A New Memetic Algorithm with Fitness Approximation for the Defect-Tolerant Logic Mapping in Crossbar-based Nano-architectures

Bo Yuan, Bin Li, *Member, IEEE*, and Thomas Weise, *Member, IEEE*, Xin Yao, *Fellow, IEEE*

***Abstract*—The defect-tolerant logic mapping (DTLM), which has been proven to be an NP-complete combinatorial search problem, is a key step for logic implementation in emerging crossbar-based nano-architectures. However, no practically satisfactory solution has been suggested for the DTLM till now. In this paper, the problem of DTLM is first modeled as a combinatorial optimization problem through introducing Maximum-Bipartite-Matching (MBM). Then, a new Memetic Algorithm with Fitness Approximation (MA/FA) is proposed to solve the optimization problem efficiently. In MA/FA, a new Greedy Re-assignment Local Search operator, capable of utilizing the domain knowledge and information from problem instances, is designed to help the algorithm find optimal logic mapping with consumption of relatively lower computational resources; A Fitness Approximation method is adopted to reduce the time consumption of fitness evaluation dramatically. In addition, a hybrid fitness evaluation strategy that combines the exact and approximated fitness evaluation methods is presented to balance the accuracy and time efficiency of fitness evaluation. The effectiveness and efficiency of proposed methods are testified and evaluated on a large set of benchmark instances of various scales, and the advantage of MA/FA on keeping good balance between effectiveness and efficiency is also observed.**

***Index Terms*—Memetic algorithms, fitness approximation, local search, crossbar-based nanoelectronics, defect-tolerant logic mapping (DTLM), maximum-bipartite-matching (MBM).**

I. Introduction

CONVENTIONAL CMOS techniques are rapidly approaching their realistic limits. Emerging nano-scale devices [1] and the corresponding nano-architecture technologies [2], which can achieve much higher device density (10^{12}/cm^2) and operation frequency (over 100 gigahertz), are expected to extend the Moore law beyond CMOS. In 2011, the world's first programmable nanoprocessor consisting of programmable, non-volatile nanowire transistor arrays (PNNTAs) has been published [3], which demonstrates that the bottom-up paradigm [4] can yield nanoprocessors and other integrated systems of the future.

Although with many attractive features and encouraging potential in future industrial applications, the nano-chips produced from both the bottom-up process and nano-imprint techniques [5] are prone to suffer high defect density due to the extremely small size of nanoelectronic devices and the difficult of controlling the fabricating process precisely. The exact level of defect density is still unknown by now, but it is assumed to be reasonable that 1% to 15% of the resources (wires, switches, etc.) on a nano-chip will be defective [6]. The researchers in Harvard and MITRE [3] characterized the threshold voltage values of nodes from the fabricated PNNTA structure in both active and inactive states. It is notable that they found that only 86% nodes in active state and 87% nodes in inactive state met the voltage requirements. The Quantum Science Research group at Hewlett-Packard fabricated an 8×8 crossbar architecture using molecular switches at the crosspoints by nano-imprint lithography [5], where 15% of the switches were defective.

Faced with such a high defect density, the future nano-chip designing industry definitely needs crafted defect-tolerant design techniques to guarantee the usability of manufactured nano-chips. One promising design paradigm for logic function implementation on a nano-chip is the defect-aware design flow [7]–[11]. The key step in defect-aware design is the defect-tolerant logic mapping (DTLM), which is defined as: given a defective crossbar and a logic function to be implemented on it, find a mapping of the logic function to the crossbar with consideration of defects. Since the defect-aware design flow is to adjust the function implementation for each particular defective nano-chip, it can utilize more defect-free resources on the crossbar architectures compared to the defect-unaware design flow whose key step is to extract defect-free sub-crossbars from original defective crossbars [12], [13].

The DTLM by its nature is equivalent to the Subgraph

Bo Yuan is with the School of Information Science and Technology, University of Science and Technology of China (USTC), Hefei 230026, China (e-mail: yuanbo@mail.ustc.edu.cn).

Xin Yao is with Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), School of Computer Science, University of Birmingham, Edgbaston, Birmingham B15 2TT, U.K. (e-mail: X.Yao@cs.bham.ac.uk).

Bin Li is with the School of Information Science and Technology, USTC, Hefei 230026, China (e-mail: binli@ustc.edu.cn).

Thomas Weise is with the Nature Inspired Computation and Applications Laboratory (NICAL), School of Computer Science and Technology, USTC; Hefei 230027, China (e-mail: tweise@ustc.edu.cn).

Isomorphism Problem (SIP), a well-known NP-complete combinatorial search problem [14], which can be defined as: return an occurrence of bipartite graph G_2 as a subgraph of bipartite graph G_1. A number of methods have been proposed to tackle such problem, e. g. a recursive algorithm [11] based on backtracking and pruning [15], [16], as well as various versions of popular heuristic algorithms specialized for the DTLM [7], [10], [17], [18]. But all of the above methods can only satisfy the requirement of solving SIP of small scale in nano-chip design so far.

In this paper, the problem of DTLM is first modeled as a combinatorial optimization problem through introducing Maximum-Bipartite-Matching (MBM) and the corresponding search space is reduced significantly. Then a new Memetic Algorithm with Fitness Approximation (MA/FA) is proposed to find a valid mapping between logic function and nano-crossbar architecture in reasonable runtime, which bypasses all the defect resources in the nano-architecture. For real-world optimization problems, it is often effective to incorporate problem-specific knowledge into local search strategies, which are referred to "memes" in the case of Memetic Algorithms [19]–[21]. This paper presents one such local search operator, called Greedy Re-assignment, which reassigns the values of parts of the individual by taking advantage of the greedy information extracted from the problem instance. Besides, an approximated MBM algorithm [7] is used together with the exact MBM algorithm [22] to evaluate the fitness of candidate solutions. Experimental investigation shows that such hybrid fitness evaluation strategy is able to reduce the runtime of the whole algorithm dramatically. The proposed algorithm is experimentally investigated on a large set of benchmark instances with various scales, and compared with the state-of-the-art recursive and heuristic algorithms. Experiment results show a good balance between efficiency and effectiveness can be obtained by the proposed MA/FA and the performance of MA/FA attributes to the introduction of the Greedy Re-assignment and Fitness Approximation strategies.

The rest of this paper is organized as follows. Section II introduces the background and definition of the DTLM problem, and briefly reviews the literatures on DTLM. In Section III, the DTLM is transferred into a combinatorial optimization problem via introducing MBM. The algorithm, MA/FA with Greedy Re-assignment and Fitness Approximation strategies, is presented in Section IV. Experimental studies and comparisons are given in Section V. Section VI concludes the paper.

II. Preliminaries

A. Nano-crossbar Architecture

A nanoelectronic crossbar consists of two layers of orthogonal nanowires. The region where two wires cross is called junction or crosspoint, which may be configured to implement a logic device. The assembly process has a stochastic nature that the probability of aligning three-terminal devices will be very low, while a two-terminal connection can be established merely by overlapping two wires perpendicularly.

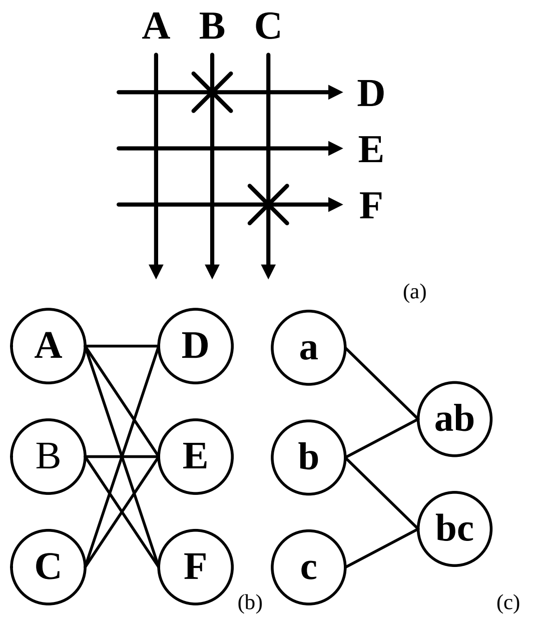

Fig. 1. (a) 3×3 nano-crossbar with two defects. (b) Bipartite graph of crossbar in (a). (c) Bipartite graph of logic function: $F = ab+bc$.

Therefore, two-terminal devices such as nanowire FETs (Field Effect Transistors), diodes, and molecular switches are preferred [6].

In this paper, for the sake of simplicity, only "stuck-at-open" defect is considered, which is representative and the most common in nano-crossbar architectures [23]. A "stuck-at-open" defect means that there is either a non-programmable switch or missing a switch at the crosspoint, thus the two cross wires at this crosspoint are always disconnected. Note that defect modeling for emerging nanoelectronics is still an ongoing research problem. Without loss of generality, we may assume that the defects are independent and uniformly distributed as previous work did [7]–[9], [11]. This is a commonly employed assumption for theoretical research [24], which allows us to focus upon the essence of the proposed method instead of the physical details of the defects. It is notable that the approach presented in this paper can be easily extended to other defect types ("stuck-at-closed" defect, "nanowire open" defect and "nanowire bridging" defect [25]) and other defect distributions (such as clustered distribution [10]) by modifying the following bipartite graph model slightly.

B. Problem Definition

An example of a defective 3×3 nanoelectronic crossbar is shown in Fig. 1(a). The crossbar consists of two sets of orthogonal nanowires. The vertical nanowires are the inputs, whereas the horizontal nanowires are the outputs. There is a programmable switch at each crosspoint. The non-programmable defective switches at the crosspoints are each represented by an "X".

A given 2D crossbar with defects can be represented by a bipartite graph, as shown in Fig. 1(b). A bipartite graph of an $n \times n$ crossbar is an undirected bipartite graph $G_1(U_{\text{vert}}, V_{\text{hor}}, E_1)$ with partitions U_{vert} and V_{hor}, having $|U_{\text{vert}}| = n$ and $|V_{\text{hor}}| = n$. U_{vert}

represents the set of input nanowires, and V_{hor} represents the set of output nanowires. E_1 consists of representative edges for all the programmable non-defective crosspoints in the crossbar.

