A High-Performance Subcircuit Recognition Method Based on the Nonlinear Graph Optimization

Nikolay Rubanov

***Abstract*—Subcircuit recognition (SR) is a problem of identifying instances of a small subcircuit in a larger circuit. Despite recent progress toward linear optimization-based SR algorithms, finding a large set of subcircuits in a multimillion transistor or gate-level netlist may still be too slow for many integrated-circuit computer-aided design applications. This paper describes a new high-performance method to identify subcircuits using a nonlinear graph optimization strategy. The method uses an advanced nonlinear technique to find a global minimum of the objective function associated with the SR problem. Unlike linear graph optimization, this method does not approximate the objective function by the first-order terms in its Taylor series expansion. In contrast, to increase the recognition rate, the second-order terms are exploited to form a set of nonlinear equations that describe the net and device match probabilities. Consequently, computing the match probabilities in the new approach is based on the nonlocal structure of connections between nets and devices. An iterative nonlinear version of the Kaczmarz method (KM) is used to solve the obtained set of nonlinear equations. The KM efficiency is improved by making an important modification in its updating scheme, which leads to fast and stable convergence of the recognition process. The experimental results show that the new method is on average three times faster than linear graph optimization algorithms such as the probabilistic match assignment algorithm.**

***Index Terms*—Design, graph optimization, set of nonlinear equations, subcircuit recognition, VLSI.**

I. Introduction

TODAY'S state-of-the-art very large-scale integration circuits contain millions of transistors, and future trends indicate that chip sizes will continue to grow exponentially [1]. Multimillion circuits may contain hundreds of subcircuits and modules that represent different logic styles. The high complexity and diversity of modern integrated circuits (ICs) challenge design verification, in particular, timing analysis, full-chip simulation, design connectivity validation, and others [2], [3]. A conventional way to handle a complex design verification problem is to exploit the "divide and conquer" strategy. A powerful technique associated with this strategy is identifying specified subcircuits and blocks. To efficiently use this technique, one has to use a high-performance (in terms of runtime and accuracy) subcircuit recognition (SR) algorithm.

Manuscript received March 17, 2005; revised October 16, 2005 and January 2, 2006. This work was performed while the author was with Magma Design Automation, Santa Clara, CA 95054 USA. This paper was recommended by Associate Editor M. D. F. Wong.

The author is with Cadence Design Systems, San Jose, CA 95034 USA.

Digital Object Identifier 10.1109/TCAD.2006.881335

SR programs have been widely used in IC computer-aided design (CAD) for about three decades [3]–[10]. Traditional SR algorithms are based on search-oriented subgraph isomorphism methods [3]–[5], [7]. The two most popular search algorithms are breadth-first search (BFS) with the application of hashing functions [3], [4], and depth-first search with backtracking [3], [5]. The search methods are optimal in the sense of the solution they find. Unfortunately, they are computationally demanding because 1) they can identify only one instance of one subcircuit at a time and 2) they use a trial-and-error technique. As a result, search-oriented methods may be very slow for large and highly symmetrical circuits.

Recently, an efficient SR method–the probabilistic match assignment algorithm (PMAA)–was proposed [9], [10]. Unlike the search algorithms, PMAA is based on a linear graph optimization strategy. It constructs a nonlinear objective function corresponding to the SR task, linearizes it, and then computes a match matrix M in an iterative minimization of the objective function. The matrix M describes matches between nets and devices in a subcircuit and circuit. PMAA is orders of magnitude faster than the search-oriented algorithms; generally, it reduces the runtime for identifying all instances of a subcircuit in a multimillion circuit from hours to tens of minutes.

Unfortunately, PMAA may still be too slow for many IC CAD applications. In particular, in transistor-level circuit timing characterization a user commonly specifies a generic library of template subcircuits to find and simulate under different operating conditions [11], [12]. The template library for large multimillion transistor designs can contain more than a hundred logic gates. In this case, using PMAA to identify the gates may take many hours of computation. As a consequence, the computational cost of the SR becomes the bottleneck in the characterization process. To keep the SR runtime at a reasonable level, the user has to select a small subset of the template library and use a circuit traversing technique to extract the gate level models [12]. Unfortunately, circuit traversing cannot handle complex circuits such as sequential and analog blocks.

Similar to transistor-level timing analysis in hierarchical layout versus schematic (HLVS) verification, many subcircuits derived from a schematic have to be recognized in a circuit derived from a layout [3], [7]. Due to the high computational complexity of the SR, an error-prone user intervention (for instance, manual device and net labeling) is commonly required to speed up HLVS validation.

This paper describes a new high-performance SR method using a nonlinear graph optimization technique. Like PMAA, the new method minimizes the objective function associated with the SR task. However, unlike the standard (linear)

0278-0070/$20.00 © 2006 IEEE

Authorized licensed use limited to: Korea Advanced Inst of Science & Tech - KAIST. Downloaded on January 31,2024 at 05:09:23 UTC from IEEE Xplore. Restrictions apply.

optimization used in PMAA, it does not approximate the objective function by the first-order terms in its Taylor series expansion. In contrast, to speed up the matching process, the second-order terms are exploited to form a set of nonlinear equations with respect to the match matrix M. Consequently, computing M in the new algorithm is based on an analysis of the nonlocal structure of the net and device connections, whereas computing M in PMAA is based on the local structure of the connection. A nonlinear version of the iterative Kaczmarz method (KM) [13], [14] is used to solve the obtained set of nonlinear equations. The KM efficiency is improved by making an important modification in its updating technique. The modified KM can be viewed as an extension of the updating algorithm in the linear case. This leads to fast and stable convergence of the optimization procedure to a resulting matrix M. The experimental results show that the new method is on average three times faster than PMAA.

This paper is organized as follows. Section II reviews the graph optimization strategy for identifying subcircuits. Section III first considers a general structure of the new method and then presents a nonlinear version of the KM to solve the set of nonlinear equations associated with the SR task. Section IV discusses the experimental results obtained with the new SR method.

II. Basics of Graph Optimization Method for Identifying Subcircuits

A. Objective Function for SR Task

A bipartite graph (BG) is a convenient way to represent a circuit in graph form. A graph consists of a set of vertices and a set of vertex pairs (edges). A graph is said to be bipartite if there is a partition of its vertices into two subsets such that the ends of an edge are in distinct subsets. In this way, a circuit may be modeled by a BG with the first vertex partition representing devices and the second one representing nets. Thus, SR can be considered the task of finding images of a small model BG describing a subcircuit in a larger object BG describing a circuit.

As mentioned previously, a key idea of optimization-based SR is to compute a vertex match matrix M in iterative minimization of the objective function associated with the SR task. Generally, M is a rectangular matrix with the following properties: 1) the matrix elements $M_{ai} \in \{0, 1\}$; 2) the number of columns (rows) equals the number of subcircuit (circuit) nets and devices; and 3) the columns of M add up to one or zero. The above properties allow M_{ai} to be treated as the match probability between the vertex a in the model graph and the vertex i in the object graph. In other words, M describes the probabilistic matching between nets and devices in a subcircuit and circuit.

The objective function for an SR incorporates the information needed to compute M. First, searching for the correct vertex correspondence can be viewed as minimizing a distance between the model and object graphs. Therefore, it is reasonable to form the objective function using the graph distance concept. In an SR, the distance measure is based on the following rule: Model graph vertices a and b can be matched with object graph vertices i and j, respectively, if an edge is between a and b in the model graph and an edge is between i and j in the object graph. If the vertices a, b are matched with the vertices i, j, respectively, the distance between the model and object graphs decreases by one. Second, the SR becomes an easier task if some of the model and object graph vertices carry attributes (or labels). The vertex labels represent the device and net properties. Therefore, to speed up the SR, the objective function takes into account the vertex labels. Third, the objective function includes the constraints imposed on M.

