Capacitance Extraction and Power Grid Analysis Using Statistical and AI Methods

(Invited Paper)

Wenjian Yu[1], Ming Yang[1], Yao Feng[1], Ganqu Cui[1], Ben Gu[2]
[1]BNRist, Dept. Computer Science & Tech., Tsinghua Univ., Beijing, China.
[2]Cadence Design Systems, Inc., Austin, USA.
Email: yu-wj@tsinghua.edu.cn

Abstract—Capacitance extraction and power grid (PG) analysis for IC design involve large-scale numerical simulation problems. As the process technology becomes more complicated and design margin is shrinking, the capacitance field solver and power-grid matrix solver with high accuracy and capability for handing large and complex structure are highly demanded. In this invited paper, we present recent application of statistical and AI methods in these two fields. The Markov-chain model and relevant analysis are presented for developing an efficient technique for handling conformal dielectrics in the floating random walk based capacitance extraction. Then, two approaches reducing the computational cost of a domain decomposition based power-grid solver are presented. One employs supervised machine learning while the other is inspired by the A*-search algorithm.

I. Introduction

Accurate parasitic extraction is crucial to the success of today's high-performance IC design. 3-D capacitance field solvers are greatly demanded to ensure the silicon success of ICs, because the process technology becomes more complicated and design margin is shrinking [1], [2], [3]. As a product-proven technique for accurate capacitance solver, the floating random walk (FRW) method is advantageous in scalability, reliability and parallelism over the traditional methods [4], [5], [6], [7], [8], [9]. In modern process technologies, conformal dielectrics are common although most dielectrics are planar. The complex conformal dielectrics bring a big challenge for efficient capacitance extraction [3]. The current method adopted by FRW based solvers relies on pre-characterizing transition cubes with special dielectric configurations and pre-calculating a large amount of surface Green's function tables (GFTs) prior to the online capacitance extraction [4]. During the FRW procedure, the occurring transition cube with complex dielectric configuration is converted to a pre-characterized cube with some dielectric approximation. This approach causes excessive offline computation of pre-characterization and runtime memory cost (at least 2 GB), and induces error at the same time.

On the other hand, modern VLSI design relies heavily on efficient power grid analysis [10], [11], [12]. Because the effect of voltage fluctuations becomes more significant, how to retain the switching speeds and satisfy the noise margin is challenging. This requests accurate and efficient simulation of large-scale power grids. The direct methods for solving sparse linear systems [13] cannot scale well with the problem size for large-scale power grids, while iterative solvers [14] often suffer from the instability of preconditioner. Among a number of specialized methods, the domain decomposition method (DDM) [11] is more suitable for practical usage, but its efficiency should be further improved .

In this paper, we present recent progress in developing accurate and efficient capacitance solver and PG matrix solver. Statistical and artificial intelligence (AI) methods are borrowed to resolve the challenges brought by the conformal dielectrics and huge size of PG. Experimental results show that the presented techniques improve the solvers' accuracy and/or reducing the computational cost. We hope these results are inspiring to future research on large-scale simulation in EDA.

II. Background

A. Floating Random Walk Method for Capacitance Extraction

The FRW method for capacitance extraction is derived from the integral formula for electric potential [5], [15]:

$$u(\boldsymbol{r}) = \oint_S P(\boldsymbol{r}, \boldsymbol{r}^{(1)}) u(\boldsymbol{r}^{(1)}) d\boldsymbol{r}^{(1)}, \quad (1)$$

where $u(\boldsymbol{r})$ is the potential of point $\boldsymbol{r}$ and $P(\boldsymbol{r}, \boldsymbol{r}^{(1)})$ is called surface Green's function. The domain enclosing $\boldsymbol{r}$ is called transition domain, whose surface is S. $P(\boldsymbol{r}, \boldsymbol{r}^{(1)})$ can be regarded as a probability density function. With Monte Carlo method, $u(\boldsymbol{r})$ can be estimated by the mean of $u(\boldsymbol{r}^{(1)})$ values.

For computing the capacitances related to a master conductor i, a Gaussian surface G_i is constructed to enclose it (see Fig. 1). According to the Gauss theorem and (1) for electric potential one can derive the electric charge of conductor i [5]:

$$Q_i = \oint_{G_i} F(\boldsymbol{r}) g \oint_{S^{(1)}} \omega(\boldsymbol{r}, \boldsymbol{r}^{(1)}) \tilde{P}(\boldsymbol{r}, \boldsymbol{r}^{(1)}) u(\boldsymbol{r}^{(1)}) d\boldsymbol{r}^{(1)} d\boldsymbol{r}, \quad (2)$$

where $F(\boldsymbol{r})$ is the dielectric permittivity at point $\boldsymbol{r}$, $\tilde{P}(\boldsymbol{r}, \boldsymbol{r}^{(1)})$ is, probably different from $P(\boldsymbol{r}, \boldsymbol{r}^{(1)})$, for sampling on $S^{(1)}$, and $\omega(\boldsymbol{r}, \boldsymbol{r}^{(1)})$ is called weight value [5]. Thus, Q_i can be estimated as the stochastic mean of sampled values on G_i, which is further the mean of sampled potentials on $S^{(1)}$ multiplying the weight value. If the potential of a sample point $\boldsymbol{r}^{(1)}$ is unknown, (1) is substituted into (2) recursively. The computation can be imaged as a floating random walk (FRW) procedure. The walk starts from the Gaussian surface, and repeatedly jumps until reaching conductor surface. After performing a number of walks, the stochastic mean of the