A two-level logic function in a sum-of-products form can also be represented by a bipartite graph $G_2(U_{\mathrm{var}}, V_{\mathrm{term}}, E_2)$, as shown in Fig. 1(c). In this scenario, U_{var} represents the set of logic variables, and V_{term} represents the set of product terms. E_2 consists of representative edges for the corresponding product terms contain the variables.

When using a crossbar structure to implement a two-level logic function, the logical relationships between the variables and the product terms in the logic function can be represented by the connections between vertical and horizontal nanowires in the crossbar. Such logic-function-to-crossbar mapping problem can be formulated as a Subgraph Isomorphism Problem (SIP) [14]: returning an occurrence of logic function bipartite graph G_2 as a subgraph of crossbar bipartite graph G_1, which is a well-known NP-complete combinatorial search problem [14].

The DTLM problem can be formally defined as the following [11]: Given a defective $m\times m$ crossbar bipartite graph $G_1(U_{\mathrm{vert}}, V_{\mathrm{hor}}, E_1)$ having $|U_{\mathrm{vert}}| = |V_{\mathrm{hor}}| = m$, and an $n\times n$ logic function bipartite graph $G_2(U_{\mathrm{var}}, V_{\mathrm{term}}, E_2)$ having $|U_{\mathrm{var}}| = |V_{\mathrm{term}}| = n$, find a node mapping ($M$: $U_{\mathrm{var}}\rightarrow U_{\mathrm{vert}}$, $V_{\mathrm{term}}\rightarrow V_{\mathrm{hor}}$) such that $\forall(n_1, n_2)\in E_2$, $n_1\in U_{\mathrm{var}}$, $n_2\in V_{\mathrm{term}}$, $(M(n_1), M(n_2))\in E_1$ holds.

C. Literatures Review

In the domain of subgraph isomorphism research, low complexity algorithms have been a subject of research during the last three decades. A certain number of algorithms were proposed to reduce the overall computational complexity of the search process by imposing restrictions on the graphs [26]. An alternative approach is that using an adequate representation of the search process and pruning unprofitable paths in the search space. A successful example that significantly reduces the size of the search space is the recursive algorithm proposed by Ullmann [15]. This algorithm is still one of the most commonly used approaches for SIP. Cordella et al. [16] suggested another recursive algorithm which grows a set of partial subgraphs until the isomorphic subgraph is found. It was testified that the growing process reduces considerably the search space by providing pruning rules and a dynamic ordering method.

Rao et al. [11] proposed a recursive algorithm based on the search tree with backtracking, in which, three enhanced heuristic pruning techniques were presented to improve the efficiency by significantly cutting down unnecessary backtracking processes. While, even with the assistance of heuristic pruning techniques, the runtime of the recursive algorithm [11] is still prohibitive for larger scale graphs duo to the recursive nature of the algorithm. It can obtain good performance only on small scale problem instances (less than or equal to 16 inputs for logic functions).

Heuristic algorithms are gaining more and more interest in recent years. Dehon and Naeimi [7] transformed the DTLM into a Maximum-Bipartite-Matching (MBM) problem. They first assumed the logic variables had been assigned to the input nanowires, then, they constructed a bipartite graph to model which product terms can be assigned to which output nanowires. Next, they abstracted the problem as to find a complete assignment from the product terms to the output nanowires, which is equivalent to the MBM problem. Although an exact MBM algorithm [21] could find the matching or an evolutionary algorithm [27] could produce a matching with near maximum cardinality, a linear-time greedy algorithm [7] was proposed to provide approximated results while running in substantially less runtime.

Yellambalase and Choi [10] evaluated three different heuristic logic mapping algorithms for DTLM with clustered defects: the row-wise matching algorithm, the column-matching-first algorithm and the redundant column-matching-first algorithm. To choose a pin assignment from the variables to the input nanowires, one heuristic [10] was proposed to greedily assign the most frequently used variables in the product terms to the input nanowires with the smallest number of defects.

Simsir et al. [17] introduced a hybrid nanowire-CMOS architecture which contained a compiler with a defect-tolerant logic mapping heuristic. The heuristic mapping algorithm employed the same greedy pin assignment method as [10]. While, instead of constructing the complete bipartite graph as [7], which is time-consuming, they constructed the bipartite matching between product terms and output nanowires step by step assisted by an exact MBM algorithm to check if a valid matching exists.

Inspired by the canonization technique which is normally used in solving the Graph Isomorphism Problem (GIP), Gogen et al. [18] proposed a novel heuristic mapping technique based on the canonization instead of the search tree with backtracking, called KNS-2DS. KNS stood for K-Neighbor-Sort which was used for initializing their main mapping heuristic, 2-Dimensional-Sort (2DS). 2DS operated on the adjacency matrixes corresponding to the crossbar and logic function bipartite graph respectively. Experiment results showed that KNS-2DS could reduce runtime significantly as opposed to the SAT-based technique [28].

By now, all the state-of-the-arts heuristic algorithms rely on the fixed heuristics which show strong bias in favor of only small set of problem instances, and it is hard for these heuristic algorithms to find valid mappings for large scale logic functions with many variables and product terms.

III. Modeling and Evaluation

A. Search Space of the DTLM

Base on the definition of the DTLM, two decision vectors can be employed to represent the mapping trial M: input mapping vector (IMV) and output mapping vector (OMV) [29], where, $IMV[v] = i$ if variable v is assigned to input nanowire i, $1\leq v\leq n$, $1\leq i\leq m$; $OMV[p] = o$ if product term p is mapped to output nanowire o, $1\leq p\leq n$, $1\leq o\leq m$. It seems that we can search the whole solution space spanned by IMV and OMV as previous works [29] and [30] did, but the extremely huge size of search space, $P(m, n)\times P(m, n)$, will make the problem very hard to be

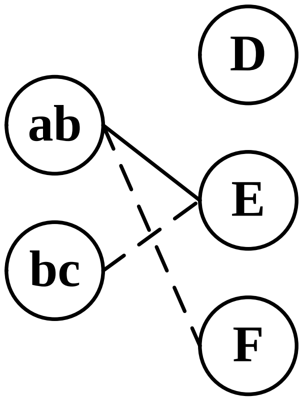

Fig. 2. Maximum-bipartite-matching (dashed lines).

solved with limited computational resource, where $P(m, n)$ is the number of n-permutations of m.

Fortunately, as suggested in [7] and [17], when logic variables are previously assigned to input nanowires (*IMV*), the solution space of another mapping vector (*OMV*) will be restricted severely. For example as shown in Fig.1, if *IMV* is set as [1, 2, 3] which means *a* is assigned to *A*, *b* is assigned to *B*, *c* is assigned to *C*, thus *ab* cannot be assigned to *D*, because there is no edge between *B* and *D* in crossbar bipartite graph. Therefore, we can construct a bipartite graph to model which product terms can be assigned to which output nanowires as shown in Fig. 2. While creating the bipartite graph, we add one node on the "left side" for each product term *p*, and one node on the "right side" for each output nanowire *o*. An edge between *p* and *o* indicates that the product term *p* is compatible with the defect pattern of the crossbar, and can be realized by *o*. Then, the problem is transformed to find a complete assignment from the product terms to the output nanowires which is equivalent to the MBM problem [22]:

Given an undirected bipartite graph $G = (U, V, E)$, where U and V are disjoint and all edges in E go between U and V. A matching is a subset of edges $M \in E$ such that for all vertices $v \in U \cup V$, at most one edge of M is incident on v. We say that a vertex $v \in U \cup V$ is matched by matching M if some edge in M is incident on v; otherwise, v is unmatched. A maximum matching is a matching of maximum cardinality, that is, a matching M such that for any matching M', we have $|M| \geq |M'|$. The set of dashed lines in Fig. 2 is a MBM in the graph.

B. Ford-Fulkerson Method for MBM

Given an undirected bipartite graph $G = (U, V, E)$, one can use the Ford-Fulkerson method [22] to find a MBM in polynomial time in $|U \cup V|$ and $|E|$. The trick is to construct a flow network where flows correspond to matchings. The corresponding flow network $G' = (V', E')$ for G is defined as follows: We let the source s and sink t be new vertices, and $V' = U \cup V \cup \{s, t\}$. The directed edges of G' are the edges of E, directed from U to V, along with $|U \cup V|$ new edges: $E' = \{(s, u): u \in U\} \cup \{(u, v): u \in U, v \in V, \text{ and } (u, v) \in E\} \cup \{(v, t): v \in V\}$. To complete the construction, unit capacity is assigned to each edge in E'. Thus, given an undirected bipartite graph G, one can find a MBM by creating the flow network G', running the Ford-Fulkerson method, and directly obtaining a maximum matching M from the integer-valued maximum flow f found.

Algorithm 1: Ford-Fulkerson Method [22]

```
//Ford-Fulkerson method for maximum flow (MBM)
Input: Flow network $G' = (V', E')$
Output: Maximum flow $f$
1: $f = 0$
2: while there exists an augmenting path $p$ do
3:   augment flow $f$ along $p$
4: end while
5: return $f$
```

The Ford-Fulkerson method is iterative as shown in **Algorithm 1** [22]. The algorithm starts with $f(u, v) = 0$ for all u, $v \in V'$, giving an initial flow of value 0. At each iteration, the flow value is increased by finding an "augmenting path" that can be thought of simply as a path from the source s to the sink t along which more flow can be sent and augmented. This process is repeated until no augmenting path can be found. The max-flow min-cut theorem proves that upon termination, this process yields a maximum flow. The details of the basic Ford-Fulkerson algorithm and its improved versions can be referenced in [22].

C. Optimization Model

Given the *IMV*, the search space of *OMV* can be significantly reduced via creating the corresponding bipartite graph modeling which product terms can be assigned to which output nanowires. Furthermore, it is possible to employ the Ford-Fulkerson method to find a MBM exactly between product terms and output nanowires. If each product term has a corresponding output nanowire in the matching, the given *IMV* is judged global optimum and a valid mapping is found. For example, given *IMV* = [1, 2, 3] for Fig. 1, its corresponding bipartite graph can be created as shown in Fig. 2, in which a MBM *OMV* = [3, 2] (dashed lines in Fig. 2) would be obtained, which means both *ab* and *bc* have corresponding output nanowires in the matching, so the given *IMV* = [1, 2, 3] is a global optimum and the combination of *IMV* = [1, 2, 3] and *OMV* = [3, 2] is a valid mapping trial *M*.