Considering the above background, here is a formal description of the objective function for the SR, beginning with some notations used throughout this paper. Suppose that $G(V, E)$ is a graph, where V is the set of vertices and $E \subseteq V \times V$ is the set of edges. Vertices g and h are said to be adjacent if the set (g, h) is an edge. The set of neighbors (neighborhood) N_g of a vertex g is the set of vertices adjacent to g. If (g, h) is an edge then g and h are called its endpoints. If vertex g is an endpoint of edge e, then g is said to be incident on e. The degree D_g of a vertex $g \in G(V, E)$ is the number of edges incident to g. The adjacency matrix B of a graph $G(V, E)$ is a $|V| \times |V|$ matrix such that its elements $B_{ij} = 1$, if $(i, j) \in E$, and $B_{ij} = 0$, otherwise. A graph $G(X, Y; E)$ is said to be bipartite with partitions X and Y if each edge $e \in E$ has one endpoint in X and one endpoint in Y.

Suppose that $G_S(V_S, E_S)$, $G_C(V_C, E_C)$ are the model and object BGs, respectively, and B_S, B_C are the adjacency matrices associated with $G_S(V_S, E_S)$ and $G_C(V_C, E_C)$, respectively. Let $V_S = X_S \cup Y_S$, $V_C = X_C \cup Y_C$, where X_S, $X_C(Y_S, Y_C)$ are the vertices corresponding to the subcircuit and circuit nets (devices), respectively. Then, the SR problem can be formulated as the following minimization task for the match matrix M [15], [16]:

$$\text{find } M^{\text{opt}} = \min_M E(M) \text{ with} \tag{1a}$$

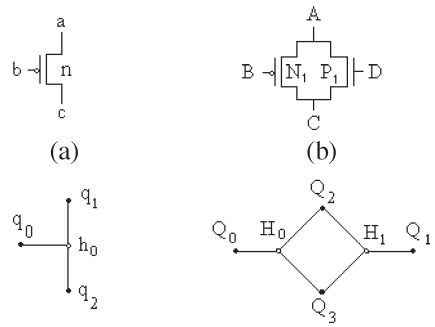
$$E(M) = -\frac{1}{2}\sum_{ai,bj} M_{ai} M_{bj} c_{ai,bj} - \lambda \sum_{ai} A_{ai} M_{ai} \tag{1b}$$

$$\text{subject } \sum_a M_{ai} \le 1, \quad \sum_i M_{ai} \le 1, \quad M_{ai} = \{0, 1\} \quad a, b = 1, \ldots, |V_S|, i, j = 1, \ldots, |V_C| \tag{2}$$

where $c_{ai,bj} = B_{S,ab} B_{C,ij}$ are the elements of the compatibility matrix C and $\lambda \ge 0$. The first sum in (1b) describes the graph distance between $G_S(V_S, E_S)$ and $G_C(V_C, E_C)$. According to this value, M should yield the largest number of the terms $M_{ai} M_{bj} c_{ai,bj}$ equal 1. The elements A_{ai}, $a \in V_S$, and $i \in V_C$ in the second sum in (1b) describe the match probabilities based on the correspondence of the vertex labels l_a and l_i. In the simplest case $A_{ai} = 0$, if $l_a \ne l_i$; $A_{ai} = 1$, if $l_a = l_i$. The vertex labels are assigned using a labeling algorithm developed in [17]. Note that the second sum can be viewed as a label-based graph distance measure. That is, matching vertices a and i with $l_a = l_i$ decreases the distance between the graphs by λ.

A simple illustrative example of the match matrix and objective function for the SR is given in Fig. 1. The subcircuit

Authorized licensed use limited to: Korea Advanced Inst of Science & Tech - KAIST. Downloaded on January 31,2024 at 05:09:23 UTC from IEEE Xplore. Restrictions apply.



	Q_0	Q_1	Q_2	Q_3	H_0	H_1
q_0	1	0	0	0	0	0
q_1	0	0	1	0	0	0
q_2	0	0	0	1	0	0
h_0	0	0	0	0	1	0

(e)

Fig. 1. Illustration of match matrix and objective function for SR: (a) subcircuit, (b) circuit, (c) model graph, (d) object graph, and (e) match matrix M.

consists of one n-type transistor [Fig. 1(a)], and the circuit is the pass gate [Fig. 1(b)]. The model and object BGs are shown in Fig. 1(c) and (d), respectively. In both the BGs, the vertices corresponding to the nets and devices are represented by black and white circles, respectively. Suppose that: 1) the transistor source and drain connections in the subcircuit and circuit (i.e., the vertices q_1, q_2, Q_2, and Q_3) have identical labels l_s; 2) the transistor gate connections (i.e., the q_0, Q_0, and Q_1) have identical labels l_g; 3) the n-type transistors (i.e., the vertices h_0 and H_0) have identical labels l_n; 4) the p-transistor (i.e., the vertex H_1) has label l_p; and 5) the net vertices cannot be matched with the device vertices. Then, the objective function can be written as

$$E(M) = -\frac{1}{2}\left(\sum_{i=0}^{2} E_{q_i}(M) + E_{h_0}(M)\right) - \lambda\left(\sum_{j=0,1} M_{q_0Q_j} + \sum_{i=1,2}\sum_{j=2,3} M_{q_iQ_j} + M_{h_0H_0}\right) \quad (3a)$$

$$\text{subject} \sum_{i=0}^{2} M_{q_iQ_j} \le 1, \qquad M_{q_iQ_j} = \{0,1\}$$
$$i = \overline{0,2}, \quad j = \overline{0,3}, \quad M_{h_0H_k} = \{0,1\}, \qquad k = 0,1 \quad (3b)$$

where $E_{q_i}(M) = \sum_{j=0,2,3} M_{q_iQ_j} M_{h_0H_0} + \sum_{j=1,2,3} M_{q_iQ_j} \times M_{h_0H_1}$, and $E_{h_0}(M) = \sum_{i=0}^{2} E_{q_i}(M)$. The match matrix M in Fig. 1(e) describes the correspondence between the nets and devices in the subcircuit and circuit.

To simplify the optimization problem (1), (2), several modifications are required. The first modification is to turn the inequality constraints into the equality ones. A standard linear programming technique adds one slack vertex to both graphs [18]. Accordingly, one slack row and column are added to the matrices M, B_S, and B_C. The second modification is to reformulate (1) from matrices containing binary values (0s and 1s) to matrices containing continuous values in the range [0, 1]. As a result

$$\text{find } \overline{M}^* = \min_{\overline{M}} \left(-\frac{1}{2}\sum_{ai,bj} \overline{M}_{ai}\overline{M}_{bj}c_{ai,bj} - \lambda\sum_{ai} A_{ai}\overline{M}_{ai}\right)$$
$$\text{subject} \sum_{a} \overline{M}_{ai} = 1, \quad \sum_{i} \overline{M}_{ai} = 1, \quad \overline{M}_{ai} \ge 0$$
$$a, b = 1, \ldots, |V_S|+1, \quad i, j = 1, \ldots, |V_C|+1 \quad (4)$$

where $\overline{M}$ denotes the match matrix augmented with one slack row and column.

Further, the constraints in (2) have to be incorporated into $E(M)$ as the penalty terms. The penalty terms construction problem is fairly complex; a consideration of it is beyond the scope of this paper and can be found elsewhere [9], [15], [16]. The resulting objective function can be written as

$$\text{find } \overline{M}^* = \min_{\overline{M}} E(\overline{M}) \quad (5)$$

where

$$E(\overline{M}) = -\frac{1}{2}\sum_{ai,bj} \overline{M}_{ai}\overline{M}_{bj}c_{ai,bj} - \lambda\sum_{ai} A_{ai}\overline{M}_{ai} + \frac{1}{\beta}\sum_{ai} \overline{M}_{ai}\log\overline{M}_{ai} + \sum_{a}\mu_a\left(\sum_{i}\overline{M}_{ai} - 1\right) + \sum_{i}\nu_i\left(\sum_{a}\overline{M}_{ai} - 1\right), \qquad a, b = 1, \ldots, |V_S|+1$$
$$i, j = 1, \ldots, |V_C| + 1. \quad (6)$$

Note that the match matrix constraints are represented in (6) by three penalty terms. The first penalty term, with the parameter β, ensures that elements of $\overline{M}$ are in the range [0, 1]. Moreover, as β increases, the values $\overline{M}_{ai}$, $\forall a$, i, approach 0s and 1s. Two other terms with the parameters μ_a and ν_i enforce the "add-up-to-one" constraint.