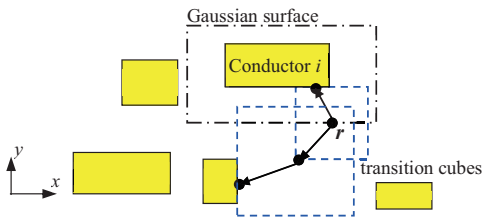


Fig. 1. Two random walks starting from $\boldsymbol{r}$ (denoted by consecutive segments with arrows) in the FRW method for capacitance simulation (2-D top view).

978-1-7281-4123-7/20/$31.00 © 2020 IEEE

weight values for the walks terminating at conductor j is capacitance C_{ij} between conductors i and j.

Each FRW includes a sequence of hops, each of which is from the center of a transition cube T to a random point selected following a certain surface Green's function. In practice, this random sampling is realized by first discretizing T's surface into $6n^2$ panels and obtaining the transition probabilities to the panels, and then randomly choosing a panel [4], [5]. Suppose $u(\boldsymbol{S}_i)$ denotes the potential of the i-th panel's center. The potential of T's center $\boldsymbol{C}$ is:

$$u(\boldsymbol{C}) = \sum_{i=1}^{6n^2} p_i u(\boldsymbol{S}_i), \tag{3}$$

where p_i is the transition probability to the i-th panel. Eq. (3) is the discretized form of (1) and explains the random hops in FRW method. Usually, the transition probabilities are pre-calculated and stored as the tables (called GFTs) for transition cubes with certain dielectric configurations [5]. By loading GFTs before performing FRWs, one can execute hops quickly.

Several approaches for pre-characterizing the transition cubes with multilayer dielectrics have been proposed [4], [5], to deal with the stratified dielectrics in practical capacitance extraction. However, they are not applicable to structures with conformal dielectrics, for which the transition cubes containing both horizontal and vertical variation of dielectrics are involved. An approximate approach was proposed in [4] for handling conformal dielectric. It pre-characterizes a so-called *eight-octant transition cube*, of which each octant contains a homogeneous dielectric (as shown in Fig. 2(a)), with various configurations of dielectric permittivities. During the FRW procedure, the transition cube with both horizontal and vertical dielectric variation (occurring around conformal dielectric, as shown in Fig. 2(b)) is approximated by the eight-octant transition cube with the permittivity of each octant obtained as the volume weighted average permittivity. Then, the transition probabilities of this transition cube are obtained from some pre-calculated GFTs for the eight-octant transition cube and the linear interpolation calculation. This approach exhibits satisfied accuracy for structures with conformal dielectrics, and is better than using a kind of "intrusive-type" transition cubes to approximate the transition cubes near conformal dielectrics [4]. However, pre-characterizing these eight-octant transition cubes still consumes a lot of computation and runtime memory. Notice that the GFT for an eight-octant cube with permittivities $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_8$ is the same as that for an eight-octant cube with permittivities $\varepsilon_1/\varepsilon_{max}, \varepsilon_2/\varepsilon_{max}, \ldots, \varepsilon_8/\varepsilon_{max}$, where $\varepsilon_{max} = \max\{\varepsilon_i\}$. So, the cube can be characterized by the normalized permittivities $\{\varepsilon_i/\varepsilon_{max}\}$. One can only consider some discrete normalized permittivities. If the normalized permittivity ranges from 0.5 to 1 with discretized step s, the number of possible eight-octant configurations will be $(0.5/s+1)^8 - (0.5/s)^8$. Suppose $s = 0.1$ and each GFT needs about 30 KB storage [4]. The memory cost for these eight-octant cubes' GFTs will be 38 GB. Considering the symmetry will reduce the number of configurations, but the cost of GFTs is still about 2 GB. If a smaller value of s is used to pursue higher accuracy, this memory cost will become even larger.

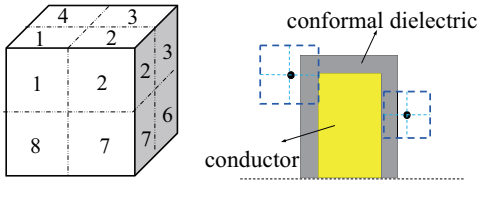


Fig. 2. (a) An eight-octant transition cube where each octant contains a homogenous dielectric (the octants are numbered). (b) The cross-section view of a conformal dielectric (in gray) and transition cubes (in blue) near it.