The MBM problem is exactly a matching problem, which can be relaxed to be an optimization problem via allowing the existence of unmatched product terms in the solution. The following objective function can be defined for the given *IMV*:

$$Objective = \sum_{p=1}^{n} m_p \cdot w_p \Big/ \sum_{p=1}^{n} w_p \quad (1)$$

Where, $m_p \in \{0, 1\}$ represents if product term p has a corresponding output nanowire o in the matching under the given *IMV*, while, weight w_p represents the impact of product term p on the fitness value. *Objective* = 1 means each product term has a corresponding output nanowire in the matching, so a valid mapping is found.

So far, the problem of DTLM is modeled as a combinatorial optimization problem, or rather as an assignment problem (AP) [31]–[35]: optimizing the pin assignment from logic variables to input nanowires (*IMV*), while evaluating the *IMV* according to (1) through the exact MBM evaluation (Ford-Fulkerson method).

D. Problem Characteristics

Although the DTLM has been modeled as an AP, it is difficult to apply the widely used effective algorithms for APs directly due to the characteristics of the DTLM itself. For example, the encoding of the solution (*IMV*) in the DTLM is essentially an incomplete permutation, while a complete permutation is necessary in algorithms for QAP (Quadratic Assignment Problem) [31] or a combination of integers for TAP (Terminal Assignment Problem) [32]. Such difference in encoding requires problem-specific operators for the DTLM, such as the mutation operator in MA/FA to be introduced in section IV. In addition, the high time-complexity of quality evaluation of candidate solutions in the DTLM is also a challenge compared with other APs. This issue has to be considered from the point of view of algorithmic efficiency to design operators for the DTLM, such as the local search operator in MA/FA to be introduced in section IV.

IV. MA/FA for the DTLM

In this section, the Memetic Algorithm (MA) specialized for the DTLM is presented. Besides incorporating successful elements of previous effective heuristic mapping algorithm [10], [17], the proposed MA gains pretty good balance between quality of solution and running time in two approaches: 1) incorporating evolutionary computation framework to enhance the global optimization; 2) introducing approximated MBM evaluation method [7] to reduce the running time of the whole algorithm. The idea of introducing local search and fitness approximation into EAs is a proven technique [36], which can help to solve complex problems more effectively and efficiently.

A. Framework of MA/FA

The procedure of proposed MA/FA is depicted in **Algorithm 2**. The Genetic Algorithm (GA) is adopted to work as the evolutionary computation framework of the Memetic Algorithm due to its success history on many assignment problems [31]–[35]. The detailed design of the elementary steps of the algorithm is introduced below.

1) Encoding

The encoding of *IMV* solutions used in our implementation is straightforward. We encode the permutation π (denotes a permutation of the set $M = \{1, 2,\ldots, m\}$) as a vector of input nanowires, such that the value j of the i^{th} component in the vector indicates that input nanowire j is assigned to logic variable i ($\pi(i) = j$).

It is notable that the logic function size n is smaller than the crossbar architecture size m in some cases, so *IMV* is an incomplete permutation. In order to take advantage of the previous crossover operators proposed previously, CX recombination, the complete permutation π is used instead of incomplete permutation. However, only the first n components will be decoded as *IMV* for the MBM-based fitness evaluation.

$G_1(U_{\text{vert}}, V_{\text{hor}}, E_1)$	Bipartite graph of crossbar architecture
$G_2(U_{\text{var}}, V_{\text{term}}, E_2)$	Bipartite graph of logic function
N	Population size
P	Parents
B	Offspring
t	Iteration counter
f	Fitness value
λ	Greedy strength factor
Δ	Exact evaluation gap
P_{cross}	Probability of crossover
P_{mut}	Probability of mutation
P_{ls}	Probability of local search

Algorithm 2: MA/FA

```
//The pseudo-code of MA/FA for the DTLM
1: $P_i$ = random permutation $\pi$, $i$=1, 2,…, $N$
2: $f(P_i)$ = **Exact_MBM_Evaluation**($P_i$), $i$=1, 2,…, $N$
3: $t$ = 0;
4: **Repeat**
5: $t$ = $t$+1
6:   **for** $i$ = 1 **to** $N$ **do**
7:     select two parents $P_j$, $P_k$,from $P$ randomly
8:     $B_i$ = **Crossover**($P_j$, $P_k$, $P_{\text{cross}}$)
9:   **end for**
10: **for** $i$ = 1 **to** $N$ **do**
11:    $B_i$ = **Mutation**($B_i$, $P_{\text{mut}}$)
12:    $B_i$ = **Greedy_Reassignment**($B_i$, $P_{\text{ls}}$, $\lambda$)
13: **end for**
14: **for** $i$ = 1 **to** $N$ **do**
15:    **if** $t\%\Delta == 0$ **then**
16:       $f(B_i)$ = **Exact_MBM_Evaluation**($B_i$)
17:    **else**
18:       $f(B_i)$ = **Approximated_MBM_Evaluation**($B_i$)
19:    **end if**
20: **end for**
21: $P$ = **Selection_for_Survival** ($P$, $B$)
22: **until** runtime reached or a valid mapping founded
```

2) Crossover

The CX recombination operator [31] has been testified to be an effective operator for assignment problems. It preserves the information contained in both parents in the sense that all alleles of the offspring are taken either from the first or from the second parent. The operator does not perform any implicit mutation, since an input nanowire j that is assigned to variable i in the child is also assigned to variable i in one or both parents.

In the first phase, all input nanowires found at the same variable in the two parents are assigned to the corresponding variables in the offspring. Then, starting with a randomly chosen variable with no assignment, a nanowire is randomly chosen from the two parents. After that, additional assignments are made to ensure that no implicit mutation occurs. Then, the next unassigned variable to the right (in case we are at the end of the genome, we proceed at its beginning) is processed in the same way until all variables have been considered.

3) Mutation

Since the logic function sizes *n* may be smaller than or equal to the crossbar architecture sizes *m*, we consider applying a mutation operator in two cases: 1) If $n<m$, we will randomly select a gene to be mutated and exchange its value with another gene from the last n-m gens. 2) If $n=m$, we will randomly select two genes and then exchange their values.

4) Selection

Selection occurs two times in the main loop of MA/FA. *Selection for reproduction* is performed before a crossover operator can be applied, which is based on a purely random basis without bias to filter individuals, and *selection for survival* is performed to reduce the population to its original size, which is achieved by choosing the best individuals from the pool of parents and children [31].

B. Greedy Re-assignment Local Search

It was thought that some universal local search methods, such as the *2-opt* [37] and the *fast-2-opt* [31] heuristics, could be applied to the problem of DTLM. These universal local search methods employ no problem-specific knowledge, and require very frequent quality evaluation of the generated solutions to gain information for guiding search, which means a large number of runs of Ford-Fulkerson algorithm for MBM are needed during the execution of the whole algorithm. While in practice, the execution of Ford-Fulkerson is very time-consuming for a moderate scale of MBM problem.

A variant of the *2-opt* heuristic was tested on the DTLM problem. In order to speed up the local search process, the variant is based on performing the first swapping found that increases the fitness. The experimental results were negative as expected. The MA with the variant of the *2-opt* heuristic cannot find a valid mapping in the given runtime on most benchmark instances.

There is a good knowledge that has been testified to be effective on most instances of DTLM, that is, a more frequently used variable needs more functional crosspoints. By assigning the most frequently used variables in the product terms to the input nanowires with the smallest number of defects, the greedy assignment heuristic might find the feasible solution with a high probability [10] [17]. However, there is no flexibility in such strong greediness, resulting in poor performance even on small scale problems as shown in Section V. C. In addition, direct application of such strategy will make all individuals identical solution (all individuals are the same).

Inspired by the knowledge, a new problem-specific local search operator is designed in MA/FA, which reassigns the input nanowires of parts of the variables via taking advantage of the greedy information extracted from the problem instances. In more detail, given a pin assignment (*IMV*), $n\times\lambda$ variables and their corresponding $n\times\lambda$ input nanowires are randomly selected

Algorithm 3: Greedy-Re-assignment-Local-Search

```
//Greedy re-assignment from logic variables to input nanowires IMV
Input: Pin assignment IMV
Output: New pin assignment IMV
1: randomly select $n\times\lambda$ variables and their corresponding $n\times\lambda$ input nanowires, mark them unvisited
2: while there are unvisited logic variables do
3:   find the unvisited variable $v$ with maximum degree
4:   find the unvisited input nanowire $i$ with maximum degree
5:   $IMV[v] = i$
6:   mark $v$ and $i$ as visited
7: end while
8: return IMV
```

and remarked as unvisited, where *n* is the number of variables and $0\leq\lambda\leq1$ is named greedy strength factor here. Then, the greedy assignment heuristic is applied on these selected variables and nanowires to get a new solution (*IMV*). The new local search process, which incorporates problem-specific knowledge with stochastic evolutionary search, is named Greedy Re-assignment Local Search (**Algorithm 3**).

The greedy strength factor λ provides a flexible control on the randomness or greediness of the local search operator. The randomness/greediness of the operator will decrease/increase along with the increasing of λ. When $\lambda=1$, the whole *IMV* will be re-assigned according to the greedy assignment heuristic [10], [17], which will result in identical solution for the current DTLM instance, therefore the operator will reach its minimum randomness and maximum greediness.

Besides randomly selecting $n\times\lambda$ variables, a greedy selection strategy that selects the $n\times\lambda$ most frequently used variables was also studied. The experimental analysis showed that, with such greedy selection strategy, the diversity of the population degraded quickly, and the runtime of the algorithm for getting a valid mapping was extended dramatically in most cases, furthermore, the algorithm could not even find a valid mapping in the given runtime on some benchmark instances.

C. MBM-oriented Fitness Evaluation

The fitness function can be defined directly from the objective function (1):

$$Fitness = \sum_{p=1}^{n} m_p \cdot w_p \Big/ \sum_{p=1}^{n} w_p \quad (2)$$

Where, m_p and w_p have been defined in Section III. C. *Fitness* = 1 means each product term has a corresponding output nanowire in the matching, so a valid mapping is found.