B. Linear Graph Optimization Approach to Minimizing Objective Function

To efficiently minimize the nonlinear objective function (6), a standard graph optimization method such as PMAA first reduces (1b) to a linear form using the first-order Taylor series expansion. The linearized objective function is

$$E(\overline{M}) = -\sum_{ai,bj} \overline{M}_{ai}\sigma_{bj}c_{ai,bj} + \frac{1}{2}\sum_{ai,bj}\sigma_{ai}\sigma_{bj}c_{ai,bj} - \lambda\sum_{ai} A_{ai}\overline{M}_{ai} + \sum_{a}\mu_a\left(\sum_{i}\overline{M}_{ai} - 1\right) + \sum_{i}\nu_i\left(\sum_{a}\overline{M}_{ai} - 1\right) + \frac{1}{\beta}\sum_{ai}\overline{M}_{ai}\log\overline{M}_{ai} \quad (7)$$

Authorized licensed use limited to: Korea Advanced Inst of Science & Tech - KAIST. Downloaded on January 31,2024 at 05:09:23 UTC from IEEE Xplore. Restrictions apply.

where the matrix σ is an approximation of $\overline{M}$. Then, alternately minimizing (7) by taking partial derivatives with respect to $\overline{M}$, μ, ν, and σ and gradually increasing β, the following system to update $\overline{M}$ is obtained [9], [16]:

$$\frac{1}{\beta}\left(1+\log\overline{M}_{ai}^{n+1}\right)=\sum_{bj}\overline{M}_{bj}^{n}c_{ai,bj}+\lambda A_{ai} \tag{8a}$$

where $\overline{M}_{bj}^{n}\equiv\sigma_{bj}$

$$\overline{M}_{ai}^{n+2}=\frac{\overline{M}_{ai}^{n+1}}{\sum\limits_{a}\overline{M}_{ai}^{n+1}}\ \text{(column normalization)}$$

$$\overline{M}_{ai}^{n+3}=\frac{\overline{M}_{ai}^{n+2}}{\sum\limits_{i}\overline{M}_{ai}^{n+2}}\ \text{(row normalization)}. \tag{8b}$$

Thus, a linear graph optimization algorithm for the SR can be generally described as follows: Start with some valid initialization of $\overline{M}$ and parameters λ, β. Then, update $\overline{M}$ using (8a), normalize $\overline{M}$ by (8b), increase β and substitute the resulting matrix $\overline{M}$ back in (8a). The previous optimization process is iteratively repeated until some convergence criterion is satisfied. It could be proven that the updating scheme (8a), (8b) converges, i.e., (7) decreases at every fixed value of β [15].

It is important to note that reducing a nonlinear minimization task to a linear one loses some information about the nonlocal structure of the vertex connections. Consequently, the SR needs more iterations to explore the nonlocal vertex neighborhood and form the correct vertex match probabilities.

To reduce the memory requirements and computational complexity of the SR process, remember that the graphs $G_S(V_S,E_S)$ and $G_C(V_C,E_C)$ are the bipartite ones. As a consequence, the matrices B_S, B_C, $\overline{M}$, and A can be presented in a block form [9]. In particular

$$A=\left[\begin{array}{c|c}A^N & 0\\ \hline 0 & A^D\end{array}\right],\quad \overline{M}=\left[\begin{array}{c|c}\overline{M}^N & 0\\ \hline 0 & \overline{M}^D\end{array}\right] \tag{9}$$

where A^N and $\overline{M}^N$ are the matrices of size $(|X_S|+1)\times(|X_C|+1)$, and A^D and $\overline{M}^D$ are the matrices of size $(|Y_S|+1)\times(|Y_C|+1)$. The matrices $\overline{M}^N(\overline{M}^D)$ and $A^N(A^D)$ describe the match probabilities and label correspondence, respectively, for the vertices associated with nets (devices). The previous matrix representation means that optimizing $\overline{M}$ can be turned into the task of alternating the optimization of $\overline{M}^N$ and $\overline{M}^D$

$$\text{find}\ \overline{M}^{N^*},\overline{M}^{D^*}=\min_{\overline{M}^N,\overline{M}^D}\left(E(\overline{M}^N)+E(\overline{M}^D)\right) \tag{10}$$

where $E(\overline{M}^N)$ and $E(\overline{M}^D)$ are described by (6).

To achieve an accurate SR, two pattern recognition techniques–error propagation and delayed decision making–are commonly incorporated into the matching process [9], [19]. The error propagation principle is used to propagate information about incompatible vertex pairs in the match matrix construction. A pair of vertices–one from the model graph and another from the object graph–is incompatible if the vertices must not be matched. Thus, after matching for a vertex pair is rejected, that pair can be used to find other incompatible vertex pairs. More specifically, first, initially incompatible vertex pairs must be obtained based on an analysis of the device and net properties. For instance, two vertices h and g, $h\in Y_S$, $g\in Y_C$, are incompatible if they correspond to devices of different types. Furthermore, two net vertices $h\in X_S$, $g\in X_C$ are incompatible if h is linked to i vertices (devices) of type t via a terminal class w (for example, transistor source, drain, or gate), but g is linked to $j<i$ vertices of the same type t via the same terminal class w. Then, given a set of initially incompatible vertex pairs, a vertex pair (u,ν) is marked incompatible if there exists a vertex $u_k\in N_u$ such that the pairs (u_k,ν_l), $\forall\,\nu_l\in N_\nu$, must not be matched.

To illustrate the error propagation principle, consider the subcircuit and circuit in Fig. 1. According to the above rules, the vertex pairs (q_1,Q_0), (q_1,Q_1), (q_2,Q_0), (q_2,Q_1), (q_0,Q_2), (q_0,Q_3), (h_0,H_1) can be marked as initially incompatible. That immediately demonstrates that the vertex pair (q_0,Q_1) is incompatible. As a consequence, the objective function (3a) can be rewritten

$$E(\overline{M}^N,\overline{M}^D)=-\frac{1}{2}\left(\sum_{i=0}^{2}E_{q_i}(\overline{M}^N,\overline{M}^D)+E_{h_0}(\overline{M}^N,\overline{M}^D)\right)-\lambda E_l(\overline{M}^N,\overline{M}^D) \tag{11}$$

where

$$E_{q_0}(\overline{M}^N,\overline{M}^D)=\overline{M}^N_{q_0Q_0}\overline{M}^D_{h_0H_0}$$

$$E_{q_i}(\overline{M}^N,\overline{M}^D)=\sum_{j=2,3}\overline{M}^N_{q_iQ_j}\overline{M}^D_{h_0H_0},\qquad i=1,2$$

$$E_{h_0}(\overline{M}^N,\overline{M}^D)=\sum_{i=0}^{2}E_{q_i}(\overline{M}^N,\overline{M}^D)$$

and

$$E_l(\overline{M}^N,\overline{M}^D)=\overline{M}^N_{q_0Q_0}+\sum_{i=1,2}\sum_{j=2,3}\overline{M}^N_{q_iQ_j}+\overline{M}^D_{h_0H_0}.$$

The soft decision-making technique is exploited to postpone forming the match probabilities for subcircuit external nets. A subcircuit net is internal if it cannot be connected to a device outside the subcircuit; otherwise the net is external. Suppose that: 1) q is a model graph vertex associated with an external net and q' is its image in the object graph and 2) $D_q\ll D_{g'}$; then, the connection of q' to a large number of extra devices may lead to a recognition error. Using the soft decision-making method allows correct matching of q and q'. This method works as follows. The beginning of the iterative SR process ignores the match matrix elements that correspond to external nets. As a consequence, the external nets do not influence matching of the devices being their neighbors. Computing the match probabilities for the external nets can be started when the match probabilities for the neighbors are close to optimal values.

Authorized licensed use limited to: Korea Advanced Inst of Science & Tech - KAIST. Downloaded on January 31,2024 at 05:09:23 UTC from IEEE Xplore. Restrictions apply.

III. Efficient SR Based on Nonlinear Graph Optimization

A. General Structure of Nonlinear Graph Optimization Algorithm for SR

A major drawback of a standard optimization-based SR method is the linearization of the objective function. Therefore, a key element in improving the recognition results is to form a set of optimization equations for $\overline{M}^N$ and $\overline{M}^D$ that takes into account the nonlocal structure of the vertex connections, i.e., the second-order terms in (6). The set of nonlinear equations then requires a solution method that will provide fast and stable computation of $\overline{M}^N$ and $\overline{M}^D$.