B. Domain Decomposition Based Power Grid Analysis

We consider the DC analysis of power grid though the DDM is also capable of simulating the transient analysis [11]. The problem can be described using the nodal analysis as $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{f}$, where $\boldsymbol{A}$ is the conductance matrix, $\boldsymbol{x}$ is the vector of node voltages and $\boldsymbol{f}$ denotes the loads of current source. Suppose that the power grid is partitioned into m subdomains by reordering the variables in $\boldsymbol{x}$. The nodes in the original system are classified into interior nodes of subdomains and interface nodes. So, the equation has the blocked sparse structure:

$$\begin{pmatrix} \boldsymbol{A}_1 & & & \boldsymbol{E}_1 \\ & \ddots & & \vdots \\ & & \boldsymbol{A}_m & \boldsymbol{E}_m \\ \boldsymbol{F}_1 & \cdots & \boldsymbol{F}_m & \boldsymbol{A}_\Gamma \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_m \\ \boldsymbol{x}_\Gamma \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_1 \\ \vdots \\ \boldsymbol{f}_m \\ \boldsymbol{f}_\Gamma \end{pmatrix} . \tag{4}$$

Here, the matrices $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_m$ correspond to the m subdomains, and $\boldsymbol{A}_\Gamma$ corresponds to the interface nodes. Matrices $\boldsymbol{E}_i$ and $\boldsymbol{F}_i$ $(i = 1, \ldots, m)$ reflect the connections between the interface and the i-th subdomain.

Eq. (4) can be rewritten as

$$\begin{pmatrix} \boldsymbol{A}_D & \boldsymbol{E} \\ \boldsymbol{F} & \boldsymbol{A}_\Gamma \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_D \\ \boldsymbol{x}_\Gamma \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_D \\ \boldsymbol{f}_\Gamma \end{pmatrix}, \tag{5}$$

where $\boldsymbol{A}_D$ is the blocked diagonal matrix including $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_m$. From the first equation, $\boldsymbol{x}_D$ can be expressed as

$$\boldsymbol{x}_D = \boldsymbol{A}_D^{-1}(\boldsymbol{f}_D - \boldsymbol{E}\boldsymbol{x}_\Gamma) . \tag{6}$$

Substituting (6) into the second equation of (5), one obtains

$$(\boldsymbol{A}_\Gamma - \boldsymbol{F}\boldsymbol{A}_D^{-1}\boldsymbol{E})\boldsymbol{x}_\Gamma = \boldsymbol{f}_\Gamma - \boldsymbol{F}\boldsymbol{A}_D^{-1}\boldsymbol{f}_D , \tag{7}$$

from which $\boldsymbol{x}_\Gamma$ can be solved. Notice that $\boldsymbol{A}_D^{-1}$ will not be explicitly computed. Due to the block diagonal property the computation is decomposed into solving the smaller equations with coefficient matrices $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_m$. This can be easily parallelized, which is the reason why the DDM based approach is practical for large-scale power grid analysis.

The matrix $\boldsymbol{A}$ is symmetric positive definite (SPD) and matrix $\boldsymbol{A}_i$ $(i = 1, \ldots, m)$ inherits this property. Therefore, the solution of each subdomain equation is realized by Cholesky factorization of $\boldsymbol{A}_i$ followed by forward/backward substitutions. This direct method is robust, but it often suffers from large memory consumption caused by fill-ins during factorization. The memory issue may not be pronounced, if the subdomain is small enough. However, to restrict the size of interface which affects the non-parallel part in the solution of (7), the size of $\boldsymbol{A}_i$ cannot be very small.

III. Markov-Chain Analysis for Handling Conformal Dielectrics with FRW Based Solver

We use Markov-chain analysis, in form of discrete random walk, to derive a special property of the transition probabilities

for the eight-octant transition cube. Then, we develop a technique generating the eight-octant cube's transition probabilities on the fly. It largely saves memory cost and improves accuracy for handling conformal dielectrics without, sacrifice.

A. Markov-Chain Model Pre-characterizing Transition Cube

Suppose the dielectric configuration of an eight-octant transition cube T is given. With the finite difference method (FDM), one can numerically compute the transition probabilities p_i in (3) for T [5]. The key point is to obtain the relationship between the potentials of T's center $\boldsymbol{C}$ and the potential of T's surface points.

Lemma 1. *With a suitable FDM grid on the eight-octant cube, a discrete random walk (DRW) method can be applied to express the electric potential of the cube's center $\boldsymbol{C}$ in terms of potentials on cube's surface, i.e. (3). Specifically, the probability that the DRW walker starting at $\boldsymbol{C}$ finally reaches panel i is the transition probability p_i in (3).*

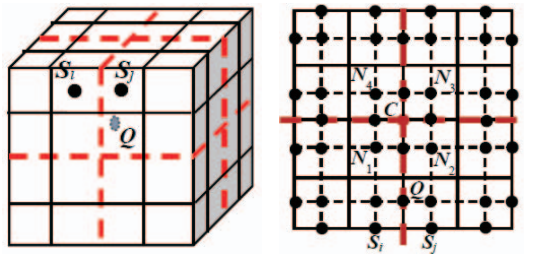
Proof. For the eight-octant transition cube, the FDM grid is formed by first dividing the cube into equal-sized cells [see Fig. 3(a)], where n is an even number, and then attach one unknown of potential to each cell's center. For each surface panel, an unknown of potential is also set at panel's center. The grid lines along the x, y, and z axes connect these unknowns as shown in Fig. 3(b), and intersect the interfaces of octants at interface grid points. For each inner or interface grid point $\boldsymbol{G}_j$, the finite difference equation [5] can be converted to:

$$u(\boldsymbol{G}_j) = \sum_{k\in \mathrm{Ngb}(j)} p_{j,k} u(\boldsymbol{G}_k), \tag{8}$$

where Ngb(j) denotes the set of indices of $\boldsymbol{G}_j$'s neighbor grid points. The coefficients $p_{j,k}$ in (8) are all positive and their sum equals 1 [5]. Similar to the DRW methods for power grid analysis [10] or thermal analysis [16], we can interpret (8) as a random walk step from $\boldsymbol{G}_j$ to its neighbor grid point. Notice that the center point $\boldsymbol{C}$ is not a grid point. So, an additional equation is introduced according to [17]:

$$u(\boldsymbol{C}) = \sum_{k=1}^{8} \frac{\varepsilon_k}{\varepsilon_1+\varepsilon_2+\varepsilon_3+\varepsilon_4+\varepsilon_5+\varepsilon_6+\varepsilon_7+\varepsilon_8} u(\boldsymbol{N}_k), \tag{9}$$

where $\boldsymbol{N}_1 \sim \boldsymbol{N}_8$ are $\boldsymbol{C}$'s adjacent inner grid points from the eight octants respectively, and $\varepsilon_1 \sim \varepsilon_8$ are the corresponding permittivities of the eight octants. The coefficients in (9) also form a set of probabilities, so that a random walk step from $\boldsymbol{C}$ to $\boldsymbol{N}_k$ can be defined. With (8) and (9), the DRW procedure starting from $\boldsymbol{C}$ and terminating at surface points $\boldsymbol{S}_i$ is well defined. According to the stochastic meaning of (3), the probability that the walker finally reaches $\boldsymbol{S}_i$ (or panel i) is the transition probability p_i in (3). □



(a) (b)

Fig. 3. The FDM grids for an eight-octant transition cube, where red dashed lines outline the octants. (a) The cube is evenly divided into $n \times n \times n$ cells. $\boldsymbol{S}_i$ and $\boldsymbol{S}_j$ are two examples of surface panel's center. (b) The top view of the eight-octant cube with the FDM grids imposed. Black points are grid points. $\boldsymbol{C}$ is the cube's center, and $\boldsymbol{N}_1 \sim \boldsymbol{N}_4$ are its four neighbor grid points (the other four neighbor grid points coincide with them in this view).

Lemma 1 connects the transition probability to a Markov-chain model, i.e. DRW procedure. So, Markov-chain analysis can be utilized to reveal a special property associated with the eight-octant cube's transition probabilities.

B. Efficient Treatment of Conformal Dielectrics

The special property of eight-octant cube's transition probabilities is as follows.

Theorem 1. *For two panel center points $\boldsymbol{S}_i$ and $\boldsymbol{S}_j$ in different octants of an eight-octant transition cube T, if they have symmetric positions with respect to T's center $\boldsymbol{C}$, the transition probabilities to them, namely p_i and p_j, satisfy:*

$$\frac{p_i}{\varepsilon_i} = \frac{p_j}{\varepsilon_j} = \frac{8\bar{p}_i}{\sum_{i=1}^{8} \varepsilon_i}, \tag{10}$$

where ε_i and ε_j are the permittivities of the octants containing $\boldsymbol{S}_i$ and $\boldsymbol{S}_j$, respectively. $\varepsilon_1 \sim \varepsilon_8$ are the permittivities of T, and $\bar{p}_i$ denotes the transition probability from $\boldsymbol{C}$ to $\boldsymbol{S}_i$ in a single-dielectric cube with same FDM discretization as T.

Its proof can be accomplished based on analyzing the Markov chain model in DRW, i.e. the the probability of a walker reaching two symmetric points on different octant surfaces. Due to page limit, we omit it here. Theorem 1 implies the following statements for an eight-octant transition cube T,

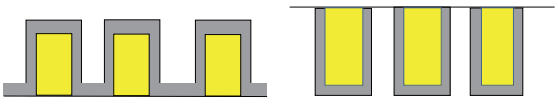
(1) The sum of transition probabilities to the surface of an octant is proportional to the octant's dielectric permittivity;

(2) For a given octant of T, the transition probabilities to its surface are constant multiplies of those for a single-dielectric transition cube.

So, we can obtain the GFT for any eight-octant transition cube from the GFT for a single-dielectric transition cube. The latter can be pre-calculated analytically [15] and is always needed. This enables the on-the-fly generation of the accurate GFT for an eight-octant transition cube. In practice, we first randomly choose an octant following the probabilities proportional to the eight octants' permittivities. Then, for selecting a point on the chosen octant's surface we just follow the single-dielectric GFT. This approach induces no precharacterization computation or memory cost.

We have implemented this technique based on RWCap3 [18]. The improved algorithm is referred to as RWCap4, and used to simulate several cases of VLSI structures with conformal dielectrics. All experiments are carried out on a Linux server with Intel Xeon E5-2650 2.0 GHz CPU.

Case 1∼2 are based on Case 1 (180-nm technology) and Case 4 (45-nm technology) in [5] by adding a conformal dielectric around the three wires in M2 layer (see Fig. 4).