An appropriate weight setting can guide the optimization algorithms to converge to the global optima (“1” for our fitness function) faster. As suggested in [30], the value of weight w_p is related to the number of variables v_p in product term *p*, that is, it is harder to map a produce term *p* whose v_p is larger. Therefore,

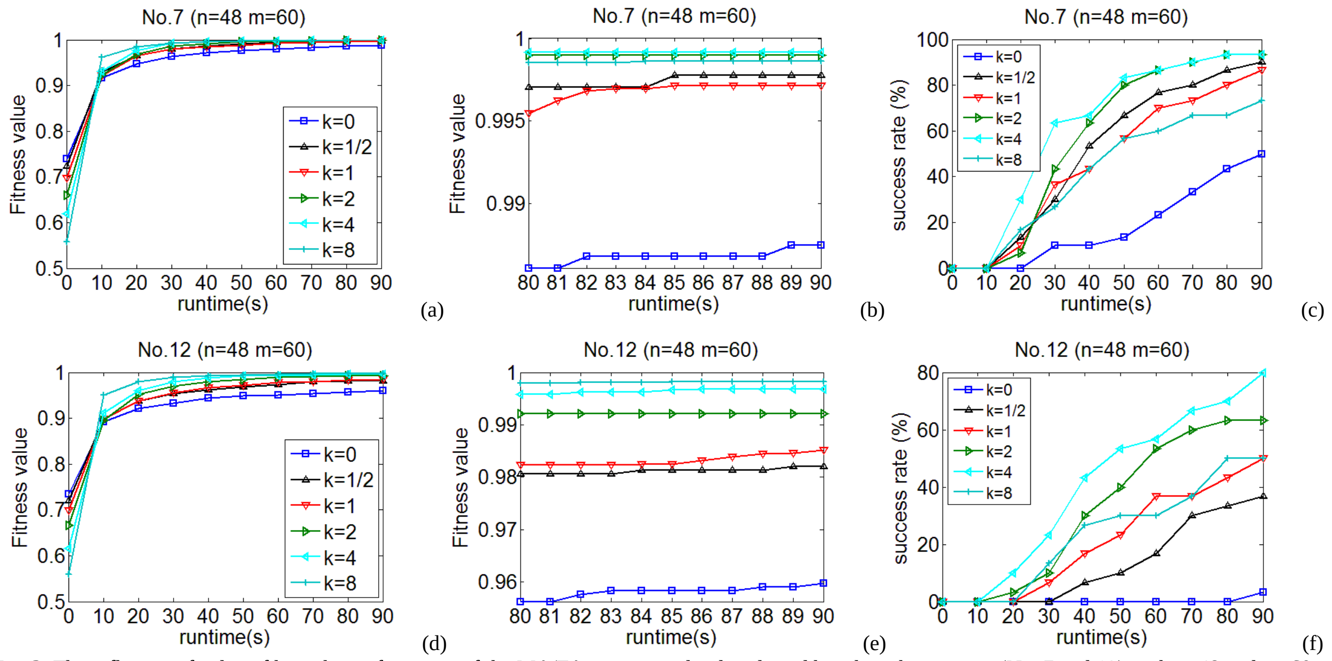

Fig. 3. The influence of value of k on the performance of the MA/FA on two randomly selected benchmark instances (No. 7 and 12) with n=48 and m=60.

w_p are expressed as v_p^k and k is set experimentally. Fig. 3(a) to (f) show the influence of the value of k on the performance of MA/FA on two randomly selected benchmark instances with n = 48 and m = 60. They are statistic mean values from 30 independent runs for each instance. Fig. 3(a) and (d) show the evolutionary curves of fitness value with runtime (limited to 90s). In order to see the differences clearly, they are enlarged for the last 10s as shown in Fig. 3(b) and (e). Fig. 3(c) and (f) show the evolutionary curves of success rate (the probability of a valid mapping is found) with runtime. Although the figures show that larger k can accelerate MA/FA converge to a higher fitness value, what we really care about is the success rate rather than the fitness value, so k is set as 4 for MA/FA as shown in Fig. 3(c) and (f). Similar experiments results are obtained for other EAs and other benchmark instances in this paper, and k = 4 is among the best values that can be chosen, so k is fixed at 4 for all the EAs in the following experiments.

D. MBM-oriented Fitness Approximation

The problem of DTLM is an emerging application that we are facing highly integrated nanoelectronic architectures. It is possible that more than millions of crossbar-based arrays in a nano-chip need to be configured in the future. Therefore, the runtime is one of the critical factors for this application from a practical perspective. In this section, an approximated fitness evaluation strategy is introduced to reduce the runtime caused by the high time-complexity of the exact MBM evaluation. This is a common technique in surrogate-assisted evolutionary algorithms.

Surrogate models [38] (e. g. polynomial models, neural networks, support vector machines) are often used for computationally expensive optimization or high-dimensional optimization, and the models need to be pre-trained offline or learned/updated online resulting in extra time consumption. Therefore, these conventional models cannot be applied to this application. Instead of approximation models, we introduce an existing linear-time greedy algorithm [7] (**Algorithm 4**) to provide approximated MBM evaluation within substantially less runtime. The approximation method does not need to be pre-trained or learned online, thus it is well suited for the DTLM from a practical point of view.

Algorithm 4: Approximated-Matching [7]

```
//Approximated algorithm for matching product terms to output nanowires OMV
Input: Bipartite graph G = (U, V, E)
Output: Pin assignment OMV
1: do {
2:   p = unmapped product term in U with largest fan-in in G
3:   do {
4:     o = nanowire randomly selected from unused output nanowires in V
5:     if p can be mapped to nanowire o then
6:       OMV[p] = o
7:       mark p as mapped
8:       mark o as used
9:     end if
10:  } while (p unmapped)
11: } while (there are unmapped p in U)
12: return OMV
```

Given an undirected bipartite graph G = (U, V, E) to represent the relationships between the product terms and the output nanowires. Let U be the set of product terms, and V the set of

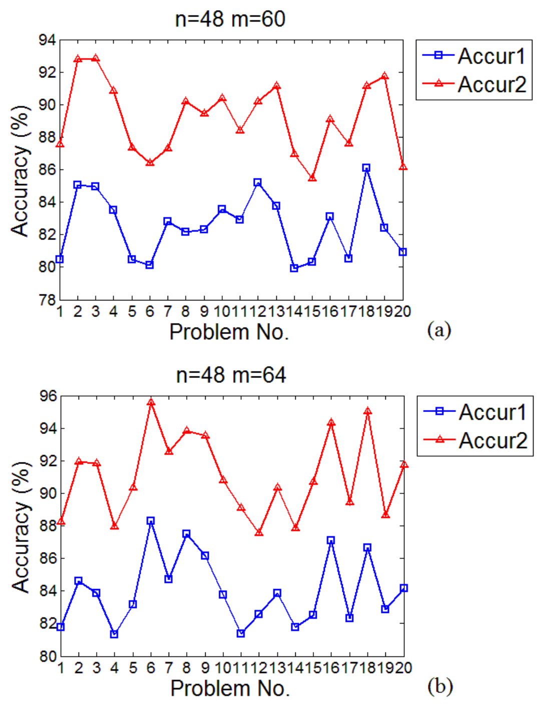

Fig. 4. Accuracy of Approximated-Matching on benchmark instances a) n=48 and m=60 and b) n=48 and m=64.

output nanowires. Let p represent a product term in U, and o an output nanowire in V. The approximated MBM algorithm (**Algorithm 4**) picks the p terms in decreasing order of their fan-in size in G (because larger fan-in product terms are harder to map), and chooses the o terms randomly. When the number of functional junctions per nanowire is bound to a constant, the number of wires tested in line 5 for each product term p is a constant. Consequently, this algorithm runs in linear time, $O(|U|)$.

In MA/FA, both the exact and approximated algorithms for MBM are used to evaluate the fitness of candidate solutions in order to balance the accuracy and speed. Specifically, we use single exact MBM evaluation for the population in every $\varDelta$ iterations, where $\varDelta$ is called exact evaluation gap here. The influence of the value of $\varDelta$ will be tested in Section V as suggested in [39].

E. Accuracy of the Approximated-Matching

A reasonable evaluation method is proposed for the accuracy of Approximated-Matching. Give a bipartite graph $G(U, V, E)$, the measure $Accur_1$ can be defined as follows:

$$|M_1 \cap M_2|/|M_1|\times 100\% \qquad (3)$$

While the measure $Accur_2$ is can be defined as:

$$|M_1 \cap M_2|/|M_2|\times 100\% \qquad (4)$$

Where M_1 (the real MBM) and M_2 are the matchings found by the Ford-Fulkerson Method and Approximated-Matching respectively. $Accur_1$ and $Accur_2$ represent how the intersection $M_1 \cap M_2$ overlaps with M_1 and M_2 respectively. Since a strict theoretical analysis of the accuracy is very hard and out of the scope of the paper, we test the accuracy of Approximated-Matching on n = 48 benchmark instances, where *IMVs* are randomly generated one hundred times to obtain statistic mean values. Fig. 4(a) and (b) show the results of $Accur_1$ and $Accur_2$. High $Accur_1$ (above 80%) means that the most of matching M_1 can be obtained by Approximated-Matching, while high $Accur_2$ (above 85%) means that the most of matching M_2 are existed in the real MBM or M_2 is a approximated subset of M_1. Therefore, Approximated-Matching can prove good approximations to the real MBMs.

MA/FA can avoid the false optima in the following three aspects: 1) The good accuracy of Approximated-Matching can relax the problem of false optima, 2) MA/FA uses the idea from *generation-based evolution control* [39] which can guarantee the correct convergence when the approximate fitness model has false optima, 3) The problem of DTLM is a combinatorial search problem in nature, thus what we are really concerned with are the global optima (when *fitness* = 1) which are always true. If the approximated evaluation gets a maximum fitness value ‘1’, a valid mapping is searched.

V. Experimental Studies

A. Benchmark Instances

As far as we know, no benchmark instances have been specialized for the DTLM, so we randomly generate a large set of benchmark graphs for logic functions and crossbar architectures as previous work did [10], [29], [30]. All the benchmark graphs used in the simulation in this paper and MA/FA source codes with supporting documents are available at: http://home.ustc.edu.cn/~yuanbo/ MAFAforDTLM.rar.