Here is a more formal description of the nonlinear graph optimization algorithm. A straightforward way to find $\overline{M}^{N^*}$, $\overline{M}^{D^*}$ in (10) is to perform alternating minimization of $E(\overline{M}^N), E(\overline{M}^D)$ by taking partial derivatives with respect to $\overline{M}^N$, $\overline{M}^D$, μ_a, and ν_i. Note that the objective function terms that contain the parameters μ_a and ν_i are linear functions with respect to the elements of $\overline{M}^N$, $\overline{M}^D$. In addition, the equations to enforce constraints corresponding to these terms can be considered independent of the equations to update $\overline{M}^N$ and $\overline{M}^D$ in minimizing the graph distance (1b). Therefore, the optimization of $E(\overline{M}^N), E(\overline{M}^D)$ with respect to μ_a, and ν_i in the new algorithm is the same as in a linear SR algorithm. This optimization is reduced to the column–row normalization (8b). Because that satisfies the match matrix constraints, it follows that the parameters μ_a and ν_i can be dropped from (6) (i.e., set $\mu_a = 0$, and $\nu_i = 0$) when minimizing with respect to $\overline{M}^N$ and $\overline{M}^D$. As a consequence, the nonlinear SR algorithm can be generally considered a two-step iterative procedure. The first step is to solve the following equations:

$$\frac{\partial\left(E(\overline{M}^N|\beta,\mu=0,\nu=0)\right)}{\partial\overline{M}^N_{ai}}=0$$
$$\frac{\partial\left(E(\overline{M}^D|\beta,\mu=0,\nu=0)\right)}{\partial\overline{M}^D_{bj}}=0 \tag{12}$$

where $a=1,\ldots,|X_S|+1,\ i=1,\ldots,|X_C|+1,\ b=1,\ldots,|Y_S|+1,\ j=1,\ldots,|Y_C|+1$. The second step is to perform the match matrix normalization. After each iteration, increase the convergence parameter β and repeat the previous steps.

Now, consider the first step in more detail. Computing the derivatives in (12) leads to the following set of nonlinear equations:

$$-\sum_{bj}\overline{M}^D_{bj}c_{ai,bj}-\lambda A^N_{ai}+\frac{1}{\beta}\left(1+\log\left(\overline{M}^N_{ai}\right)\right)=0 \tag{13a}$$

$$-\sum_{ai}\overline{M}^N_{ai}c_{ai,bj}-\lambda A^D_{bj}+\frac{1}{\beta}\left(1+\log\left(\overline{M}^D_{bj}\right)\right)=0. \tag{13b}$$

Comparison of (13a) and (13b) with (8a) demonstrates the main difference between the nonlinear and linear methods. According to (13a), (13b) updates of the matrices $\overline{M}^N$ and $\overline{M}^D$ are coordinated. For example, updating an element $\overline{M}^N_{ai}$ in (13a) is coordinated with updates of the elements $\overline{M}^D_{bj}$, $\forall b\in N_a$, $\forall j\in N_i$. Therefore, the updating technique tends to ensure correspondence between the match probabilities for the devices connected to the nets a and i. In other words, the device match probabilities (i.e., the elements $\overline{M}^D_{bj}$, $\forall b\in N_a$, $\forall j\in N_i$) are determined by the nonlocal structure of the device connections. Similarly, in solving (13b), updating an element $\overline{M}^D_{bj}$ is coordinated with updates of the elements $\overline{M}^N_{ai}$, $\forall a\in N_b, \forall i\in N_j$; therefore, the net match probabilities (i.e., the elements $\overline{M}^N_{ai}, \forall a\in N_b, \forall i\in N_j$) are determined by the nonlocal structure of the net connections. In a linear optimization-based SR algorithm, updating the elements $\overline{M}^N_{ai}$ $(\overline{M}^D_{bj})$ at the current iteration is determined by the values of $\overline{M}^D_{bj}$, $\forall b\in N_a, \forall j\in N_i$, $(\overline{M}^N_{ai}, \forall a\in N_b, \forall i\in N_j)$ obtained at the previous iteration. Furthermore, updating the elements $\overline{M}^N_{ai}$ $(\overline{M}^D_{bj})$ does not affect the elements $\overline{M}^D_{bj}$ $(\overline{M}^N_{ai})$. Moreover, the elements of $\overline{M}^N$ are updated independently of one another; the same is true for the elements of $\overline{M}^D$. Summarizing, in contrast to (8a), updating $\overline{M}^N$ and $\overline{M}^D$ in (13a) and (13b) takes into account the nonlocal vertex neighborhood. This reduces the number of iterations and, thus, significantly speeds up the SR.

Here is an example using (13a) and (13b) for the subcircuit and circuit in Fig. 1. Incorporating the penalty terms into (11) and computing the partial derivatives with respect to $\overline{M}^N$, $\overline{M}^D$ yields

$$\begin{aligned}
&-\overline{M}^D_{h_0H_0}-\lambda+\frac{1}{\beta}\left(1+\log\left(\overline{M}^N_{q_0Q_0}\right)\right)=0\\
&-\overline{M}^D_{h_0H_0}-\lambda+\frac{1}{\beta}\left(1+\log\left(\overline{M}^N_{q_iQ_j}\right)\right)=0 \qquad i=1,2,\ j=2,3\\
&-\overline{M}^N_{q_0Q_0}-\sum_{i=1,2}\sum_{j=2,3}\overline{M}^N_{q_iQ_j}-\lambda+\frac{1}{\beta}\left(1+\log\left(\overline{M}^D_{h_0H_0}\right)\right)=0.
\end{aligned} \tag{14}$$

With the above background, the nonlinear optimization algorithm for the SR can be presented as follows: Suppose that β_0, $\beta_{\max}$ are, respectively, the initial and maximum values of the parameter β, and β_r is a weighting factor. Suppose that $SA_{\max}$ is the maximum value of the iteration counter SA. Then, a general form of the algorithm is described in Fig. 2. In this algorithm, initial values are 1) $\beta_0=1.0$, $\beta_r=1.2$, $\beta_{\max}=2.0$; 2) $\lambda=0.75$; 3) $SA_{\max}$ to 5; and 4) $\overline{M}^N_{ai}$ to $(1+\varepsilon_{ai})/|V_C|$ and $\overline{M}^D_{cl}$ to $(1+\varepsilon_{cl})/|V_C|$, where ε_{ai}, ε_{cl} are random values in the range $[-0.1, 0.1]$.

Here are some comments related to the algorithm initialization. First, the values for β_0, $\beta_{\max}$, β_r, $SA_{\max}$ are chosen according to the following requirements: 1) the match matrix elements should all have roughly the same values when $\beta=\beta_0$; 2) the resulting match matrices should represent all subcircuit instances in a circuit when $\beta=\beta_{\max}$; and 3) the value β_r should provide gradual progress toward the match matrices consisting of zeros and ones. The gradual progress reduces the

Authorized licensed use limited to: Korea Advanced Inst of Science & Tech - KAIST. Downloaded on January 31,2024 at 05:09:23 UTC from IEEE Xplore. Restrictions apply.

Algorithm: The SR algorithm based on the nonlinear graph optimization

1. Initialize $\overline{M}_{ai}^{N}$, $\overline{M}_{bj}^{D}$, λ, β_0, $\beta_{\max}$, β_r, $SA_{\max}$.

2. $\beta = \beta_0$, $n = 0$.

