A Scaled Random Walk Solver for Fast Power Grid Analysis [*†]

Baktash Boghrati, Sachin Sapatnekar
University of Minnesota, Minneapolis, MN, USA.
{baktash, sachin}@umn.edu

Abstract

The analysis of on-chip power grids requires the solution of large systems of linear algebraic equations with specific properties. Lately, a class of random walk based solvers have been developed that are capable of handling these systems: these are especially useful when only a small part of the original system must be solved. These methods build a probabilistic network that corresponds to the power grid. However, this construction does not fully exploit the properties of the problem and can result in large variances for the random walks, and consequently, large run times. This paper presents an efficient methodology, inspired by the idea of importance sampling, to improve the runtime of random walk based solvers. Experimental results show significant speedups, as compared to naive random walks used by the state-of-the-art random walk solvers.

I. Introduction

The supply network is a critical part of any integrated circuit, and is responsible for delivering supply currents and maintaining a stable supply level that is at or near V_{dd} (or ground). Variations in the supply voltage level in nanometer-scale circuits can result in significant performance fluctuations and unpredictability. To ensure correct operation, it is essential to solve the supply network analysis problem efficiently.

We consider the problem of analyzing V_{dd} grids here; the problem associated with ground grids corresponds to a variable translation and is equivalent. The power grid equations are:

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

where $G \in \Re^{n\times n}$ is the left hand side matrix, $\mathbf{V} \in \Re^n$ is the vector of unknowns, and $\mathbf{E} \in \Re^n$ is the right hand side vector. Then G is symmetric, i.e., the $(i,j)^{\text{th}}$ element of G, $g_{ij} = g_{ji}\ \forall\ 1 \le i,j \le n$, and diagonally dominant, i.e., $g_{ii} \ge \sum_{j,j\neq i} \mid g_{ij} \mid$.

The size of the power grid in a modern processor is extremely large. Several approaches have been proposed in the past, including direct methods, those incorporating hierarchy [1] and multigrid schemes [2], iterative methods, and random walk methods [3]. It is often necessary to analyze only a part of this network as changes are made to the system. Of the above methods, random walk methods have the distinction of permitting localized analysis, i.e., solving a small number of nodes in the network without solving the entire network: this is a problem that direct and iterative solvers are incapable of handling as efficiently. This problem rises in different stages of the VLSI chip design, for example, within the design loop a change in the network is applied and the solution of that portion is of interest only. Random walk based solvers, due to their locality property, make it possible to solve for nodes in the network without solving for the entire system.

Early implementations of the random walk scheme for power grids were based on new insights into the Doyle and Snell [4] model with new efficiencies when an entire network is to be solved [3], and then recognizing that this approach was essentially an approximate LU factor computation [5].

The novelty of this paper is in pointing out that even in applying random walks to solve a single node (depending on the scenario, this analysis may be the problem to be solved, or may be a unit problem in the solution of the entire grid), further efficiencies are possible. The solution draws upon some key characteristics of power grids: first, that in any reasonably designed grid that is to be analyzed, the solutions at all nodes are very similar in value, and second, that for the most part (and for DC analysis in particular), the directionality of current loads from the functional blocks is similar, drawing current from the power network (or delivering current into the ground network). In other words, the entries of the right hand side vector, $\mathbf{E}$, are greater than or equal to zero: this is true in practice for DC analysis, and also for AC analysis, even in the presence of package inductances, which are handled as in [6].

In general, the performance of a naïve random walk method may be improved by guiding the randomness in a productive direction [7, 8]. In this work, we develop a heuristic method inspired by Importance Sampling (IS) [9], a variance reduction technique, to reduce the runtime of the random walk solver, while maintaining the level of accuracy. The idea behind the IS method, as discussed in more detail in Section II. C, is to modify the distribution of the naïve random walks so that the metric provided by each walk, referred to as its gain, is approximately the same. The change in the distribution causes a bias in the solution which is compensated for, using a weighted average scheme, and is referred to as *scaling*.