For benchmark graphs of crossbar architectures, we set different sizes: m = 16, 24, 60, 64 and defect density ($1-|E_1|/m^2$): p = 15% (the worst case value [6]) and uniform defect distribution (which is the most common assumption in nanoelectronics [7]–[9], [11]). For ease of recording and comparison, we give such a name rule, C_nA$_1$_pB$_1$_C$_1$, where A$_1$ is the size of the graph, B$_1$ is the defect density and C$_1$ is the sequence order in the benchmark set with the same attributes (both m and p). For example, graph C_m60_p15%_5 means that the graph is of size 60, defect density 15% and it is the 15th graph in the benchmark set with the same attributes m = 60 and p = 15%.

For benchmark graphs of logic functions, we set different sizes: n = 16, 24, 48 and average logic density ($|E_2|/n^2$): p = 40% (a typical value [29]) and uniform edge distribution. For ease of recording and comparison, we give such a naming rule, F_nA$_2$_pB$_2$_C$_2$, where A$_2$ is the size of the graph, B$_2$ is the logic density and C$_2$ is the sequence order in the benchmark set with the same attributes (both n and p). For example, graph F_n48_p40%_3 means that the graph is of size 48, logic density 40% and it is the third graph in the benchmark set with the same attributes n = 48 and p = 40%.

B. Parameters Setting

The population size N is set N = 40 according to the problem scale, since the computational complexity of the *fitness* does not

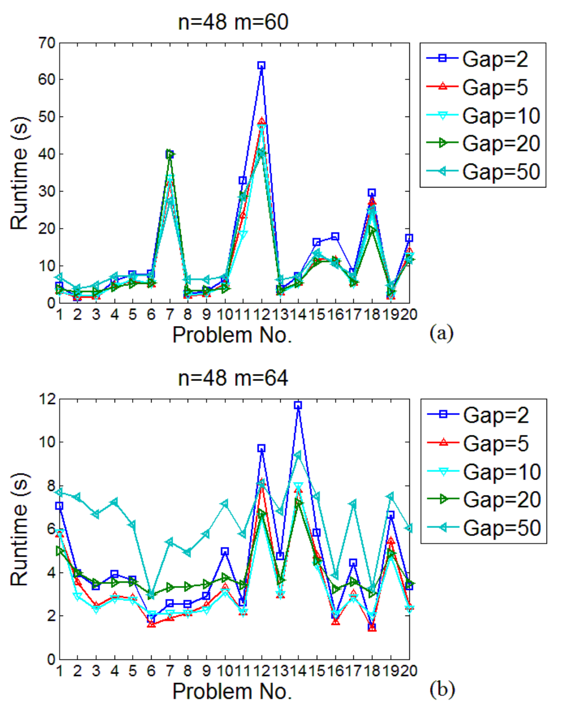

Fig. 5. The influence of exact evaluation gap *Δ* in MA/FA on benchmark instances a) *n*=48 and *m*=60 and b) *n*=48 and *m*=64.

allow evolving much larger populations in reasonable time. A large greedy strength factor λ will weaken the stochastic nature of evolutionary algorithm, thus we set $\lambda = 0.1$ empirically. We set optimal parameters $P_{cross} = 0.8$, $P_{mut} = 0.2$ and $P_{ls} = 0.8$ experimentally by cross validation.

In addition, MA/FA uses the idea from *generation-based evolution control* [39]. In order to test the influence of exact evaluation gap Δ on the performance of MA/FA, we record the average runtime (in seconds) of the algorithms if they find a valid mapping on $n = 48$ benchmark instances. Thirty independent runs are executed to obtain statistic mean values. A few different values of exact evaluation gap Δ are tested as shown in Fig. 5(a) and (b). The experimental results show that MA/FA have better performance on $\Delta = 5$ and 10, and the sensitivity is low between $\Delta = 5$ and 10. Although 100% mapping success rate can be obtained in most cases, higher success rates can be obtained by setting $\Delta = 10$ on some hard problems (such as No. 7, 12 and 18 in Fig. 5(a)), thus Δ is fixed at 10 in the paper.

All the experiments in this paper are performed on 2.66GHz Intel Core 2 Quad processors Q6700 platform with 6G memory. However, all tested algorithms are implemented as monolithic processes and no CPU core parallelism is exploited.

C. *Comparisons with the State-of-the-art Algorithms*

The heuristic mapping algorithm (HMA) [17] and recursive mapping algorithm (RMA) [11] are two representative algorithms for the DTLM whose performances have been testified successfully. Therefore, they are used for comparison in this paper. As stated above, they can only deal with small scale problems, so we set two experiments in the comparisons, $n = m = 16$ and $n = m = 24$.

The runtime of the three algorithms (HMA, RMA, and MA/FA) is limited to 60s and 90s for $n = m = 16$ and $n = m = 24$ respectively. All the algorithms are run independently for thirty times on each benchmark instance. For different problem scales, we randomly select twenty benchmark instances (mapping logic graphs to crossbar graphs) for comprehensive comparison. It is notable that the heuristic mapping algorithm is a deterministic algorithm, so the same result will be obtained after being run multiple times. Therefore, the success rate will be either 0% or

TABLE I
EXPERIMENTAL RESULTS OF HMA [17], RMA [11] AND MA/FA ON *n*=16, *m*=16 BENCHMARK INSTANCES

No.	HMA [17]		RMA [11]		MA/FA	
	Psucc	*Avg*	*Psucc*	*Avg*	*Psucc*	*Avg*
1	0%	NA	10%	7.328	100%	0.013
2	100%	0.001	70%	8.36	100%	0.002
3	100%	0.002	60%	13.608	100%	0.002
4	100%	0.015	37%	6.344	100%	0.004
5	100%	0.001	33%	11.222	100%	0.01
6	100%	0.004	27%	12.348	100%	0.003
7	100%	0.001	20%	15.732	100%	0.003
8	100%	0.006	13%	31.433	100%	0.031
9	0%	NA	17%	21.54	100%	0.046
10	0%	NA	53%	12.73	100%	0.003
11	100%	0.003	10%	3.944	100%	0.025
12	0%	0.001	37%	17.298	100%	0.007
13	100%	NA	37%	25.357	100%	0.004
14	100%	0.024	17%	21.445	100%	0.007
15	100%	0.009	47%	11.287	100%	0.002
16	100%	0.001	53%	8.586	100%	0.003
17	0%	NA	20%	17.638	100%	0.01
18	0%	NA	73%	13.089	100%	0.003
19	100%	0.001	73%	4.504	100%	0.002
20	100%	0.001	73%	10.462	100%	0.002

TABLE II
EXPERIMENTAL RESULTS OF HMA [17], RMA [11] AND MA/FA ON *n*=24, *m*=24 BENCHMARK INSTANCES

No.	HMA [17]		RMA [11]		MA/FA	
	Psucc	*Avg*	*Psucc*	*Avg*	*Psucc*	*Avg*
1	0%	NA	0%	NA	100%	1.487
2	100%	0.034	3%	10.574	100%	0.018
3	0%	NA	0%	NA	100%	0.266
4	0%	NA	0%	NA	100%	0.238
5	0%	NA	0%	NA	100%	0.904
6	100%	0.033	0%	NA	100%	0.217
7	0%	NA	0%	NA	100%	2.357
8	0%	NA	0%	NA	100%	1.277
9	0%	NA	0%	NA	100%	2.268
10	0%	NA	0%	NA	100%	0.588
11	0%	NA	0%	NA	100%	0.74
12	0%	NA	0%	NA	100%	0.179
13	0%	NA	0%	NA	100%	0.914
14	0%	NA	0%	NA	100%	0.211
15	0%	NA	3%	38.213	100%	0.087
16	0%	NA	0%	NA	100%	0.268
17	0%	NA	0%	NA	100%	0.173
18	0%	NA	0%	NA	100%	0.611
19	0%	NA	0%	NA	100%	4.199
20	0%	NA	0%	NA	100%	5.776

TABLE III
EXPERIMENTAL RESULTS OF HMA [17], GA, MA AND MA/FA ON n=48, m=60 BENCHMARK INSTANCES

No.	HMA		GA			MA			MA/FA		
	Psucc	*Avg*	*Psucc*	*Avg*	*Std*	*Psucc*	*Avg*	*Std*	*Psucc*	*Avg*	*Std*
1	0%	NA	100%	15.36	4.25	100%	6.06	1.55	100%	**2.95**	0.89
2	100%	0.12	100%	5.65	2.19	100%	1.94	0.69	100%	1.89	0.25
3	0%	NA	100%	6.37	2.66	100%	2.36	0.4	100%	**1.98**	0.08
4	0%	NA	100%	19.97	5.32	100%	8.79	2.33	100%	**4.6**	1.98
5	0%	NA	100%	22.96	6.07	100%	10.51	2.66	100%	**5.54**	1.29
6	0%	NA	100%	27.66	9.97	100%	10.19	2.31	100%	**4.97**	1.63
7	0%	NA	67%	58.33	16.55	93%	50.09	14.16	90%	30.04	16.51
8	0%	NA	100%	8.39	2.08	100%	3.22	1.15	100%	**2.2**	0.48
9	0%	NA	100%	13.42	4.02	100%	4.04	1.1	100%	**2.27**	0.5
10	0%	NA	100%	18.72	4.58	100%	8.11	2.6	100%	**4.45**	1.53
11	0%	NA	77%	57.61	13.06	100%	33.69	11.95	100%	**22.7**	12.38
12	0%	NA	30%	64.98	9.39	57%	61.55	13.24	70%	**46.09**	18.48
13	0%	NA	100%	11.32	2.61	100%	4.65	1.55	100%	**2.55**	0.75
14	0%	NA	100%	27.18	7.38	100%	9.7	3.49	100%	**4.46**	1.03
15	0%	NA	97%	43.22	15.03	100%	19.76	6.59	100%	**13.3**	6.71
16	0%	NA	97%	40.29	11.63	100%	24.13	13.19	100%	**11.62**	5.47
17	0%	NA	100%	28.05	8.43	100%	10.02	1.81	100%	**5.66**	2.19
18	0%	NA	93%	54.72	15.56	100%	42.46	11.67	100%	**24.5**	13.41
19	100%	0.16	100%	8.2	2.29	100%	2.44	0.91	100%	**1.96**	0.05
20	0%	NA	90%	48.3	16.46	100%	22.61	5.56	100%	**10.63**	4.44