3. **While** ($\beta \le \beta_{\max}$)

3.1 Solve the set of nonlinear equations with respect to $\overline{M}_{ai}^{N,(n)}$ and $\overline{M}_{cl}^{D,(n)}$:

$$-\sum_{bj}\overline{M}_{bj}^{D,(n)}c_{ai,bj}-\lambda A_{ai}^{N}+\frac{1}{\beta}(1+log(\overline{M}_{ai}^{N,(n)}))=0$$

$$-\sum_{ai}\overline{M}_{ai}^{N,(n)}c_{ai,bj}-\lambda A_{bj}^{D}+\frac{1}{\beta}(1+log(\overline{M}_{bj}^{D,(n)}))=0$$

3.2 While ($SA \le SA_{\max}$)

3.2.1 Update $\overline{M}^{N,(n)}$ and $\overline{M}^{D,(n)}$ by the matrix row normalization:

$$\overline{M}_{ai}^{N,(n)}=\frac{\overline{M}_{ai}^{N,(n)}}{\sum_{a}\overline{M}_{ai}^{N,(n)}} \qquad \overline{M}_{bj}^{D,(n)}=\frac{\overline{M}_{bj}^{D,(n)}}{\sum_{b}\overline{M}_{bj}^{D,(n)}}$$

3.2.2 Update $\overline{M}^{N,(n)}$ and $\overline{M}^{D,(n)}$ by the matrix column normalization:

$$\overline{M}_{ai}^{N,(n)}=\frac{\overline{M}_{ai}^{N,(n)}}{\sum_{i}\overline{M}_{ai}^{N,(n)}} \qquad \overline{M}_{bj}^{D,(n)}=\frac{\overline{M}_{bj}^{D,(n)}}{\sum_{j}\overline{M}_{bj}^{D,(n)}}$$

3.2.3 $SA = SA + 1$

3.3 $n = n+1$, $\beta = \beta\beta_r$

4. Reconstruct subcircuit instances from resulting matrices $\overline{M}^{N}$ and $\overline{M}^{D}$.

Fig. 2. General form of SR algorithm based on nonlinear graph optimization. Initialization values are given at end of Section III-A.

chance of getting trapped in local minima of the objective function; 4) the choice of $SA_{\max}$ should ensure stable convergence of the match matrices at every fixed value of β. Optimal values for the above parameters were found empirically during a number of experiments. Second, the matrices $\overline{M}^{N}$ and $\overline{M}^{D}$ are initialized to quasi-random numbers with an average value $1/|V_C|$. Third, the connectivity-based first sum in (1b) should dominate the label-based second sum in (1b) in the vertex matching process. Therefore, in the algorithm $\lambda = 0.75$.

B. Solving Set of Nonlinear Equations Using KM

The choice of a method to solve (13a) and (13b) is extremely important to ensure a fast and stable SR process. The choice is mainly determined by two factors. First, the method should be efficient from the viewpoint of computational complexity and memory use. Second, in the nonlinear case, it is hard to prove the convergence of the SR algorithm (i.e., the convergence of the procedure in Fig. 2), in contrast to the linear graph optimization approach. Therefore, to validate the algorithm convergence, it is desired to use a method that can be viewed as an extension of the updating technique in the linear case.

Start by considering the computational aspects of a set of nonlinear (13a) and (13b). The traditional Newton's techniques (the Newton–Raphson method, the damped Newton method, the quasi-Newton methods, and others [20], [21]) have several serious disadvantages when applied to nonlinear equations corresponding to the SR task. In particular, (13a) and (13b) are completely specified by the structure of the net and device connections in a subcircuit and circuit. Therefore, given a subcircuit/circuit pair, it is not guaranteed that the Jacobian of that set of nonlinear equations will exist at every iteration step. In addition, it may be impractical to recompute all the Jacobian entries for multimillion transistor circuits. Furthermore, an

Authorized licensed use limited to: Korea Advanced Inst of Science & Tech - KAIST. Downloaded on January 31,2024 at 05:09:23 UTC from IEEE Xplore. Restrictions apply.

appropriate initial guess is rather critical to the Newton's methods convergence. Unfortunately, such a guess is usually impossible in SR because little or no *a priori* information is available about the net and device matching. Finally, (13a) and (13b) are, in general, sparse because the circuit devices and most circuit nets have small numbers of connections.

A nonlinear version of the iterative KM is widely used to overcome computational complexity and convergence problems. This method has proved successful in many signal processing and pattern recognition applications [13], [14], [22]. The original KM is a form of backpropagation for solving linear problems [13]. Generally, it allows computation of a least-norm solution of a set of linear equations and converges with a geometrical progression rate controlled by a single relaxation parameter. The essential features of the KM and its nonlinear version are processing one equation at a time and using derivative and adjoint derivative operators to update unknowns. The line-by-line updating scheme is well suited for sparse systems and efficient from a memory-use standpoint.

Here is a brief description of the nonlinear KM properties from the viewpoint of SR algorithm convergence. As mentioned above, a key idea in solving (13a) and (13b) is using a method that can be viewed as an extension of the updating scheme in linear graph optimization. In general, a standard nonlinear version of the KM does not possess this feature. However, a modification of the KM allows it to meet that requirement. Namely, two relaxation parameters are used to control the rate at which the KM updates $\overline{M}^N$, $\overline{M}^D$ at every iteration step. More specifically, different parameters are used to update the linear and nonlinear terms in (13a) and (13b). It is important to note that this modification preserves all the original convergence properties of the KM.

With the above background, here is a formal description of the nonlinear KM. Consider a set of nonlinear equations

$$V_p(W) = 0, \qquad p = 1, \ldots, P \tag{15}$$

where V_p is a nonlinear operator, and $W \in R^s$, R^s is s-dimensional Euclidean space. Suppose that W is initialized to a vector $W^0 \in R^S$. Then, one has to find $w \in R^s$ such that $(W^0 + w)$ is an actual solution of (15). As a result, (15) are transformed into

$$V_p(W^0 + w) = 0, \qquad p = 1, \ldots, P. \tag{16}$$

The nonlinear KM computes w in (16) using the following rule:

$$w = -V_p'^{\#}(W^0) C_p^{-1} V_p(W^0) \tag{17}$$

where $C_p = V_p'(W^0) V_p'^{\#}(W^0)$, and $V_p'^{\#}$ denotes an operator adjoint to the operator V_p' [13], [14]. Thus, one sweep (iteration) of the nonlinear KM to solve (15) reads

$$\begin{aligned} W_0 &= W^i \\ \text{For } p &= 1, \ldots, P \\ W_p &= W_{p-1} - \omega_p h_p \\ W^{i+1} &= W_P \end{aligned} \tag{18}$$

where $h_p = V_p'^{\#}(W_{p-1}) C_p^{-1} V_p(W_{p-1})$, and ω_P is a relaxation parameter. If V_p is a linear operator, the set of (15) is consistent and $W^0 \in \sum_{p=1}^{P} \text{range}(V_p^{\#})$ (for example, $W^0 = 0$), then the iteration process (18) converges to a solution W^* with the least norm subject to the assumption $0 < \omega_p < 2$ [13].

Unfortunately, in a general nonlinear case, there are no simple results on the KM convergence. However, it can be explained what each sweep in the nonlinear KM will actually do. Suppose that (18) has a solution W^* and $\omega_p = \omega, \forall p$. Then

$$\begin{aligned} \|W_p - W^*\|^2 &= \|W_{p-1} - W^* - \omega h_p\|^2 \\ &= \|W_{p-1} - W^*\|^2 - 2\omega (W_{p-1} - W^*, h_p) \\ &\quad + \omega^2 \|h_p\|^2. \end{aligned} \tag{19}$$

Ignoring higher order terms in the product $(W_p - W^*, h_p)$ gives

$$(W_p - W^*, h_p) \approx \left(C_p^{-1} V_p(W_{p-1}), V_p(W_{p-1})\right). \tag{20}$$

Thus, if $V_p(W_{p-1}) \neq 0$, operator C_p is positive definite and ω is sufficiently small, W_p should be closer to W^* than W_{p-1}. Note that the norm convergence (19) is not critical to the initial guess W^0; in practice, the nonlinear KM quickly converges to a solution starting with quasi-random initial conditions [13], [14].