II. Background

A. Random Walk Method

We briefly review the random walk method in the context of power network analysis. Consider a node i in a power network, connected by resistors to each neighboring node $j = 1, \cdots, degree(i)$ by a conductance γ_{ij}, and with a grounded current source at i, I_i (possibly of value 0). Let v_t be the voltage at any node t in the network. The application of Kirchhoff's

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

current law, Kirchhoff's voltage law, and the device equations implies: $\sum_{j=1}^{degree(i)} \gamma_{ij}(v_j - v_i) + I_i = 0$. The terms in this equation can be rearranged as:

$$v_i = \sum_{j=1}^{degree(i)} \frac{\gamma_{ij}}{\Gamma_i} v_j + \frac{I_i}{\Gamma_i} \tag{2}$$

where $\Gamma_i = \sum_{j=1}^{degree(i)} \gamma_{ij}$. Equation (2) implies that the voltage of node i is a weighted sum of the voltages of its neighbors. Due to diagonal dominance, all weights lie between 0 and 1, and the sum of the weights is less than 1 (and equal to 1 if we add in conductances to ground at i). For a power grid that has N non-V_{DD} nodes, we may write N equations of this form, one for each node. The solution of this system of N equations provides the exact solution to Equation (1).

The random walk framework is based on an analogy between this system of equations and a random walk game on a network of roads. The nodes in this network have a one-to-one correspondence with those in the power grid. Nodes that are connected by resistors in the power grid are connected by roads in this network. At a given node, the probability, p_{ij}, of taking a specific road maps on to the corresponding weight, γ_{ij}/Γ_i, for the edge (Equation (2)). Each node has a motel that charges a *motel cost* of $m_i = I_i/\Gamma_i$ when visited. The nodes that correspond to fixed voltages v_i (e.g., V_{DD} or ground nodes) provide a reward of v_i, and are called *home nodes.*

A system of equations defined by Equation (2), for all i, can be modeled as a random walk game on this network of roads, where the walker goes from node i to its adjacent node j with probability p_{ij} and pays a motel cost, m_i, at each visited node. The walk terminates when the walker arrives at a home node.

For a node of interest, i, a random walker starts with zero money and a credit card that allows motel costs to be charged until a reward is obtained for reaching a home node. We define this amount for each random walk as the *gain* of that walk. Then the expected value of the gain, v_i, over all walks that begin at node i is given by

$$v_i = \sum_{j=1}^{degree(i)} p_{ij} v_j + m_i \tag{3}$$

It is easy to see that this equation and Equation (2) correspond exactly. The notation, v_i, is overloaded to denote both the average gain of the random walk and the voltage of the node, precisely because the two are identical, i.e., the expected value of the walk gain, given that the walk started from node i, maps directly to the voltage at node i.

The expected value of v_i from the random walk can be estimated by running a sufficiently large number of random walks from i and calculating the average result to get the estimated voltage of node i without solving for any other node within the network. If the number of walks from i is M_i, then:

$$v_i = \frac{1}{M_i} \sum_{k=1}^{M_i} V_i^k \tag{4}$$

where V_i^k is the walk gain in the k^{th} random walk. We refer to this approach as the *naïve simulation approach*. This approach is always unbiased [7] since $v_i = E_P[V_i]$ where V_i is the random variable with probability mass function (PMF) P_i^k equal to the probability of k^{th} random walk in Equation (4), which is equal to the product of the transition probabilities of the k^{th} random walk starting from node i. Next we will see how this naïve approach can be modified to reduce its variance using importance sampling.

B. The Notion of Importance Sampling

The idea behind the IS method, as discussed in more detail in Section II. C, is to modify the distribution of the *naïve* random walks so that the gain of each walk is approximately the same. The change in the distribution causes a bias in the solution which is compensated for, using a weighted average scheme, and is referred to as *scaling*.

Figure 1 shows a portion of a graph on which random walks are run. For both the naïve and the optimally scaled game, each vertex in the graph represents a node in the power network. The edges are annotated with a set of transition probabilities (only two of which are actually shown in the figure), and the dashed edges represent connections to the rest of the network.

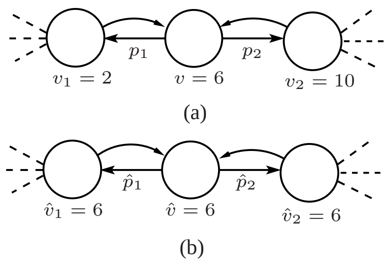


Fig. 1. Example of the (a) naïve and (b) optimally scaled random walk games.