TABLE IV
EXPERIMENTAL RESULTS OF HMA [17], GA, MA AND MA/FA ON n=48, m=64 BENCHMARK INSTANCES

No.	HMA		GA			MA			MA/FA		
	Psucc	*Avg*	*Psucc*	*Avg*	*Std*	*Psucc*	*Avg*	*Std*	*Psucc*	*Avg*	*Std*
1	0%	NA	100%	22.88	5.17	100%	8.61	2.41	100%	**5.52**	1.64
2	0%	NA	100%	13.3	3.96	100%	6.02	2.69	100%	**3.29**	1.15
3	0%	NA	100%	12.51	3.21	100%	4.73	1.33	100%	**2.64**	0.82
4	0%	NA	100%	14.27	3.61	100%	4.9	1.39	100%	**2.73**	0.81
5	0%	NA	100%	12.5	3.91	100%	5.23	1.81	100%	**2.64**	0.76
6	100%	0.05	100%	5.45	1.9	100%	2.37	0.73	100%	**2.03**	0.24
7	0%	NA	100%	10.13	2.55	100%	3.18	0.76	100%	**2.1**	0.06
8	100%	0.15	100%	10.65	2.91	100%	3.4	1.12	100%	**2.14**	0.13
9	0%	NA	100%	11.17	3.03	100%	4.1	0.85	100%	**2.25**	0.38
10	100%	0.16	100%	15.84	4.18	100%	6.54	1.6	100%	**3.3**	1.14
11	0%	NA	100%	11.61	3.87	100%	3.65	1.14	100%	**2.21**	0.19
12	0%	NA	100%	34.22	8.51	100%	15.25	2.95	100%	**7.12**	1.88
13	0%	NA	100%	16.42	3.62	100%	6.15	1.89	100%	**2.97**	0.97
14	0%	NA	100%	33.17	11.84	100%	14.78	3.29	100%	**8.47**	2.26
15	0%	NA	100%	17.08	5.63	100%	8.38	2.28	100%	**4.46**	1.48
16	0%	NA	100%	5.88	2.54	100%	2.45	0.79	100%	**2.12**	0.09
17	0%	NA	100%	18.96	5.41	100%	6.03	1.87	100%	**3.16**	0.93
18	0%	NA	100%	4.48	2.13	100%	1.69	0.8	100%	1.92	0.49
19	0%	NA	100%	26.98	6.21	100%	10.35	3.36	100%	**4.25**	1.29
20	0%	NA	100%	11.8	3.9	100%	4.03	1.51	100%	**2.38**	0.41

100%. Table I and II record the experimental results of different algorithms including:

- *Psucc*: the success rate of the algorithms, i.e., the fraction of the thirty runs that found a valid mapping.
- *Avg*: the average runtime (in seconds) of the algorithms if they find a valid mapping in thirty runs.

Table I shows the experimental results of HMA, RMA and MA/FA on the benchmark instances of n = 16, m = 16. It can been seen that: 1) HMA has success rate of 100% on more than half of the test instances (14 out of 20), while has success rate of 0% on other 6 test instances. The runtime of HMA is very short, this is due to that HMA uses the greedy pin assignment heuristic and the incomplete bipartite graph construction strategy. 2) RMA can find valid mappings on all instances, but it is very time-consuming compared with the other algorithms, this is due to the nature of recursion it adopts. Besides, RMA has low success rate (<50%) on most test instances (13 out of 20), although it was granted a long preset runtime (60s). 3) MA/FA can achieve success rate of 100% on all test instances with very short runtime, even on the instances that are hard for HMA and RMA (such as No. 1, 9, and 17).

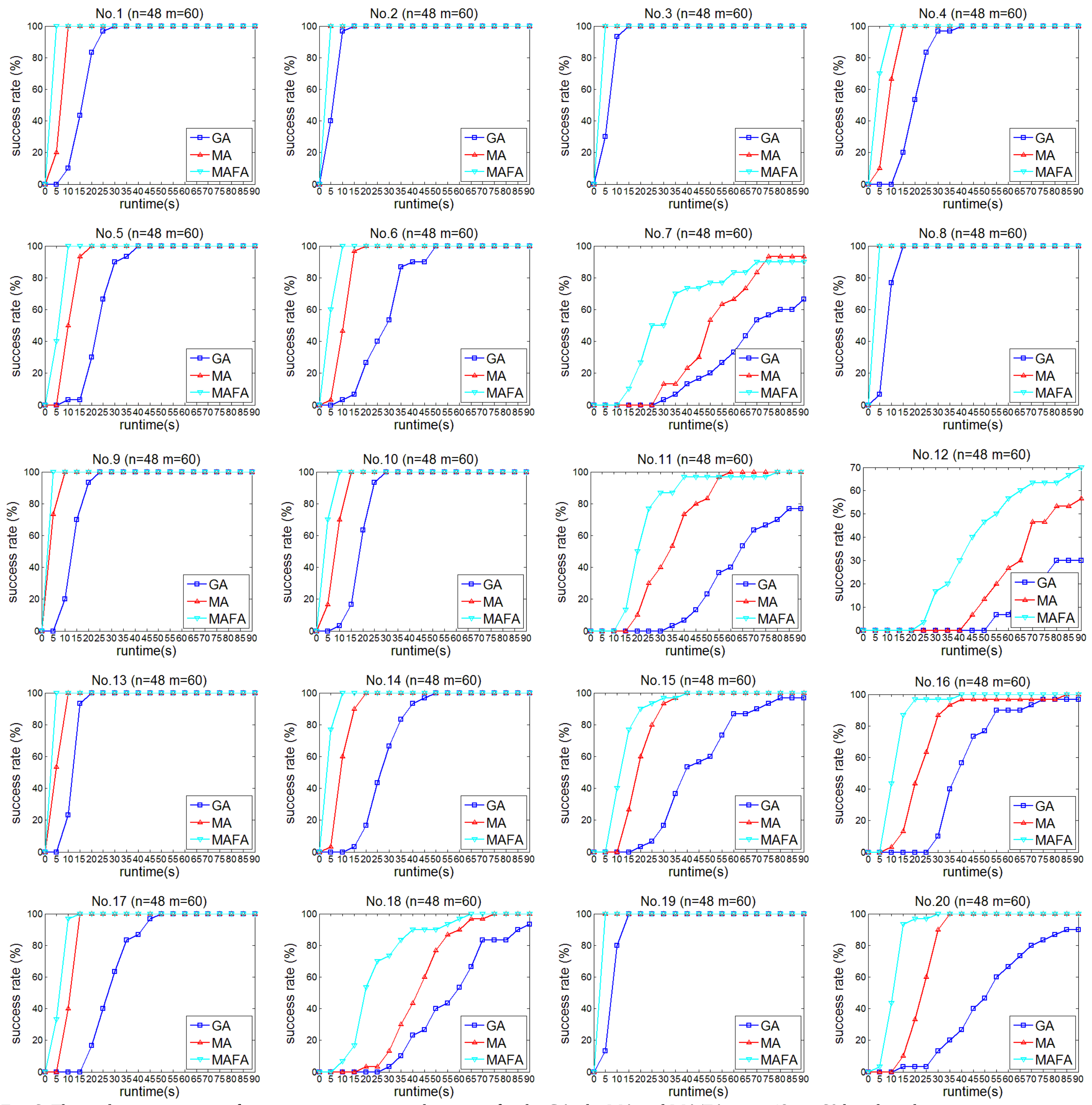

Fig. 6. The evolutionary curves of mapping success rate with runtime for the GA, the MA and MA/FA on n=48, m=60 benchmark instances

Table II shows the experimental results of HMA, RMA, and MA/FA on the benchmark instances of n = 24, m = 24. It can be seen that: 1) HMA works on only two test instances (No. 2 and 6) and the runtime is short (0.05s). 2) Granted a long runtime (90s), RMA can solve only two test instances (No. 2 and 15) with quite low success rate (3% in both cases) and long runtime (10.574s and 38.213s). 3) MA/FA can achieve success rate of 100% on all test instances with very short runtime (0.018s–5.776s).

The above experiment results reveal that the recursive mapping algorithm [11] is very time-consuming with low success rates even on n = m = 16 problems, while the heuristic mapping algorithm [17] has a fairly poor performance on benchmark instances of n=m=24 although it is very fast. MA/FA can solve all benchmark instances efficiently and effectively.

D. Effectiveness of the Greedy Assignment Local Search and Fitness Approximation

Because the runtime of the recursive algorithm is prohibitively long, we only test the heuristic mapping algorithm for large scale problems. In addition, two other EAs are added to the comparisons. One is a MA following the flow of MA/FA without the Fitness Approximation operation, that is, Δ =1 in MA/FA. Another EA is the GA following the flow of the MA without the Greedy Re-assignment Local Search. With the

