICCAD*, pp. 485–488, 2004.
[6] M. Zhao et al. Hierarchical analysis of power distribution networks. *Proc. DAC*, pp. 150–155, 2000.
[7] G. H. Golub and C. F. van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 3rd edition, 1996.
[8] L. T. Pillage, R. A. Rohrer, and C. Visweswariah. *Electronic Circuit and System Simulation Methods*. McGraw-Hill, New York, NY, 1994.
[9] Y. Fu et al. A novel technique for incremental analysis of on-chip power distribution networks. *Proc. ICCAD*, pp. 817–823, 2007.
[10] Y. Ye, Z. Zhu, and J. R. Philips. Generalized Krylov recycling methods for solution of multiple related linear equation systems in electromagnetic analysis. *Proc. DAC*, pp. 682–687, 2008.
[11] B. Boghrati and S. Sapatnekar. Incremental solution of power grids using random walks. *Proc. ASPDAC*, page 757–762, 2010.
[12] J. M Hammersley and D. C Handscomb. *Monte Carlo Methods*. Taylor & Francis, 1964.
[13] F. Castro, M. Sbert, and J. H Halton. Efficient reuse of paths for random walk radiosity. *Computers & Graphics*, 32(1):65–81, 2008.
[14] H. Qian, S. R. Nassif, and S. S. Sapatnekar. Power grid analysis using random walks. *IEEE TCAD*, 24(8):1204–1224, August 2005.
[15] S. R Nassif. Power grid analysis benchmarks. *Proc. ASPDAC*, page 376–381, 2008.
[16] H. Qian and S. S. Sapatnekar. Stochastic preconditioning for diagonally dominant matrices. *SIAM Journal on Scientific Computing*, 30(3):1178–1204, March 2008.
[17] LAPACK. http://www.netlib.org/lapack/.