In the random walk game, the walker must to thoroughly inspect vertices on either side of v, i.e., the walks that start with transitions to v_1 and those that first go to v_2. For the naïve game, the walks through v_1 will have an average gain of $v_1 = 2$ (equal to its solution), while those through v_2 will have an average gain of $v_2 = 10$. This indicates that the variance of the data samples that are averaged to compute v may be large.

In the ideal optimally-scaled game shown in Figure 1(b), the left hand side matrix is modified such that the solution of all the nodes are equal to (or practically, close to) 6. Regardless of the direction chosen, the average walk gain will be the same and equal to 6. If this is true at each node, the walk gain will have zero variance. Hence, the stopping criterion will kick in much more quickly since the samples have lower variance. In order to permit this change, as we will soon see, the probabilities of the corresponding edges must be changed from the naïve game and scaling factors must be employed.

C. The Theory of Importance Sampling

Importance sampling is based on the observation that:

$$\begin{aligned} v_i &= E_{Q_i}[V_i] = \sum_k V_i^k Q_i^k \\ &= \sum_k \left(V_i^k \frac{Q_i^k}{\hat{Q}_i^{\,k}} \right) \hat{Q}_i^{\,k} = E_{\hat{Q}_i}[V_i L_i] \end{aligned} \tag{5}$$

where ${\hat{Q}_i}^k Q_i^k > 0, \forall k$. The terms Q_i and $\hat{Q}_i$ are PMFs of the random variable V_i in the naïve and scaled problem, respectively, and V_i^k denotes the k_{th} sample of V_i with probability Q_i^k and $\hat{Q}_i^k$, in the naïve and scaled problem, respectively.

The term $L_i^k = Q_i^k / \hat{Q}_i^k$ is called the likelihood ratio [7]. For the random walk solver for linear equations, the condition ${\hat{Q}_i}^k Q_i^k > 0, \forall k$ ensures that the structure (roads connecting the motels) of the random walk game is preserved.

For a suitable ${\hat{Q}_i}^k$, we can modify Equation (4) using Equation (5) to form a new unbiased estimator for v_i as [7]:

$$\hat{v}_i = \frac{1}{M_i} \sum_{k=1}^{M_i} V_i^k L_i^k \tag{6}$$

where the probability of the random walks is guided by $\hat{Q}_i^k$, and hence the notation $\hat{v}_i$. The variance of this estimator is [7]:

$$\sigma^2_{\hat{v}_i} = \frac{E_{\hat{Q}}[(V_i L_i)^2] - v_i^2}{M_i} = \frac{\mu^2_{\hat{Q}} - v_i^2}{M_i} \tag{7}$$

where $\mu^2_{\hat{Q}}$ is the second moment of the estimator $V_i L_i$ and L_i is a function of k as defined above. If $\hat{Q}_i$ is chosen appropriately, $\hat{v}_i$ will have significantly lower variance than v_i. In the optimal case, IS can achieve exactly zero variance, where the optimal choice of the PMF $\hat{Q}_i$ is given by:

$$\hat{Q}_i^k = V_i^k Q_i^k / v_i \tag{8}$$

If $V_i^k L_i^k = v_i, \forall k$, i.e., all walk gain samples are the same, from Equation (7), we have zero variance, which leads to Equation (8). The name "importance sampling" refers to the notion that $\hat{Q}_i^k$ is proportional to the amount of contribution of the k^{th} walk on the average walk gain, i.e., its importance.

It is clear that since this optimal choice requires some knowledge of v_i, and since v_i is the unknown that we are trying to compute, zero variance is impractical. However, this intuition is the founding idea of our proposed heuristic.