Now, the above theory can be used to solve (13a) and (13b). It is evident that the nonlinear operators V_p are described by the following equations:

$$\begin{aligned} V_{ai}^N(\overline{M}) &= -\sum_{bj} \overline{M}_{bj}^D c_{ai,bj} - \lambda A_{ai}^N + \frac{1}{\beta}\left(1 + \log\left(\overline{M}_{ai}^N\right)\right) \\ &\forall b \in N_a, \forall j \in N_i \\ V_{bj}^D(\overline{M}) &= -\sum_{ai} \overline{M}_{ai}^N c_{ai,bj} - \lambda A_{bj}^D + \frac{1}{\beta}\left(1 + \log\left(\overline{M}_{bj}^D\right)\right) \\ &\forall a \in N_b, \forall i \in N_j \end{aligned} \tag{21}$$

where $a = 1, \ldots, |X_S| + 1$, $i = 1, \ldots, |X_C| + 1$, $b = 1, \ldots, |Y_S| + 1$, $j = 1, \ldots, |Y_C| + 1$. Then, the operators V_p' can be written

$$\begin{aligned} V_{ai}^{N\prime}(\overline{M}^{N,n}) \overline{m}^N &= -\sum_{bj} \overline{m}_{bj}^D c_{ai,bj} + \frac{1}{\beta} \frac{\overline{m}_{ai}^N}{\left(\overline{M}_{ai}^{N,n}\right)} \\ V_{bj}^{D\prime}(\overline{M}^{D,n}) \overline{m}^D &= -\sum_{ai} \overline{m}_{ai}^N c_{ai,bj} + \frac{1}{\beta} \frac{\overline{m}_{bj}^D}{\left(\overline{M}_{bj}^{D,n}\right)}. \end{aligned} \tag{22}$$

To find the adjoint operators $V_{ai}^{N\prime\#}$ and $V_{bj}^{D\prime\#}$ use a formal scalar product relation

$$(V_p(W)w, d) = \left(w, V_p^{\#}(W)d\right), \qquad d \in R. \tag{23}$$

Authorized licensed use limited to: Korea Advanced Inst of Science & Tech - KAIST. Downloaded on January 31,2024 at 05:09:23 UTC from IEEE Xplore. Restrictions apply.

According to (23) $V_{ai}^{N\prime\#}$ and $V_{bj}^{D\prime\#}$ are vectors of the same size as $\overline{M}^N$ and $\overline{M}^D$, respectively. The vectors are sparse; their components have the following form:

$$V_{ai}^{N\prime\#}(\overline{M}^{N,n})d = \begin{cases} -c_{ai,bj}d, & \forall b \in N_a, \forall j \in N_i \\ d/\left(\beta \overline{M}_{ai}^{N,n}\right) & \\ 0, \quad \text{otherwise} & \end{cases}$$
$$V_{bj}^{D\prime\#}(\overline{M}^{D,n})d = \begin{cases} -c_{ai,bj}d, & \forall a \in N_b, \forall i \in N_j \\ d/\left(\beta \overline{M}_{bj}^{D,n}\right) & \\ 0, \quad \text{otherwise.} & \end{cases} \tag{24}$$

As a consequence, the relations for $C_{ai}^{N,n} = V_{ai}^{N\prime}(\overline{M}^{N,n}) \times V_{ai}^{N\prime\#}(\overline{M}^{N,n})$ and $C_{bj}^{D,n} = V_{bj}^{D\prime}(\overline{M}^{D,n})V_{bj}^{D\prime\#}(\overline{M}^{D,n})$ at iteration n are

$$C_{ai}^{N,n} = \sum_{bj} c_{ai,bj} + \frac{1}{\beta^2}\frac{1}{\left(\overline{M}_{ai}^{N,n}\right)^2}$$
$$C_{bj}^{D,n} = \sum_{ai} c_{ai,bj} + \frac{1}{\beta^2}\frac{1}{\left(\overline{M}_{bj}^{D,n}\right)^2}. \tag{25}$$

Substitute (21), (24), and (25) into (17) to obtain the equations to update $\overline{M}^N$ and $\overline{M}^D$ in (13a)

$$\overline{M}_{ai}^{N,n+1} = \overline{M}_{ai}^{N,n} - \omega \frac{\frac{1}{\beta \overline{M}_{ai}^{N,n}} d_{ai}^{N,n}}{C_{ai}^{N,n}} \tag{26}$$
$$\overline{M}_{bj}^{D,n+1} = \overline{M}_{bj}^{D,n} - \omega \frac{q_{bj} d_{ai}^{N,n}}{C_{ai}^{N,n}} \tag{27}$$

where $d_{ai}^{N,n} = -\sum_{bj} c_{ai,bj}\overline{M}_{bj}^{D,n} - \lambda A_{bj} + (1/\beta)(1 + \log(\overline{M}_{ai}^{N,n}))$, $q_{bj} = -1$ for $b \in N_a$, $j \in N_i$, $q_{bj} = 0$, otherwise. The equations to update $\overline{M}^N$ and $\overline{M}^D$ in (13b) can be read

$$\overline{M}_{bj}^{D,n+1} = \overline{M}_{bj}^{D,n} - \omega \frac{\frac{1}{\beta \overline{M}_{bj}^{D,n}} d_{bj}^{D,n}}{C_{bj}^{D,n}} \tag{28}$$
$$\overline{M}_{ai}^{N,n+1} = \overline{M}_{ai}^{N,n} - \omega \frac{q_{ai} d_{bj}^{D,n}}{C_{bj}^{D,n}} \tag{29}$$

where $d_{bj}^{D,n} = -\sum_{ai} c_{ai,bj}\overline{M}_{ai}^{N,n} \lambda A_{ai} + (1/\beta)(1 + \log(\overline{M}_{bj}^{D,n}))$, $q_{ai} = -1$ for $a \in N_b$, $i \in N_j$, $q_{ai} = 0$, otherwise.

One can establish relations among (26), (27), and (8a) with $\overline{M} = \overline{M}^N$; similar relations can be established among (28), (29), and (8a) with $\overline{M} = \overline{M}^D$. First, because the elements $\overline{M}_{bj}^D$, are not updated in (6) (when $\overline{M} = \overline{M}^N$) two different relaxation parameters ω' and ω'' are needed in (26) and (27). Note that use of more than one relaxation parameter does not contradict the general KM convergence result. The parameter ω' controls updating of the elements $\overline{M}_{ai}^N$, and the parameter ω'' controls updating of the elements $\overline{M}_{bj}^D$. Therefore, $\omega = \omega'$ was set in (26) and $\omega = \omega''$ in (27). As a result, (27) can be reduced to the linear case by setting $\omega'' \to 0$. Now, consider (26). According to the KM theory, the operator C_p can be replaced in (17) by an easy-to-evaluate operator $\hat{C}_p$, provided that $\hat{C}_p > C_p$ [14]. In the case here, $C_{ai}^{N,n}$ can be replaced by $\hat{C}_{ai}^{N,n}$, where $\hat{C}_{ai}^{N,n} = r(1/\beta \overline{M}_{ai}^{N,n})^2 \geq C_{ai}^{N,n}$, $\overline{M}_{ai}^{N,n} \in [0,1]$, $a = 1, \ldots, |X_S| + 1$, $i = 1, \ldots, |X_C| + 1$, and r is an appropriate factor. Then, after some obvious manipulations, (26) is transformed to following formula to update $\overline{M}^N$:

$$\frac{1}{\beta}\left(1 + \log \overline{M}_{ai}^{N,n} + \left(\overline{M}_{ai}^{N,n+1} - \overline{M}_{ai}^{N,n}\right)\frac{r}{\omega_p' \overline{M}_{ai}^{N,n}}\right) = \sum_{bj} \overline{M}_{bj}^{D,n} c_{ai,bj} + \lambda A_{ai}. \tag{30}$$

If $\omega_p\prime \approx 1$ and $r \approx 1$, then using the first-order Taylor series expansion of the log function reduces the formula (30) to (8a)

$$\frac{1}{\beta}\left(1 + \log \overline{M}_{ai}^{N,n+1}\right) \approx \sum_{bj} \overline{M}_{bj}^{D,n} c_{ai,bj} + \lambda A_{ai}. \tag{31}$$

The assumptions about ω_p' and t can be justified as follows. First, the above value of the relaxation parameter lies between two extreme values (0 and 2); that avoids too small and too large changes of $\overline{M}_{ai}^N$ and $\overline{M}_{bj}^D$. Second, the parameter r can be estimated based on the relation between $\hat{C}_{ai}^{N,n}$ and $C_{ai}^{N,n}$. If $\overline{M}_{ai}^{N,n} \approx 0$, then generally, $1/(\beta \overline{M}_{ai}^{N,n})^2 \gg \sum_{bj} c_{ai,bj}$, and, as a consequence, $r \approx 1$. If $\overline{M}_{ai}^{N,n} \approx 1$, then $r \approx \sum_{bj} c_{ai,bj}$. However, in the latter case, $r \approx 1$ can still be used. Indeed, on the one hand, setting $r \approx 1$ may lead to $C_{ai}^{N,n} < \hat{C}_{ai}^{N,n}$ and therefore, worsen the KM convergence conditions. On the other hand, setting $r \approx 1$ reduces the nonlinear updating scheme to the linear one that is provably convergent. Thus, a good convergence of the KM can be expected when $r \approx 1$.