(a) (b)

Fig. 4. A portion of test cases' cross-section view. The coating conformal dielectric has the same permittivity as its adjacent dielectric. (a) Case 1, based on Case 1 in [5]. The thickness of conformal dielectric is 40nm. (b) Case 2, based on Case 4 in [5]. The thickness of conformal dielectric is 5nm.

Case 3∼6 include 3 parallel wires in M2 layer and 10 crossing parallel wires in M3 layer. The M2 wires are coated by different conformal dielectrics and embedded in stratified dielectrics. The size of each wire is 45nm×400nm×90nm. The spacing between wires is 45nm for Case 3∼5 and 28nm for Case 6. For Case 5, there are multiple conformal dielectrics.

For Case 1∼6, one of the M2 wires is set as the master conductor and its capacitances are extracted. The termination criterion for the FRW algorithm is 0.5% 1-σ error on the total capacitance. The cases are simulated with the proposed algorithm and the golden-standard tool Raphael [19] based on FDM. The results are listed in Table I. From it we see that the on-the-fly sampling scheme on eight-octant transition cubes ensures good accuracy. Besides, RWCap4 consumes about 50 MB memory, same as RWCap3. This means the proposed approach does not cost extra memory. More experiments are carried out to compare the runtime of RWCap3 and RWCap4. The results reveal that the latter is only a little bit slower.

TABLE I
THE COMPUTATIONAL RESULTS FOR THE SIX CASES WITH CONFORMAL DIELECTRICS

Case	RWCap4					Raphael
	#walk	#hop	Time(s)	Cap.(fF)	Error	Cap.(fF)
Case 1	103K	52.5	3.74	2.060	0.5%	2.050
Case 2	48K	52.9	1.24	0.401	0.5%	0.399
Case 3	67K	68.9	2.09	0.104	-1.0%	0.105
Case 4	72K	82.4	2.73	0.106	0.9%	0.105
Case 5	65K	67.4	2.14	0.104	-1.0%	0.105
Case 6	34K	38.6	0.56	0.161	1.2%	0.159

The experimental results validate that the proposed technique avoids the pre-calculation and runtime memory cost for characterizing the eight-octant transition cubes. Moreover, the improved FRW algorithm is more accurate than the approach in [4] as no interpolation computation is needed.

IV. AI Methods for DDM Power-Grid Solver

To reduce the memory cost and runtime of the DDM based power-grid solver, two AI inspired techniques are presented.

A. The Matrix Reordering Problem

In the parallel DDM for power grid analysis, factorizing the subdomain matrices are the major work. To reduce its cost, we consider the matrix reordering problem: *Given a sparse SPD matrix $\boldsymbol{A}_i$, find a row permutation $\boldsymbol{P}$ such that the number of nonzeros (or the amount of work required to compute) in the factorization of $\boldsymbol{PA}_i\boldsymbol{P}^T$ is minimized.* It is equivalent to finding an order of the nodes in the undirected graph corresponding to the sparse matrix. As this is an NP-hard problem, heuristics are used to reduce fill-ins. Three basic strategies exist [20]: (1) ordering the nodes with minimum degree first and its variants, (2) nested dissection which recursively partitions the graph with node separators, (3) band reduction which constrains nonzeros to a small band region around the diagonal. The representative of the first strategy is the AMD algorithm [21], while that of the second is *metis* [22]. The reverse Cuthill-McKee algorithm belongs to the last one, used for matrices generated from structural mechanism problem. Although AMD is the most widely-used, the optimal ordering algorithm varies case by case. Consequently, there is a strong need to develop an efficient approach to automatically choose the optimal ordering algorithm in practice.

B. A Supervised Learning Based Approach

We propose a supervised learning based approach to improve the solution of subdomain matrices. It is composed of several steps. Firstly, a set of subdomain matrices collected from large-scale power grid analysis are tested with different matrix orderings. According to the fill-in reduction or peak memory usage the optimal ordering method is labeled for each matrix. Then, the set is spitted into two subsets, one as the training set and the other as the testing set. With the training set, two supervised learning methods based on support vector machine (SVM) and multilayer perception neural network (MLP-NN) respectively, are developed to train two possible classifiers for choosing the optimal ordering method. The simplicity of the classifier ensures less computation in the inference. Finally, the classifiers are used to infer the better ordering for a specified test matrix.

Support vector machine (SVM) is a widely used machine-learning technique for classification. It computes an optimal hyperplane to separate two groups of labeled data. If the sets of data to discriminate are not linearly separable, the kernel trick can be used. It maps the original data to a higher dimensional space presumably making the separation easier. After this mapping the inner product in the original space needs to be changed to a new one, which is called kernel function and used for training the SVM classifier. Alternatively, MLP-NN can also be used for classification. With linear transfomation and nonlinear activation function, MLP-NN maps inputs into outputs. It uses the error backpropagation to train parameters, and the softmax function at the output layer for classification.

Our idea is to automatically select the best matrix ordering algorithm from the three discussed in last subsection, via a well-trained machine learning classifier. The key points include the training data with labels, the representation of a data (subdomain matrix here), and the technique for classification. For the training data, we can obtain them from industrial design of power grid, with the help of our industrial partner. Then, we shall test these matrices with different ordering approaches to get the label. For our problem, the number of fill-ins and memory usage during the Cholesky factorization are the two criteria for judgment.