III. Fast Random Walk Solver

A. Scaled Random Walks

Our fast random walk solver is a heuristic technique, inspired by IS, to speedup the random walk solver for power network analysis. The essential idea of the approach is to modify the transition probabilities, p_{ij}, in Equation (3) such that the solution of all the nodes in the network is equal to α and use *scaling factors*, s_{ij}, to find the solution, v_i, to node i. The scaled form of Equation (3) can be written as:

$$v_i = \sum_{j=1}^{degree(i)} \hat{p}_{ij} s_{ij} v_j + m_i \tag{9}$$

where $\hat{p}_{ij}$ (analogous to $\hat{Q}_i$ in Section II. C) denotes the new modified transition probabilities, s_{ij} (analogous to L_i) denotes the scaling factors corresponding to roads from node i to its neighbor j, and m_i is the motel cost at node i as in Equation (3). In the language of IS, the random walks that are constructed using Equation (3) provide an estimate of v_i while those corresponding to Equation (9) (we will shortly define them) are denoted by $\hat{v}_i$; both represent the voltage of node i.

Note that that for Equation (9) to model the same equations as (2) and (3), we must have:

$$s_{ij} = \begin{cases} 1, & i = j \\ p_{ij}/\hat{p}_{ij}, & i \neq j \end{cases}$$

Clearly, $\hat{p}_{ij}$ and p_{ij} both must be valid probabilities that lie between 0 and 1, and therefore $s_{ij} \geq 0$. Moreover, for the structure of the random walks to be intact, required for the IS to be valid as discussed in Section II. C, we must have $s_{ij} > 0$ and $p_{ij} \times \hat{p}_{ij} > 0$.

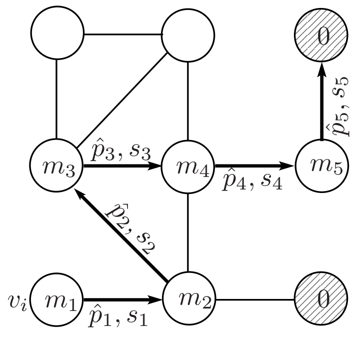


Fig. 2. An example of a scaled random walk game, with a highlighted path showing a walk of length $N = 6$.

Figure 2 shows an example of a graph on which the random walk game may be executed. The home nodes are indicated by the shaded circles and the rest of the vertices are the motel nodes. The highlighted path shows a walk of length $N = 6$ starting from vertex v_i, as the walker takes to reach a home node. The cost of the r^{th} motel node is reflected by m_r and the home rewards are all zero. The modified probabilities and the scales in the graph are shown by $\hat{p}_r$ and s_r, respectively.

The scaled random walk from node v_i, similar to the naïve game, starts with zero money and a credit card on which debt may be racked up. In the scaled walk, the walker accumulates a *multiplier* on the way, which models the scaling scheme. On the first transition the walker pays the price of the motel, m_1, and a road is picked with probability $\hat{p}_1$; the value of the multiplier is set to s_1. Next, the second transition is chosen and the motel costs are paid – but due to the accumulated multiplier, the amount to be paid is not m_2 but $s_1 \times m_2$. The multiplier changes to $s_1 \times s_2$, so that the motel cost at the third transition is $s_1 \times s_2 \times m_3$, the multiplier is $s_1 \times s_2 \times s_3$, and so on. The walker keeps paying and accumulating new terms in the multiplier until a home node is reached, where a reward (also scaled by the multiplier) is received. Therefore, the total gain of a walk of length N can be written as:

$$\hat{V}_i^k = m_1 + s_1 m_2 + s_1 s_2 m_3 + \cdots + \prod_{r=1}^{N-1} s_r m_N \tag{10}$$

where r denotes the motels visited along the way from node i, and k denotes that this is the k^{th} walk among the M_i walks performed to estimate v_i. The result of this walk is included in an average that is used to estimate v_i:

$$\hat{v}_i = \frac{1}{M_i} \sum_{k=1}^{M_i} \hat{V}_i^k \tag{11}$$

As in [3], the number of walks, M_i, required for getting the solution with the desired tolerance is determined dynamically based on the estimated solution and its variance at each point in the random walk game such that the error is smaller than a user defined threshold with a confidence of 99%.

B. Computing the Scaling Factors

The scaling factors in Section III. A give us the freedom to modify the road probabilities, $\hat{p}_{ij}$, and yet find the solution to the original equation using Equation (10). A key unanswered issue is the choice of scaling factors. The IS approach suggests that we choose the $\hat{p}_{ij}$ such that every walk has a gain of α, where α is to be selected to get a feasible random walk and the best possible speedup. Based on this, we will now present a set of results that show how these factors may be computed.