Based on the above results, a formal algorithm for solving (13a) and (13b) can be presented as in Fig. 3. It was found empirically during a number of the experiments that setting the sweep counter $\text{swp}_{\max}$ to 1 provides good convergence of the KM. In addition, the nonlinear KM quickly converges to a solution starting with the quasi-random initial conditions given at the end of Section III-A. The matrix elements may become negative during the optimization process. Therefore, it is important to check the elements" positivity; negative matrix values are changed to 0.0001. In the algorithm, ω_p' was set to equal 0.9 and $\omega_p'' = 0.1$.

C. Construction of Subcircuit Instances Using Match Matrix-Driven BFS

The final step of the nonlinear SR algorithm is to reconstruct the subcircuit instances based on an analysis of the resulting matrices $\overline{M}^N$ and $\overline{M}^D$. This step is identical to the subcircuit instance construction (SIC) procedure developed in PMAA [9]. Here is a description of the basic principles of the SIC algorithm. Suppose that $p_m = 0.9$ is a matching threshold. That is, a match between the vertices a (b) and i (j) associated with nets (devices) is rejected if $\overline{M}_{a,i}^N(\overline{M}_{b,j}^D) < p_m$. Let u be a model graph vertex corresponding to an internal net, although all the conclusions hold when u corresponds to a device. In the first

Authorized licensed use limited to: Korea Advanced Inst of Science & Tech - KAIST. Downloaded on January 31,2024 at 05:09:23 UTC from IEEE Xplore. Restrictions apply.

Algorithm: The nonlinear Kaczmarz method to compute $\overline{M}^N$ and $\overline{M}^D$

1. Initialize ω_p', ω_p'', swp_{max}

2. $n = 0$

3. While($n < swp_{max}$)

3.1 For $b \leftarrow 1,..,|Y_S|+1,\ j \leftarrow 1,..,|Y_C|+1$

$\overline{M}_{bj}^{D,(n)} = 0$, if vertex pair (b, j) must not match

3.2 For $a \leftarrow 1,..,|X_S|+1,\ i \leftarrow 1,..,|X_C|+1$

$\overline{M}_{ai}^{N,(n)} = 0$, if vertex pair (a, i) must not match

3.3 For $b \leftarrow 1,..,|Y_S|+1,\ j \leftarrow 1,..,|Y_C|+1,\ \overline{M}_{bj}^{D,(n)} \neq 0$

3.3.1 Update $\overline{M}_{bj}^{D,(n)}$ and $\overline{M}_{ai}^{N,(n)}$ according to (26), (27)

3.3.2 Check positivity of $\overline{M}_{ai}^{N,(n)}$ and $\overline{M}_{bj}^{D,(n)}$

3.4 For $a \leftarrow 1,..,|X_S|+1,\ i \leftarrow 1,..,|X_C|+1,\ \overline{M}_{ai}^{N,(n)} \neq 0$

3.4.1 Update $\overline{M}_{bj}^{D,(n)}$ and $\overline{M}_{ai}^{N,(n)}$ according to (28), (29)

3.4.2 Check positivity of $\overline{M}_{bj}^{D,(n)}$ and $\overline{M}_{ai}^{N,(n)}$

3.5 $n = n+1$

4. $\overline{M}^N = \overline{M}^{N,(swp_{max}-1)}$, $\overline{M}^D = \overline{M}^{D,(swp_{max}-1)}$

Fig. 3. Nonlinear KM to solve set of nonlinear equations with respect to $\overline{M}^N$ and $\overline{M}^D$. Initialization values are given at end of Section III-B.

phase of the SIC procedure, analyze $\overline{M}^N$ and match u with a vertex $\nu \in G_C(V_C, E_C)$ such that $\overline{M}^N_{u,\nu} > p_m$. In the second phase of the SIC algorithm, exploit the BFS to traverse the model and object BGs to find other matching pairs. First, form the set $N_\nu = \{n_k^\nu\}$ consisting of the neighbors of ν. Because vertices in N_ν correspond to the circuit devices, analyze $\overline{M}^D$ to obtain a set of the vertices $m_k^u \in G_S(V_S, E_S)$, $\forall k$, such that $\overline{M}^D_{m_k^u, n_k^\nu} > p_m$. Accordingly, match the pairs (m_k^u, n_k^ν), and put n_k^ν, $\forall k$, into the vertex list used by the BFS to continue the matching process. Thus, the SIC procedure can be viewed as a match-matrices driven BFS.

A formal description of the general framework of the SIC procedure follows. Suppose that: 1) list U contains the object graph vertices visited during the BFS; 2) list D contains the vertices to continue the BFS; 3) list Q_S contains all the vertices corresponding to subcircuit external nets; 4) hash table T contains the matched vertex pairs; 5) given a vertex ν function FindImage(u) analyzes $\overline{M}^N$ ($\overline{M}^D$) and returns either a vertex ν being an image of u or NULL if the image is not found; and 6) function CheckSearch() returns TRUE if all the vertices in $G_S(V_S, E_S)$ are matched, otherwise it returns FALSE. Furthermore, some standard search functions will be used: 1) front(L) returns the front element in list L; 2) add(ν, L) adds vertex ν to the end of list L; 3) delete(ν, L) deletes vertex ν from list L; 4) insert(u, ν, H) inserts vertex pair (u, ν) in hash table H; and 5) clear(H) removes all elements from hash table H. The SIC procedure can then be described as in Fig. 4. For more detail on the SIC algorithm implementation, see [9].

Algorithm: The subcircuit instances construction procedure

Input: Start search vertex $u \in G_S(V_S, E_S)$, $u \notin Q_S$

Output: Matched vertex pairs stored in the hash table T

1. v = *FindImage*(u)
2. **if**(v == *NULL*)
3. **goto** line 15
4. *add*(v, D), *insert*(u, v, T)
5. **while** (D != $\varnothing$)
6. v = *front*(D)
7. *delete*(v, D), *add*(v, U)
8. **for**($n_k \in N_v$, $n_k \notin U$)
9. m_k = *FindImage*(n_k)
10. **if**(m_k == *NULL*)
11. **goto** line 15
12. *insert*(m_k, n_k, T)
13. **if** ($n_k \notin U$ && $n_k \notin D$ && $m_k \notin Q_S$)
14. *add*(n_k, D)
15. **if***(CheckSearch()*== FALSE*)*
16. *clear*(T)
17. **return** T

Fig. 4. General form of subcircuit instances construction procedure.

IV. Experimental Results

A series of experiments investigated the efficiency of the new SR algorithm. In the experiments, the SR results obtained using the new method were compared with results obtained using PMAA. Because no SR benchmark suite has been introduced for use by the design automation community, the authors evaluated the nonlinear recognition method using their own benchmark circuits. These circuits were presented in simulated program with integrated circuit emphasis (SPICE) netlist format and contained combinational gates, sequential gates (latches and flip-flops), dynamic gates, and memory cells. The routine to implement the nonlinear SR algorithm was written in C++ and, according to the procedures in Figs. 2–4, could be divided into the following parts: 1) the algorithm preprocessing, including the initialization of the match matrices and parameters and conversion of a subcircuit and circuit to

Authorized licensed use limited to: Korea Advanced Inst of Science & Tech - KAIST. Downloaded on January 31,2024 at 05:09:23 UTC from IEEE Xplore. Restrictions apply.