To make the proposed approach useful for the power grid analysis, its runtime efficiency at the inference stage is of major concern. The representation of a test data affects the efficiency. In the problem, data is the subdomain matrix, whose dimension is larger than several thousands. Instead of directly depicting the matrix we consider its graph counterpart. Each matrix corresponds to an undirected graph, and the property of graph affects the choice of fill-reducing ordering. So, we use a couple of features from the view point of graph to represent the matrix data.

The following list summarizes a number of important features which can capture the major characteristics of a undirected graph (subdomain matrix).

- N_n: the number of nodes (matrix dimension);
- N_e: the number of edges (number of nonzeros in matrix);
- avg_d: the average degree of a node;
- $density$: the density of sparse matrix, i.e. $\frac{2N_e}{N_n(N_n-1)}$;
- max_d: the maximum degree of a node;

- min_d: the minimum degree of a node;
- $diameter$: the maximum distance between two nodes.

We also use the SVM-recursive feature elimination (SVM-RFE) technique to further reduce the features [23]. It first trains a SVM classifier with all features. Then, it estimates the relatively contribution of each feature and the least important features are removed. This procedure is repeated until the loss on the classification accuracy is above a threshold. Finally, five features are selected: N_n, N_e, avg_d, max_d and $diameter$.

Calculating $diameter$ is of $O(N_n^2)$ complexity.This could be expensive for practical use. So, we devise an approximate approach to calculating the graph diameter. The idea is that we find the farthest node from an initial node and record their distance. Then, we repeatedly change the initial node and calculate its farthest node until the corresponding distance does not increases. The procedure usually converges in a couple of iterations, so that its complexity is about $O(N_n)$. Empirical results have shown that the approximate diameter well captures the feature of graph.

For SVM classifier, the kernel trick with the Gaussian radial basis function is applied. For MLP-NN, we build a three-layer network which includes only one hidden layer. The size of hidden layer is 16. The activation function is set to ReLU. The gradient decent algorithm is used to training the weights and bias in the network. The learning rate is fixed at 1e-3. The maximum iteration number to set to 100. All tolerances for convergence are 1e-3.

Two large-scale power grids from real design (with 40 million and 22 million nodes), provided by our industrial partner, are tested. They are partitioned separately, resulting in 2660 subdomain matrices in total. The dimension of these matrices ranges from 11810 to 36473. A program is written in Matlab2016b to test them with three ordering approaches: AMD, *metis* and RCM. For AMD and RCM the build-in commands "`amd`" and "`symrcm`" are used respectively. For *metis*, the Matlab interface of metis-5.1.0 [22] provided in SuiteSparse [24] is employed. Applying the generated orders followed by Cholesky factorization ("`chol`" in Matlab) we measure the number of fill-ins or peak memory usage to determine the optimal ordering algorithm for each matrix. For the tested matrices, we find out that it is always AMD or *metis*. So, this leads to a binary classification problem.

We first construct two datasets: Dataset1 with 2128 randomly selected matrices (80% of the whole set) as the training subset and the remaining matrices as the testing subset, Dataset2 with 1862 matrices (70% of the whole set) as the training subset and the remainder as the testing subset. With the number of fill-ins and memory cost as the criterion separately, the numbers of matrices whose best orderings are AMD or *metis* are listed in Table III. We can observe that although AMD wins for most cases there are about 29% of the matrices whose best ordering algorithm is *metis*. And, the criterion of memory is similar to fill-in, as the number of fill-ins largely determines the peak memory cost.

The classifiers are implemented in Python 2.7.6, with Scikit-Learn package. The SVM based classifiers for optimal fill-ins and peak memory are denoted as Classifier1 and Classifier2 respectively. To evaluate their performance, we measure four metrics: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). The inference results on the two testing subsets are listed in Table IV. Here, "positive" and "negative" refer to AMD and $metis$, respectively. From the results we observe that the classifiers perform very well.

TABLE II
THE DISTRIBUTIONS OF THE OPTIMAL ORDERING ALGORITHMS

	Criterion	Training Subset		Testing Subset	
		AMD	*metis*	AMD	*metis*
Dataset1	Fill-in	1508	620	384	148
	Memory	1494	634	377	155
Dataset2	Fill-in	1329	533	563	235
	Memory	1319	543	552	246

TABLE III
THE PERFORMANCE OF THE BINARY CLASSIFIERS BASED ON SVM

Classifier1	TP	TN	FP	FN	Classifier2	TP	TN	FP	FN
Dataset1	381	128	3	20	Dataset1	374	128	3	27
Dateset2	562	208	2	27	Dataset2	551	208	2	38

More computational results on the two datasets are given in Table V. "Time" means the inference time for the whole testing subset. "Accuracy" means the fraction correct, i.e. (TP+TN)/(TP+TN+FP+FN), which is the fraction of all instances correctly categorized. "Average Mem." is the average memory cost for factorizing a matrix. For comparison, the average memory costs corresponding to the situation where AMD ordering is always applied and an oracle situation are also provided. The latter stands for the ideal scenario where the best among the three orderings is applied for each matrix. However, the knowledge of the best ordering algorithm is hardly achievable in practice.