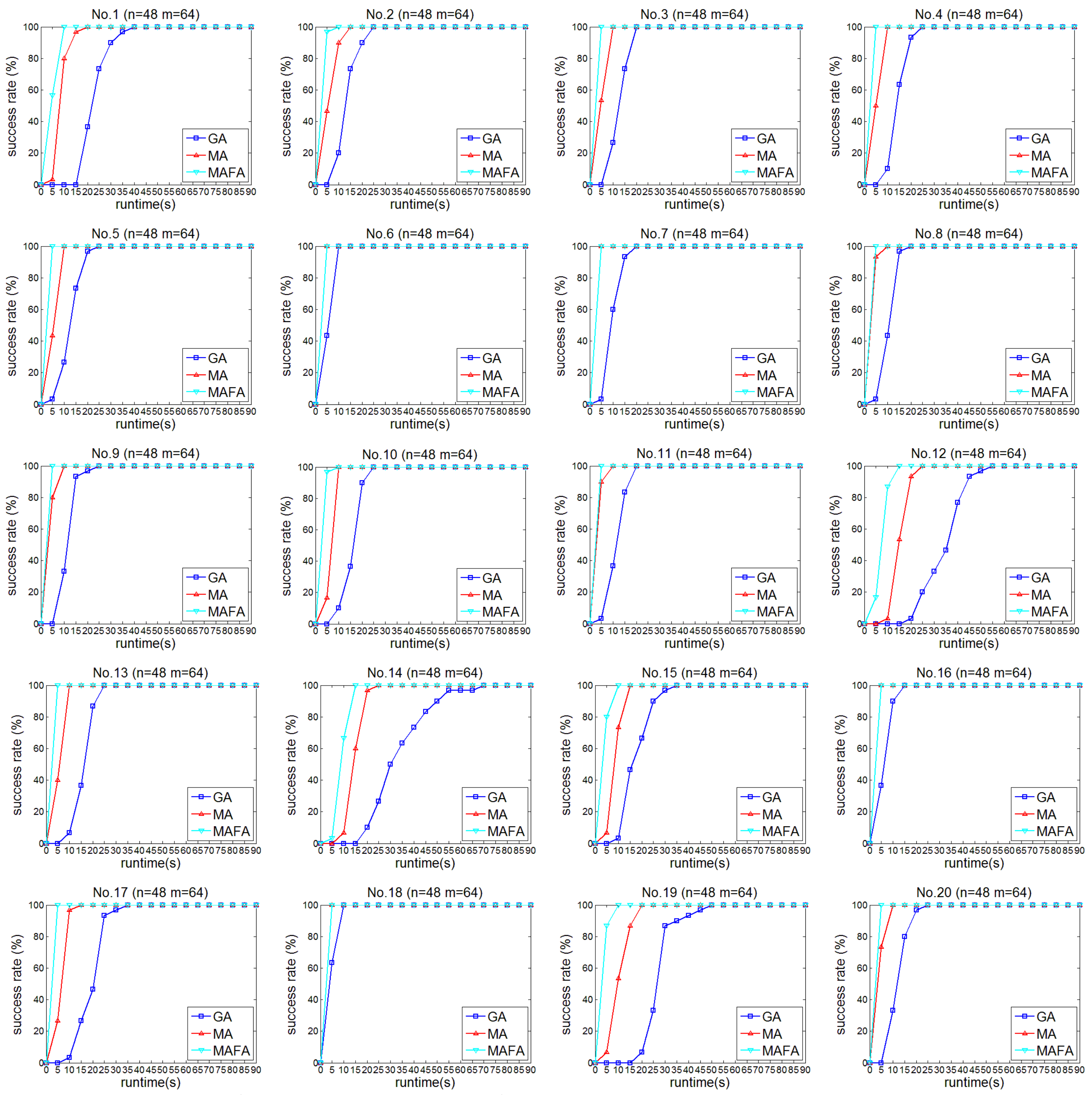

Fig. 7. The evolutionary curves of mapping success rate with runtime for the GA, the MA and MA/FA on *n*=48, *m*=64 benchmark instances

comparison with these two algorithms, the effectiveness of Greedy Re-assignment Local Search and Fitness Approximation can be investigated.

The parameters of MA/FA are set as in the previous experiments. The parameters of the MA are set as the same as MA/FA, except for value of $\Delta = 1$. The parameters N and P_{cross} of the GA are set as the same as other EAs, and we set $P_{mut} = 0.8$ for the GA experimentally by cross validation.

Because MA/FA uses a hybrid fitness evaluation strategy, so we give a pre-determined maximum runtime for the algorithms. The runtime of the three EAs is limited to 90s and all the algorithms are run independently for thirty times on each benchmark instances. For different problem sizes, we randomly select twenty benchmark instances (mapping logic graphs to crossbar graphs) for comprehensive comparison. Table III and IV show the experimental results of different algorithms (HMA, the GA, the MA and MA/FA) including:

- *Psucc*: the success rate of the algorithms, i.e., the fraction of the thirty runs where they find a valid mapping.
- *Avg*: the average runtime (in seconds) of the algorithms if they find a valid mapping in thirty runs.
- *Std*: the standard deviation of the runtime (in seconds) of the algorithms if they find a valid mapping in thirty runs.

We perform statistical tests for the runtimes of paired EAs, GA vs. MA, GA vs. MA/FA and MA vs. MA/FA, on each single benchmark instance. In particular, a two-tailed *t*-test is conducted with a null hypothesis stating that there is no difference between two algorithms in comparison. The null hypothesis is rejected if the *p*-value is smaller than the significance level $\alpha = 0.05$. The runtime of the algorithm which is statistically shorter than both other EAs will be highlighted in bold.

Table III shows the experimental results of HMA, the GA, the MA and MA/FA on the benchmark instances of n = 48, m = 60. It can be seen that: 1) HMA only works on two test instances (No. 2 and 19) with short runtime (0.12s and 0.16s). 2) The GA has success rate of 100% on 13 test instances, and low success rate (<50%) on No. 12. 3) Compared to the GA, the MA improves the success rate significantly on test instances No. 7, 11, and 12. Besides, the runtime is reduced to approximately 1/2 to 1/3 on most instances (No. 1–6, 8–10, 13–17, 19, 20). The comparison (the GA vs. the MA) demonstrates the advantages of introducing the Greedy Re-assignment Local Search. 4) Compared to the MA, the runtime of MA/FA is reduced further on most test instances (18 out of 20) as highlighted in bold. Also, MA/FA maintains as higher success rate as the MA, except on test instances No. 7, where the success rate is reduced slightly (3%). The comparison (the MA vs. MA/FA) demonstrates the advantages of introducing the Fitness Approximation for MBM evaluation.

Table IV shows the experimental results of HMA, the GA, the MA and MA/FA on the benchmark instances of n = 48, m = 60. It can be seen that: 1) HMA works only on three test instances (No. 6, 8, and 10) with short runtime (0.05s, 0.15s, and 0.16s), while the EAs have success rate of 100% on all test instances. 2) Compared to the GA, the runtime of the MA is reduced to approximately 1/2 to 1/3 on all test instances. 3) Compared to the MA, the runtime of MA/FA is reduced further on almost all test instances (19 out of 20) as highlighted in bold. The comparisons (the GA vs. the MA, the MA vs. MA/FA) demonstrate the advantages of introducing the Greedy Re-assignment Local Search and the Fitness Approximation Strategy.

Given fixed n and p, the density of valid solution increases with the crossbar size m as suggested in [40], therefore the success rate of mapping is improved and the runtime is reduced as shown in Table III and IV. In addition, the accuracy of the approximated MBM evaluation also increases along with the crossbar size m as shown in Fig. 4, because a matching of approximated maximum cardinality is easier to find in a denser graph, therefore the performance of MA/FA can be further improved.

Fig. 6 and 7 show the evolutionary curves of mapping success rate with runtime for the GA, the MA and MA/FA on the benchmark instances of n = 48, m = 60 and 64. Good performance of the MAs, especially MA/FA, can be observed obviously.

VI. Conclusion

In this paper, a new framework to solve the DTLM via modeling the problem as a combinatorial optimization problem is presented. A new Memetic Algorithm is proposed to implement the framework, in which a Greedy Re-assignment Local Search operator is designed to make good use of the domain knowledge and the information extracted from the problem instances, a Fitness Approximation method is adopted to reduce the time consumption in fitness evaluation operation. In particular, a hybrid fitness evaluation strategy is presented, which incorporates approximated fitness evaluation with the exact fitness evaluation to get a proper balance between accuracy and time efficiency of fitness evaluation. The performance of proposed methods are testified and evaluated on a large set of benchmark instances of various scales. Experiment results show that the Greedy Re-assignment Local Search can help algorithm to find optimal solution with consumption of lower computational resources, while the hybrid fitness evaluation strategy with Fitness Approximation can reduce the time consumption for fitness evaluation dramatically. It is also obviously observed that the proposed MA/FA algorithm has the advantage on getting good balance between effectiveness and efficiency on various DLTM problem instances.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 61071024**,** 61175065, 61110312, and 61329302), the European Union 7th Framework Program (No 247619), the National Natural Science Foundation of Anhui Province (No. 1108085J16), the Chinese Academy of Sciences (CAS) Fellowship for Young International Scientists (No. 2011Y1GB01) and the Special Financial Grant (No. 201104329) from the China Postdoctoral Science Foundation. The work of Xin Yao was also supported by the Royal Society Wolfson Research Merit Award.

References

[1] G. Bourianoff, J. E. Brewer, R. Cavin, J. A. Hutchby, and V. Zhirnov, “Boolean logic and alternative information-processing devices,” *Computer,* vol. 41, no. 5, pp. 38-46, May 2008.

[2] R. Cavin, J. A. Hutchby, V. Zhirnov, J. E. Brewer, and G. Bourianoff, “Emerging research architectures”, *Computer,* vol. 41, no. 5, pp. 33-37, May 2008.

[3] H. Yan, H. S. Choe, SW. Nam, Y. Hu, S. Das, J. F. Klemic, J. C. Ellenbogen, and C. M. Lieber, “Programmable nanowire circuits for nanoprocessors,” *Nature*, vol. 470, pp. 240-244, Feb. 2011.

[4] W. Lu and C. M. Lieber, “Nanoelectronics from the bottom up,” *Nat. Mater.*, vol. 6, pp. 841-850, 2007.

[5] Y. Chen, G. -Y. Jung, D. A. A. Ohlberg, X. M. Li, D. R. Stewart, J. O. Jeppesen, K. A. Nielsen, J. F. Stoddart, and R. S. Williams, “Nanoscale molecular-switch crossbar circuits,” *Nanotechnology,* vol. 14, no. 4, pp. 462-468, 2003.

[6] M. Haselman and S. Hauck, “The future of integrated circuits: a survey of nanoelectronics,” *Proc. IEEE,* vol. 98, no. 1, pp. 11-38, Jan. 2010.

[7] A. DeHon and H. Naeimi, “Seven strategies for tolerating highly defective fabrication,” *IEEE Des. Test Comput.,* vol.22, no.4, pp. 306-315, 2005.