Theorem 1 *For a system of linear equations defined by Equation (9), if $s_{ij}v_j$ is at a constant value of α for all neighbors j of node i, and the value of v_i is also α then:*

$$1 - \sum_{j=1}^{degree(i)} p_{ij}/s_{ij} = m_i/\alpha \tag{12}$$

Proof: Based on the conditions shown above, we have:

$$\alpha = \sum_{j=1}^{degree(i)} \hat{p}_{ij}\alpha + m_i \tag{13}$$

Substituting p_{ij}/s_{ij} for $\hat{p}_{ij}$, we have the result. □

The conditions of Theorem 1 state that:

$$s_{ij} = \frac{\alpha}{v_j} \tag{14}$$

For a V_{dd} grid, the values of all node voltages are roughly similar in magnitude for a reasonable candidate power grid: the variations of a well-designed grid are within 10% of V_{dd}. In a use case where such an analyzer is being applied to the inner loop of an optimizer (e.g., one that sets the optimal wire widths in the grid), it is likely that for any candidate configuration, the voltage values may differ by more than 10%, but not by orders of magnitude. Therefore, since the v_j values are similar in magnitude, it is a reasonable approximation to assume that $s_{ij} = s_i$, $\forall j$. Note that we will use this approximation only to reorient the probabilities in the original random walk game for importance sampling. Therefore, by the definition of importance sampling, *this approximation does not affect the correctness of the random walks*, but only the variance of the samples, and hence the convergence speed.

The quality of this approximation is directly related to the amount by which the variance may be reduced. If a power grid has a catastrophic fault that causes its voltage to be very far from V_{dd}, this may not be a good solution, but it is an excellent choice within an optimizer where realistically, most of the candidate solutions have voltages close to V_{dd}.

Corollary 1 Under the assumptions that $s_{ij} = s_i \; \forall j$, the scaling factor s_i can be computed as:

$$s_i = \frac{1}{1 - m_i/\alpha} \sum_{j=1}^{degree(i)} p_{ij} \tag{15}$$

The proof is trivial and follows directly from Theorem 1.

Corollary 1 provides a recipe for determining the scaling factors in Equation (9). However, the parameter α has not been precisely defined. We will now consider constraints on the feasible values of α that are required to maintain the physical constraints associated with a random walk game.

Theorem 2 *For the case where $m_i \geq 0 \; \forall i$, α satisfies the feasibility and physicality of the random walk game if:*

$$\alpha > \max_i(m_i) \tag{16}$$

Proof: For all i and j, we must have:

$$0 < s_{ij} \tag{17}$$
$$0 < \hat{p}_{ij} \leq 1 \tag{18}$$
$$\sum_{j=1}^{degree(i)} \hat{p}_{ij} \leq 1 \tag{19}$$

The first condition is required to keep the structure of the random walk game intact, as required by the IS method in Section II. C: in other words, the graph for the original game is isomorphic to that of the modified game. In the next two constraints, $\hat{p}_{ij} > 0$ implies that s_{ij} is finite (i.e., it avoids a divide-by-zero operation in Equation (10)), and the remaining constraints are basic requirements on a PMF.

Equations (15) and (17) together imply that $\alpha > m_i \; \forall i$, which immediately leads to Equation (16).

Equations (15) and (19) together imply that $m_i/\alpha > 0$. This is self-consistent with the result of this theorem. □

Note that for the case where $m_i \leq 0 \; \forall i$ (for the ground net), a result similar to Theorem 2 may be derived: that $\alpha < \min_i(m_i)$.

C. Choosing the Value of α

Based on Theorem 2, the value of α may be chosen as

$$\alpha = \max_i(m_i)\beta, \; \beta > 1 \tag{20}$$

where β is chosen empirically for the best speedup.