TABLE IV
RESULTS OF THE SVM BASED CLASSIFIERS ON THE TESTING SUBSETS

		Time (s)	Accuracy (%)	Average Mem.(MB)	Average Mem.(MB) AMD	Average Mem.(MB) Oracle
Dataset1	Classifier1	12.7	95.7	2265	3232	2192
	Classifier2	12.7	94.4	2266		
Dataset2	Classifier1	19.1	96.4	2240	3251	2168
	Classifier2	19.1	95.0	2242		

From the results we observe the high accuracy of inference. Compared to the total runtime for factorizing all matrices in the testing subset of Dataset1 (ordered by AMD), which is 45.0 seconds, the inference costs much less time. Notice that it can be further reduced if the feature selection and SVM classifier are implemented in C. Compared with the conventional approach where the AMD algorithm is applied to all cases, the proposed method reduces the memory cost by 30%. It is only slightly larger than the oracle case. For a brute-force approach where both ordering algorithms (i.e., AMD and $metis$) and Cholesky factorizations are run, it costs 67.2 seconds. Accordingly, the proposed method only consumes 47.0 seconds. These are the benefits of the proposed approach for automatically selecting the matrix ordering algorithm.

For the MLP-NN based classifiers for optimal fill-ins and memory usage, their performance are very similar to those based on SVM, in terms of inference time and accuracy. And, more experiments are carried out to evaluate how the size of training subset affects the performance of the classifier [25]. The results show that the accuracy slowly decreases as the training subset becomes smaller. However, even trained with only 133 matrices both classifiers can have an accuracy around

93%. This implies that the proposed approach is very stable in terms of training set size.

C. An A*-Search Inspired Approach

Efficiency of the supervised learning based approach depends on existing heuristic techniques for matrix reordering, which care more about the runtime instead of the performance of outputted matrix order. Nowadays, the power grid analysis consumes a large portion of time for VLSI design. To find the worst-case scenario a large number of simulations are executed, and a power grid endures successive modifications during the design cycle. So, a better matrix order is beneficial across many simulations, as long as the power grid's structure and partitioning which determine the sparse patterns do not change. In this sense, pursuing a slower reordering approach is meaningful if it could bring remarkably fewer fill-ins.

We make an analogy between the matrix reordering problem and the traveling salesman problem, so that some AI technique for path search can be borrowed to produce a better matrix order. One of them, which is figured out to be appropriate, is the A* search algorithm [26].

The problem can be regarded as finding a node order (node sequence) for a undirected graph. This is a tree-search problem if each tree node is a partial node sequence (root node is a null sequence, and a child node is grown by appending a graph node to its parent node's sequence). The search process is branching and traversing the tree. So, a complete sequence is a leaf node in this tree, and the search process is finding a path from the root node to the leaf node. Exhaustive search for a sequence with the fewest fill-ins (scores) is obviously NP-hard. The idea of A*-search is at each intermediate tree node determining the order of search (augmenting the sequence) through estimating the final scores. This makes the paths with fewer fill-ins are first searched, and with a global record of minimum #fill-in some later tree-searches can be pruned to reduce the total computation. Inspired by this idea, we estimate the final #fill-in as the sum of #fill-in happened for attaining current incomplete sequence (intermediate tree node) and an evaluation of future fill-ins. We use the AMD algorithm performed on the unfactorized submatrix (or remaining graph) to make this evaluation, as it runs fast and is close to optimal. And, each time we only append the node with the fewest fill-ins estimation instead of trying all branches.

We have implemented the algorithm in C based on [24]. The test matrices are from the dataset described in last subsection. An experimental result is listed in Table V, from which we see that A*-search inspired approach runs very slow. So, we further improve it with randomized technique, i.e. randomly appending 32 nodes and choose the one with the minimum fill-in estimation. As shown in Table V, this improved version produces more than $5,000\times$ speedup. More results compared with AMD are given in Table VI. We see the fill-ins can be reduced by up to 20%, while the runtime is just several minutes. The reduction ratio of fill-ins is basically proportional to the increase ratio of nonzeros in Cholesky factorization after using the AMD ordering.

TABLE V
PERFORMANCE OF THE A*-SEARCH INSPIRED APPROACH AND ITS IMPROVED VERSION ON A SUBDOMAIN MATRIX WITH $N_n = 26,817$

A*-Search Version	Time(s)	#Fill-in
Original	345,464	216,518
Improved	630	217,149

TABLE VI
PERFORMANCE OF THE IMPROVED A*-SEARCH INSPIRED APPROACH

Matrix	N_n	N_e	AMD	Improved A*-Search Approach		
			#Fill-in	Time(s)	#Fill-in	Reduction
1	16,723	39,288	79,631	189	77,206	3.1%
2	28,429	67,297	186,726	591	167,895	10.1%
3	18,283	42,628	175,783	265	147,295	16.2%
4	14,233	33,162	131,250	156	105,527	19.6%

ACKNOWLEDGEMENTS

The authors are grateful to Prof. Xin Li with Duke University for the discussion on AI based PG analysis. The work is partially supported by Beijing National Research Center for Information Science and Technology (BNR2019ZS01001).