[8] T. Hogg and G. Snider, "Defect-tolerant adder circuits with nanoscale crossbars," *IEEE Trans. Nanotechnol.*, vol. 5, no. 2, pp. 97-100, Mar. 2006.
[9] T. Hogg and G. Snider, "Defect-tolerant logic with nanoscale crossbar circuits," *J. Electr. Testing: Theor. Appls,* vol. 23, no. 2-3, pp. 117-129, 2007.
[10] Y. Yellambalase and M. Choi, "Cost-driven repair optimization of reconfigurable nanowire crossbar systems with clustered defects," *J. Syst. Archit.*, vol. 54, no. 8, pp. 729-741, Aug. 2008.
[11] W. Rao, A. Orailoglu, and R. Karri, "Logic mapping in crossbar-based nanoarchitectures," *IEEE Des. Test Comput.*, vol. 26, no. 1, pp. 68-76, Jan. 2009.
[12] M. B. Tahoori, "Application-Independent Defect Tolerance of Reconfigurable Nanoarchitectures," *ACM J. Emerging Technol. Comput. Syst.*, vol. 2, no. 3, pp. 197-218, Jul. 2005.
[13] A. Al-Yamani, S. Ramsundar, and D. K. Pradham, "A Defect Tolerance Scheme for Nanotechnology Circuits," *IEEE Trans. Circ. Syst. I, Regul. Pap.*, vol. 54, no. 11, pp. 2402-2409, Nov. 2007.
[14] M. S. Garey and D. S. Johnson, *Computers and Intractibility: A Guide to NP-Completeness*, Freeman, New York, 1979.
[15] J.R. Ullmann, "An Algorithm for Subgraph Isomorphism," *J. Assoc. Comput. Mach.*, vol. 23, no. 1, pp. 31-42, Jan. 1976.
[16] L. P. Cordella, P. Foggia, C. Sansone and M. Vento, "A (sub)graph isomorphism algorithm for matching large graphs," *IEEE Trans. Pattern. Anal. Mach. Intell.*, vol. 26, no. 10, pp. 1367-1372, Oct. 2004.
[17] M. O. Simsir, S. Cadamibi, F. Ivancic, M. Roetteler, and N. K. Jha, "A hybrid nano-CMOS architecture for defect and fault tolerance," *ACM J. Emerging Technol. Comput. Syst.*, vol. 5, no. 3, Article 14, Aug. 2009.
[18] S. Goren, H. F. Ugurdag, and O. Palaz, "Defect-aware nanocrossbar logic mapping through matrix canonization using two-dimensional radix sort," *ACM J. Emerging Technol. Comput. Syst.*, vol. 7, no. 3, Article 12, Aug. 2011.
[19] N. Krasnogor and J. Smith, "A tutorial for competent memetic algorithms: Model, taxonomy, and design issues," *IEEE Trans. Evol. Comput.*, vol. 9, no. 5, pp. 474-488, Oct. 2005.
[20] X. S. Chen, Y. S. Ong, M. H. Lim, and K. C. Tan, "A Multi-Facet Survey on Memetic Computation," *IEEE Trans. Evol. Comput.,*, vol. 15, no. 5, pp. 591-607, Oct. 2011.
[21] M. Tang and X. Yao, "A memetic algorithm for VLSI floorplanning", *IEEE Trans. Syst., Man, Cybern. Part B: Cybern.*, vol. 37, no. 1, pp. 62-69, Feb. 2007.
[22] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2001.
[23] DeHon, "Nanowire-based programmable architectures," *ACM J. Emerging Technol. Comput. Syst.*, vol. 1, no. 2, pp. 109-162, Jul. 2005.
[24] J. Dai, L. Wang, and F. Jain, "Analysis of defect tolerance in molecular crossbar electronics," *IEEE Trans. VLSI Syst.*, vol. 17, no. 4, pp. 529-540, Apr. 2009.
[25] M. Crocker, X. S. Hu, and M. Niemier, "Defects and faults in QCA-based PLAs," *ACM J. Emerging Technol. Comput. Syst.*, vol. 5, no. 2, Article 8, Jul. 2009.
[26] E. M. Luks, "Isomorphism of Graphs of Bounded Valence can be Tested in Polynomial Time," *J. Computer System Science*, pp. 42-65, 1982.
[27] J. He and X. Yao, "Maximum Cardinality Matching by Evolutionary Algorithms," In *Proceedings of the 2002 UK Workshop on Computational Intelligence*, pp.53-60, Sep. 2002.
[28] Y. Zheng and C. Huang, "Defect-aware logic mapping for nanowire-based programmable logic arrays via satisfiability," In *Proceedings of the Conference on Design Automation and Test in Europe*. pp. 1279-1283, Apr. 2009.
[29] C. Tune and M. B. Tahoori, "Variation Tolerant Logic Mapping for Crossbar Array Nano Architectures," *In Proc. 15th Asia and South Pacific Design Automation Conference*, pp. 858-860, Jan. 2010.
[30] B. Yuan and B. Li, "Coverage optimization for defect-tolerant logic mapping on nanoelectronic crossbar architectures," *J. Comput. Sci. Technol.*, vol. 27, no. 5, pp. 979-988, Sept. 2012.
[31] P. Merz and F. Bernd, "Fitness landscape analysis and memetic algorithms for the quadratic assignment problem," *IEEE Trans. Evol. Comput.*, vol.4, no.4, pp. 337-352, Nov. 2000.
[32] S. Salcedo-Sanz and X. Yao, "A hybrid hopfield network-genetic algorithm approach for the terminal assignment problem," *IEEE Trans. Syst., Man, Cybern. Part B: Cybern.*, vol. 34, no. 6, pp. 2343-2353, Dec. 2004.
[33] S. Salcedo-Sanz, Y. Xu and X. Yao, "Hybrid meta-heuristics algorithms for task assignment in heterogeneous computing systems," *Comput. Oper. Res.*, vol. 33, no. 3, pp. 820-835, 2006.
[34] H. Lau, T. M. Chan and W. T. Tsui, "Item-location assignment using fuzzy logic guided genetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 12, no. 6, pp. 765-780, Dec. 2008.
[35] C. F. T. Soares, A. C. de Mesquita Filho, and A. Petraglia, "Optimizing capacitance ratio assignment for low-sensitivity SC filter implementation," *IEEE Trans. Evol. Comput.*, vol. 14, no. 3, pp. 375-380, Jun. 2010.
[36] K.-H. Liang, X. Yao, and C. Newton, "Evolutionary search of approximated N-dimensional landscapes," *Int. J. Knowledge-Based Intell. Eng. Syst.*, vol. 4, no. 3, pp. 172-183, Jul. 2000.
[37] E. S. Buffa, G. C. Armour, and T. E. Vollmann, "Allocating facilities with CRAFT," *Harvard Business Review*, vol. 42, pp. 136-158, Mar. 1964.
[38] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Computing*, vol. 9, no.1, pp. 3-12, 2005.
[39] Y. Jin, M. Olhofer, and B. Sendhoff, "A framework for evolutionary optimization with approximate fitness functions," *IEEE Trans. Evol. Comput.*, vol. 6, no. 5, pp. 481-494, Oct. 2002.
[40] Y. Su and W. Rao, "Runtime analysis for defect-tolerant logic mapping on nanoscale crossbar architectures," *In Proceedings of the IEEE/ACM International Symposium on Nanoscale Architectures*, pp. 75-78, Jul. 2009
[41] Bo Yuan, Bin Li, Thomas Weise, and Xin Yao, "A New Memetic Algorithm with Fitness Approximation for the Defect-Tolerant Logic Mapping in Crossbar-based Nano-architectures," IEEE Transactions on Evolutionary Computation (IEEE-EC), vol. 18, no. 6, pp. 846-849, December 2014, doi: 10.1109/TEVC.2013.2288779

Bo Yuan received the B.S. degree in electronic information science and technology from University of Science and Technology of China (USTC), China, in 2009. He is currently pursuing the Ph.D. degree in electronic science and technology from USTC. Between 2012- 2013, he was an exchange Ph. D. student with the Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), School of Computer Science, University of Birmingham, UK. His current research interests include computational intelligence, electronic design automation, graph theory and machine learning.

Xin Yao (M'91—SM'96—F'03) is a Chair (Professor) of Computer Science and the Director of CERCIA (the Centre of Excellence for Research in Computational Intelligence and Applications), University of Birmingham, UK. He is an IEEE Fellow and a Distinguished Lecturer of IEEE Computational Intelligence Society (CIS). His work won the 2001 IEEE Donald G. Fink Prize Paper Award, 2010 IEEE Transactions on Evolutionary Computation Outstanding Paper Award, 2010 BT Gordon Radley Award for Best Author of Innovation (Finalist), 2011 IEEE Transactions on Neural Networks Outstanding Paper Award, and many other best paper awards at conferences. He won the prestigious Royal Society Wolfson Research Merit Award in 2012 and was selected to receive the 2013 IEEE CIS Evolutionary Computation Pioneer Award. He was the Editor-in-Chief (2003- 08) of IEEE Transactions on Evolutionary Computation. His major research interests include evolutionary computation and ensemble learning. He has more than 400 refereed publications in international journals and conferences.

Bin Li (M'07) received the B.S. degree from Hefei University of Technology, Hefei, China, in 1992, the M.Sc. degree from Institute of Plasma Physics, China Academy of Science, Hefei, China, in 1995, and the Ph.D. degree from University of Science and Technology of China (USTC), China in 2001. He is currently a professor with the School of Information Science and Technology, USTC, Hefei, China. He has authored and co-authored more than 40 refereed publications. His major research interests include evolutionary computation, memetic algorithms, pattern recognition, and real-world applications. Dr. Li is the Founding Chair of IEEE CIS Hefei Chapter, Counselor of IEEE USTC Student Branch, senior member of Chinese Institute of Electronics (CIE), member of the Technical Committee of Electronic Circuits and Systems Section of CIE.

Thomas Weise (M'10) received the Diplom Informatiker (equivalent to M.Sc.) degree from the Department of Computer Science, Chemnitz University of Technology, Chemnitz, Germany, in 2005, and the Ph.D. degree at the Distributed Systems Group of the Fachbereich Elektrotechnik and Informatik, University of Kassel, Kassel, Germany in 2009. He then took a position post-doctoral researcher at the School of Computer Science and Technology, University of Science and Technology of China (USTC), Hefei, China. He now is Associate Professor at the USTC-Birmingham Joint Research Institute in Intelligent Computation and Its Applications (UBRI) which is part of the same school. His major research interests include the Traveling Salesman Problem, planning for logistics applications, Evolutionary Computation, Genetic Programming, and real-world applications of optimization algorithms. His experience ranges from applying GP to distributed systems and multi-agent systems, efficient web service composition for Service Oriented Architectures, to solving large-scale real-world vehicle routing problems for multimodal logistics and transportation. Besides being the author/co-author of over 60 refereed publications, Dr. Weise also authors the electronic book *Global Optimization Algorithms -- Theory and Application* which is freely available at his website (http://www.it-weise.de/).