We now present an intuitive feel for the considerations for choosing β. Qualitatively, β determines the probability of transition to a home node at each node of the network. The choice of α alters probabilities p_{ij} to $\hat{p}_{ij}$, but it can be verified from Equation (12) that in general, at any node i, $\sum_i p_{ij} \neq \sum_i \hat{p}_{ij}$. In the original circuit, typically $\sum_i p_{ij} = 1$ at many nodes, but this is not the case in the modified circuit. This introduces a new home transition probability at node i for the fast solver:

$$\hat{h}_i = 1 - \sum_{j=1}^{degree(i)} \hat{p}_{ij} = \frac{m_i}{\max_i(m_i)\beta} \tag{21}$$

As β increases, the home probabilities decrease and therefore it will be less likely for the random walker to reach a home node. As a result, individual walk lengths increase as β increases, and the walker spends more time exploring “far away" parts of the network which have small contributions to the solution due to the locality property of power grids.

On the other hand as β decreases, the home probabilities become larger, making it more likely for random walks to often terminate at nearby home nodes, even within the radius of locality. To achieve accuracy, this implies the need for a larger number of walks, implying a larger total number of steps in the random walk, and hence, larger runtime for the solver. Another factor is that in order to keep m_i unchanged, required by Equation (9), the edges connected to these new home nodes must have a reward of zero, rather than V_{dd}. Therefore, the conditions that led to $s_{ij} = s_i \forall i$ are somewhat violated since one neighbor contributes a value of zero.

Experimental results empirically indicate that values of $5 \leq \beta \leq 50$ give the best results. This value could change if the topology of the benchmarks (e.g., average degree of each node) changes, but companies tend to use a similar style from design to design, and it is likely that this value will not change remarkably, once calibrated.

D. Fast Random Walks Example

In this section, we present an example to provide some intuition as to how the solver works and why it reduces the variance. Figure 3 shows a simple random walk game, in both the naïve and the scaled form. In this figure, each node is represented by a vertex where the shaded vertices are the home nodes with known values of zero. The numbers within each vertex correspond to the motel cost at the node, and the transition probabilities are shown on the edges. The scaled game has scale labels on the edges as well.

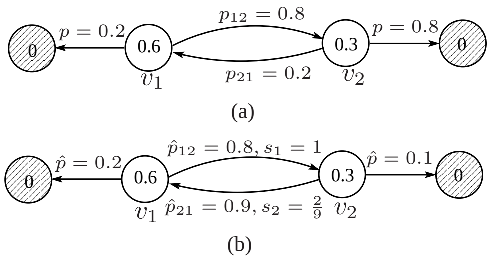


Fig. 3. Simple example of the naïve 3(a) and scaled 3(b) games to demonstrate how the fast random walk solver works.

These games model the following set of equations:

$$\begin{bmatrix} 1 & -0.8 \\ -0.8 & 4 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 0.6 \\ 1.2 \end{bmatrix} \tag{22}$$

where the solution is $v_1 = 1$ and $v_2 = 0.5$. We use $\beta = 5$ and seek the the solution of node v_1.

In the naïve random walk game in Figure 3(a) the gain of the walks started from v_1 takes one of the forms:

$$V_i^k = \begin{cases} 0.6 + 0.9t \\ 0.9(1+t) \end{cases} , t = 0, 1, 2, \ldots \tag{23}$$

The first equation corresponds to a termination at the left home node, and the second to a termination at the right home node. Here, t is the number of times that the walker traverses the cycle v_1–v_2–v_1, and the walk length is $2t+1$ [$2t$] for a termination on the left [right] node. It is easy to see from this equation that the walk gain increases linearly with its length, and depending on the walk length, it falls in the interval $[0.6, \infty)$.

For the scaled random walk in Figure 3(b), some vigorous algebraic manipulations show that the gain of the walks started from v_1 takes one of the forms:

$$V_i^k = \begin{cases} 0.9 \sum_{l=0}^{t} (2/9)^l \\ 0.9 \sum_{l=0}^{t} (2/9)^l - 0.3(2/9)^t \end{cases} , t = 0, 1, 2, \ldots \tag{24}$$

In this game, as t increases (i.e., the walk length increases), the walk gain increases at most to 1.16. Therefore, for this scaled game, the walk gains fall into the interval $[0.6, 1.16)$ which has a much smaller variance compared to the naïve game.