REFERENCES

[1] W. Yu and X. Wang, *Advanced Field-Solver Techniques for RC Extraction of Integrated Circuits,* Springer Inc., 2014.
[2] R. Shen, S. X.-D. Tan, J. Cui, et al., "Variational capacitance extraction and modeling based on orthogonal polynomial method," *IEEE Trans. VLSI*, vol. 18, pp. 1556-1566, 2009.
[3] Y. Zhou, Z. Li and W. Shi, "Fast capacitance extraction in multilayer, conformal and embedded dielectric using hybrid boundary element method," in *Proc. DAC*, 2007, pp. 835-840.
[4] G. Rollins, (Jul. 2010), "SESS Rapid3D 20X Performance Improvement," [Online]. (Power Point slideshow, 36 pages)
[5] W. Yu, H. Zhuang, C. Zhang, G. Hu and Z. Liu, "RWCap: A floating random walk solver for 3-D capacitance extraction of VLSI interconnects," *IEEE Trans. Computer-Aided Design*, vol. 32, pp. 353-366, 2013.
[6] K. Zhai, W. Yu and H. Zhuang, "GPU-friendly floating random walk algorithm for capacitance extraction of VLSI interconnects," in *Proc. DATE*, 2013, pp. 1661-1666.
[7] C. Zhang, W. Yu, Q. Wang and Y. Shi, "Fast random walk based capacitance extraction for the 3-D IC structures with cylindrical inter-tier-vias," *IEEE Trans. Computer-Aided Design*, vol. 34, pp. 1977-1990, 2015.
[8] Z. Xu, C. Zhang and W. Yu, "Floating random walk based capacitance extraction for general non-Manhattan conductor structures," *IEEE Trans. Computer-Aided Design*, vol. 36, pp. 120-133, 2017.
[9] W. Yu, Z. Xu, B. Li and C. Zhuo, "Floating random walk based capacitance simulation considering general floating metals," *IEEE Trans. Computer-Aided Design*, vol. 37, pp. 1711-1715, 2018.
[10] H. Qian, S. R. Nassif and S. S. Sapatnekar, "Power grid analysis using random walks," *IEEE Trans. Computer-Aided Design*, vol. 24, pp. 1204-1224, 2005.
[11] K. Sun, Q. Zhou, K. Mohanram and D. C. Sorensen, "Parallel domain decomposition for simulation of large-scale power grids," in *Proc. ICCAD*, 2007, pp. 54-59.
[12] J. M. Silva, J. R. Phillips and L. M. Silveira, "Efficient simulation of power grids," *IEEE Trans. Computer-Aided Design*, vol. 29, pp. 1523-1532, 2010.
[13] T. A. Davis and E.Palamadai Natarajan, "Algorithm 907: KLU, a direct sparse solver for circuit simulation problems," *ACM Trans. Math. Softw.*, vol. 37, pp. 36-52, 2010.
[14] J. Yang, Z. Li, Y. Cai, and Q. Zhou, "PowerRush: An efficient simulator for static power grid analysis," *IEEE Trans. VLSI*, vol. 22, pp. 2103-2116, 2014.
[15] Y. Le Coz and R. B. Iverson, "A stochastic algorithm for high speed capacitance extraction in integrated circuits," *Solid-State Electronics*, vol. 35, pp. 1005-1012, 1992.
[16] Y. Liang, W. Yu and H. Qian, "A hybrid random walk algorithm for 3-D thermal analysis of integrated circuits," in *Proc. ASP-DAC*, 2014, pp. 849-854.
[17] R. Schlott, "A Monte Carlo method for the Dirichlet problem of dielectric wedges," *IEEE Trans. Microwave Theory Tech.*, vol. 36, pp. 724-730, 2005.
[18] *RWCap*, http://numbda.cs.tsinghua.edu.cn/download/RWCap_v3.html
[19] Synopsys Inc., *Raphael*, https://www.synopsys.com/silicon/tcad/ interconnect-simulation/raphael.html.
[20] T. A. Davis, *Direct Methods for Sparse Linear Systems,* SIAM Press, 2006.
[21] P. R. Amestoy, T. A. Davis and I. S. Duff, "Algorithm 837: AMD, an approximate minimum degree ordering algorithm," *ACM Trans. Math. Softw.*, vol. 30, pp. 381-388, 2004.
[22] G. Karypis, "METIS, a software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices," University of Minnesota, Tech. Rep., 2013.
[23] I. Guyon, J. Weston, S. Barnhill and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine Learning*, vol. 46, pp. 389-422, 2002.
[24] T. A. Davis, *SuiteSparse*, http://faculty.cse.tamu.edu/davis/suitesparse. html
[25] G. Cui, W. Yu, X. Li, Z. Zeng, and B. Gu, "Machine-learning-driven matrix ordering for power grid analysis," in *Proc. DATE*, 2019, pp. 978-981.
[26] P. E. Hart, N. J. Nilsson, B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cyber.*, vol. 4, pp. 100-107, 1968.