TABLE I
PERFORMANCE EVALUATION OF NEW SR METHOD. SUBCIRCUIT TYPES ARE COMBINATIONAL GATE (C), SEQUENTIAL GATE (S), MEMORY CELL (M). N IS NUMBER OF SUBCIRCUIT INSTANCES IN CIRCUIT. CIRCUIT SIZES ARE GIVEN IN NUMBERS OF TRANSISTORS

Circuits		*Subcircuits*			*PMAA*	*The new method*
#	Size	Type	Size	N	Runtime	Runtime
1	1508150	C	32	350	12.4m	4.5m
2	1292698	S	40	300	14.6m	5m
3	1155900	C	6	6500	7.3m	2.4m
4	1141422	C	40	398	9.5m	3.2m
5	699626	S	18	280	6.1m	2.1m
6	693540	C	4	3900	4.5m	1.4m
7	581100	S	18	100	4.7m	1.9m
8	546944	C	30	128	4.2m	1.3m
9	478848	M	6	64K	3.1m	1m
10	449760	S	14	180	4.3m	1.5m
11	396589	C	31	401	3m	1.4m
12	316305	C	13	108	112s	36s
13	305939	S	36	71	141s	47s
14	204976	S	20	557	68s	22s
15	149200	C	12	400	55s	19s

the model and object BGs, respectively; 2) solving the set of nonlinear equations; 3) match matrices normalization; and 4) BFS-based reconstruction of the subcircuit instances. Note that in solving the set of nonlinear equations precomputing and storing the compatibility matrix C (i.e., the elements $c_{ai,bj}$, $\forall a$, i, b, j) were avoided. Instead, the elements $c_{ai,bj}$ were computed as needed based on an analysis of the vertex connections in the model and object graphs. That is important because C can rapidly grow in size for large circuits.

The results of the conducted experiments are given in Table I. This table shows the subcircuit types, circuit, and subcircuit sizes in the number of transistors, the number of the subcircuit instances in the circuits, and the CPU runtimes. All experiments were performed on an Intel Pentium 4 computer running at 2 GHz.

Based on the analysis of Table I, it can be seen that similar to PMAA the runtime of the nonlinear SR algorithm mainly depends on the object circuit size and secondarily depends on the number of subcircuit instances. Therefore, the algorithms exhibit a similar scalability. According to Table I, the new method finds all the subcircuit instances an average of three times faster than PMAA. Note that 1) about 90% of the execution time was spent in solving the set of the nonlinear equations and 2) similar to PMAA, the new algorithm needs $O(k(|X_S||X_C| + |Y_S||Y_C|)/1024)$ KB of memory, where k is the number of bytes to represent the match matrix elements. The new algorithm successfully recognized all the subcircuit instances in the circuits. Commonly, an efficient graph isomorphism algorithm is used to determine whether the identified subcircuits are correct by comparing the subcircuit instances and the model subcircuit. In this paper, the Gemini II algorithm [23] was used to justify the SR results obtained using the new method.

V. CONCLUSION

Linear optimization-based SR algorithms such as PMAA [9] may still be too slow for many IC CAD applications. In this paper, we described a new, efficient SR method using nonlinear graph optimization. A key idea of the new method is to exploit the second-order terms in the objective function corresponding to the SR task. This allows formation of a set of nonlinear equations with respect to the vertex match matrices $\overline{M}^N$ and $\overline{M}^D$. As a consequence, computing $\overline{M}^N$ and $\overline{M}^D$ in the SR process takes into account the nonlocal structure of the vertex connections. To solve the set of nonlinear equations, a nonlinear version of the KM was used. To validate the new algorithm convergence, an important modification was made to the KM updating scheme. The modified KM was shown as an extension of the linear technique to update $\overline{M}^N$ and $\overline{M}^D$. The experimental results demonstrated that the new method is on average three times faster than PMAA.

REFERENCES

[1] Semiconductor Industry Association, *International Technology Roadmap for Semiconductors*, 2002 Update.
[2] B. Lu, D.-Z. Du, and S. Sapatnekar, Eds., Layout Optimization in VLSI Design. New York: Kluwer, 2001.
[3] T. S. Chanak, "Netlist processing for custom VLSI via pattern matching," Comput. Syst. Lab., Stanford Univ., Stanford, CA, Tech. Rep. CSL-TR-95-681, 1995.
[4] M. Ohlrich, C. Ebeling, E. Ginting, and L. Sather, "SubGemini: Identifying subcircuits using fast subgraph isomorphism algorithm," in *Proc. ACM/IEEE Des. Autom. Conf.*, 1993, pp. 31–37.
[5] G. Pelz and U. Röttcher, "Pattern matching and refinement hybrid approach to circuit comparison," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 10, no. 2, pp. 264–276, Feb. 1994.
[6] Z. Ling and D. Yun, "An efficient subcircuit extraction algorithm by resource management," in *Proc. 2nd IEEE Int. ASIC Conf.*, 1996, pp. 9–14.

Authorized licensed use limited to: Korea Advanced Inst of Science & Tech - KAIST. Downloaded on January 31,2024 at 05:09:23 UTC from IEEE Xplore. Restrictions apply.

[7] W. Kim and H. Shin, "Hierarchical LVS based on hierarchy rebuilding," in *Proc. Asia South Pacific Design Autom. Conf.*, 1998, pp. 379–384.
[8] L. Yang and R. Shi, "FROSTY: A fast hierarchy extractor for industrial CMOS circuits," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 2003, pp. 741–746.
[9] N. Rubanov, "SubIslands: The probabilistic match assignment algorithm for subcircuit recognition," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 1, pp. 26–38, Jan. 2003.
[10] ——, "Fast and accurate identifying subcircuits using an optimization-based technique," in *Proc. IEEE Int. Symp. Signals, Circuits, Syst.*, 2003, pp. 69–71.
[11] A. Conn *et al.*, "Gradient-based optimization of custom circuits using a static-timing formulation," in *Proc. ACM/IEEE Des. Autom. Conf.*, 1999, pp. 452–459.
[12] A. Srinivasan and H. Chaudri, "Method of incremental recharacterization to estimate performance of integrated designs," U.S. Patent 6 851 095B1, Feb. 1, 2005.
[13] Y. Censor and S. Zenios, *Parallel Optimization. Theory, Algorithms and Applications*. New York: Oxford Univ. Press, 1997.
[14] F. Natterer, *Numerical Solution of Bilinear Inverse Problems*, Preprint Series. Muenster, Germany: Westfalische-Wilhelms Univ., 1996.
[15] A. Rangarajan, A. Yuille, and E. Mjolsness, "A convergence proof for the softassign quadratic assignment algorithm," in *Advances in Neural Information Processing Systems*, vol. 9. Cambridge, MA: MIT Press, 1997, pp. 620–626.
[16] S. Gold and A. Rangarajan, "Softmax to softassign: Neural network algorithms for combinatorial optimization," *J. Artif. Neural Netw.*, vol. 2, no. 4, pp. 381–399, 1996.
[17] N. Rubanov, "Bipartite graph labeling algorithm for the subcircuit recognition problem," in *Proc. IEEE Int. Conf. Electron., Circuits, Syst.*, 2001, pp. 1285–1289.
[18] D. Luenberger, *Linear and Nonlinear Programming*. Reading, MA: Addison-Wesley, 1984.
[19] L. Devroye, L. Gyoerfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*. New York: Springer-Verlag, 1996.
[20] G. Engeln-Muellges and F. Uhlig, *Numerical Algorithms With C*. New York: Springer-Verlag, 1996.
[21] S. Kelley, *Iterative Methods for Linear and Nonlinear Equations*. Philadelphia, PA: SIAM, 1995.
[22] G. Goodwin and K. Sin, *Adaptive Filtering, Prediction and Control*. Englewood Cliffs, NJ: Prentice-Hall, 1984.
[23] C. Ebeling, "Gemini II: A second generation layout validation tool," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 1988, pp. 322–325.

Nikolay Rubanov was born in Krupki, Belarus. He received the M.S. degree in radiophysics and the Ph.D. degree in computer science, both from Belarussian State University, Minsk, Belarus, in 1990 and 1994, respectively.

From 1994 to 1998, he was an Associate Professor with the Belarusian State University, Minsk. From 1995 to 1996, he was a Visiting Scientist with the Institute of the Numerical Mathematics, University Muenster, Germany. From 2000 to 2003, he worked with Circuit Semantics, Inc., San Jose, CA. From 2003 to 2005, he worked with Magma Design Automation, Santa Clara, CA. In 2005, he joined Cadence Design Systems, San Jose, CA. His research interests include digital signal processing, graph theory and pattern recognition, and their application to computer-aided design of very large-scale integration circuits.

Authorized licensed use limited to: Korea Advanced Inst of Science & Tech - KAIST. Downloaded on January 31,2024 at 05:09:23 UTC from IEEE Xplore. Restrictions apply.