In practice for a properly chosen β, the loop gain in this example and total accumulated walk scale in general case will be less than or equal to one. To see this, consider an ideal power network with the solution of all one and the scaled game for solving it. Comparing Equations (3) (with $v_i = 1$) and Equation (12) we can see that in the scaled game, the RHS $m_i/\alpha = m_i/(\max_i(m_i)\beta) \leq m_i$, the RHS of the naïve game. Therefore, we must have $s_i \leq 1$. For the general case, the solution of the networks of interest for this work deviates no more than, say, 20% and setting β large enough ensures the scaling factors are less than or equal to one.

For efficiency purposes, we stop the walk as soon as the accumulated scale falls below a threshold close to machine precision since the motel costs are bounded and continuing the walk will have no effect on the solution due to round off error.

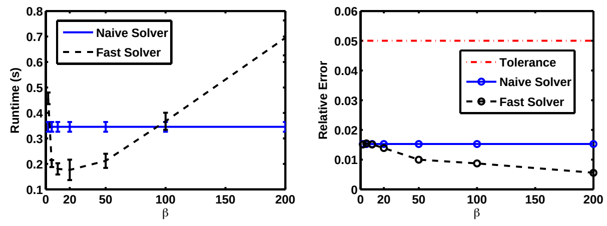
IV. Experimental Results

The proposed fast random walk solver and the naïve random walk solver are implemented in C++ and compared on a UNIX machine with a 2GHz CPU and 8GB of memory. To ensure a fair comparison, the fast random walk solver is implemented by adding the scaling scheme into the naïve solver so that the core random walk engine is the same for both. These solvers are applied to three benchmarks summarized in Table I. This table represents the statistics of the LHS matrix for a DC analysis. For each matrix, we list the size, i.e., the number of unknowns, the number of non-zeros in G, and the average number of non-zeros per row, as a sparsity metric.

TABLE I
Benchmark Details: Statistics of the LHS Matrix

Name	Size	Number of Nonzeros	Average Nonzeros/Row
c1	16194	98030	6.1
c2	26300	165810	6.3
c3	29551	178345	6.0

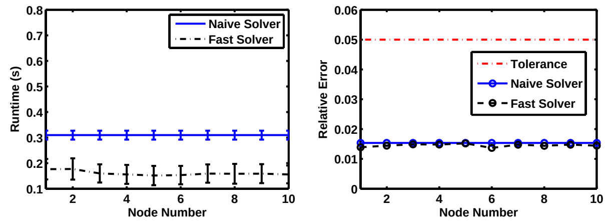
We first examine the efficiency and accuracy of the fast solver by finding the speedup of the fast solver over the naïve solver, for the same relative error. Next we compare the statistics of the walk gains for the naïve and fast solver, as defined by Equations (4) and (10), respectively, to study the variance reduction in the fast solver. In our experiments, we focus on the case that the solution of a single node in the network is of interest. We pick ten nodes randomly and apply the solvers for each of these nodes 1000 times and show that the average results are consistent for all the nodes in all the benchmarks. The nominal voltage of the network, V_{DD}, is $1.2V$.



(a) Runtime (b) Relative Error

Fig. 4. Runtime and relative error vs. β for a random node from circuit c2.

Figure 4 shows the runtime and relative error of the naïve and fast solver versus β, for the given tolerance of 5%, for a single randomly selected node from benchmark $c2$. Figure 4(a) suggests that for $5 \leq \beta \leq 50$, the fast solver achieves a significant speedup while the relative error, shown in Figure 4(b), remains less than or equal to the naïve solver. In Figure 4(a), the bars on the figure denote the standard deviation of the runtime over the 1000 runs.



(a) Runtime (b) Relative Error

Fig. 5. Runtime and relative error for 10 randomly chosen nodes (from c2) with $\beta = 20$.

As discussed in Section III. C, the value of the parameter β is empirically chosen. Here, we set $\beta = 20$ for our experiments. Figure 5 shows the runtime and relative error of the solvers for ten randomly chosen nodes from benchmark $c2$ with tolerance of 5% and $\beta = 20$. In Figure 5(a) the bars on the figure show the standard deviation of the runtime over the 1000 different runs. It can be seen that the fast solver consistently runs faster than the naïve solver for all the nodes while the relative error is the same. Note that it is well-known that the actual error can be significantly below the tolerance, as seen in the figure: this is an artifact of the choice of stopping criterion in [3].

TABLE II

RUNTIME (SEC) AND SPEEDUP OF THE FAST RANDOM WALK SOLVER FOR TEN RANDOMLY CHOSEN NODES FROM EACH BENCHMARK FOR $\beta = 20$

Node	Runtime (s) [Speedup]		
	c1	c2	c3
1	0.067 [1.6×]	0.176 [1.8×]	0.169 [1.3×]
2	0.070 [1.5×]	0.177 [1.7×]	0.170 [1.3×]
3	0.070 [1.5×]	0.160 [1.9×]	0.152 [1.5×]
4	0.068 [1.5×]	0.156 [2.0×]	0.152 [1.5×]
5	0.065 [1.6×]	0.152 [2.0×]	0.153 [1.5×]
6	0.064 [1.6×]	0.153 [2.0×]	0.137 [1.6×]
7	0.054 [1.9×]	0.159 [1.9×]	0.154 [1.5×]
8	0.068 [1.5×]	0.159 [2.0×]	0.153 [1.5×]
9	0.064 [1.6×]	0.159 [1.9×]	0.155 [1.4×]
10	0.055 [1.9×]	0.156 [2.0×]	0.151 [1.5×]
Average	**0.065 [1.6×]**	**0.161 [1.9×]**	**0.155 [1.5×]**

Details of the runtime and speedup of the fast random walk solver are listed in Table II for all the benchmarks where ten nodes are selected randomly from each benchmark and $\beta = 20$. Each row represents a node number and each column represents the circuit that it comes from. This table indicates consistent runtime reduction for all the benchmarks and for each of the nodes of each benchmark.

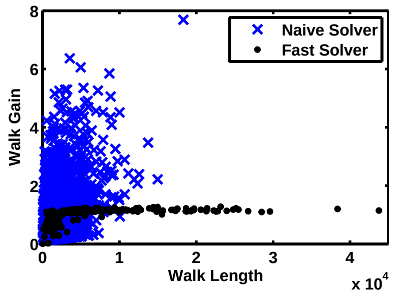


Fig. 6. Distribution of the walk gain vs. walk length ($\beta = 20$, $v = 1.06$).

Finally the scatter plot of Figure 6 shows the distribution of walk gain versus walk length of the naïve solver and the fast solver, as defined by Equations (4) and (10), for a randomly chosen node from benchmark $c2$ with solution of $1.06V$. As this figure indicates, the walk gain of the fast solver is much denser around the solution than that of the naïve solver. This indicates the effectiveness of the variance reduction technique described in Section III. D.

REFERENCES

[1] M. Zhao et al. Hierarchical analysis of power distribution networks. In *Proc. DAC*, pages 150–155, 2000.

[2] J. N. Kozhaya, S. R. Nassif, and F. N. Najm. Multigrid-like technique for power grid analysis. In *Proc. ICCAD*, pages 480–487, San Jose, California, 2001. IEEE Press.

[3] H. Qian, S. R. Nassif, and S. S. Sapatnekar. Random walks in a supply network. In *Proc. DAC*, pages 93–98, 2003.

[4] P. G. Doyle and J. L. Snell. *Random Walks and Electric Networks*. Mathematical Association of America, Washington, DC, 1984.

[5] H. Qian and S. S. Sapatnekar. A hybrid linear equation solver and its application in quadratic placement. In *Proc. ICCAD*, pages 905–909, 2005.

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

[7] I. Kuruganti and S. G. Strickland. Importance sampling for markov chains: computing variance and determining optimal measures. In *Proc. Winter Simulation Conference*, pages 273–280, 1996.

[8] S. Andradottir, D. P. Heyman, and T. J. Ott. Potentially unlimited variance reduction in importance sampling of Markov chains. *Advances in applied probability*, 28(1):166–188, 1996.

[9] P. W. Glynn and D. L. Iglehart. Importance sampling for stochastic simulations. *Management Science*, 35(11):1367–1392, November 1989.