IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 29, 2019, accepted January 8, 2020, date of publication January 10, 2020, date of current version January 17, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2965723

Dynamic Placement Optimization for Bio-Inspired Self-Repairing Hardware

LIU XIUBIN[1], QIAN YANLING[1], FENG XIANGLI[1], ZHUO QINGQI[2], AND LI YUE[1]
[1]Science and Technology on Integrated Logistics Support Laboratory, National University of Defense Technology, Changsha 410073, China
[2]School of Aerospace Engineering, Tsinghua University, Beijing 100084, China

Corresponding author: Li Yue (liyue@nudt.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 51675523.

ABSTRACT Bio-inspired self-repairing hardware is a distributed self-adaptive system, characterized by powerful fault-tolerant ability and environment adaptivity. However, it suffers from some difficulties such as large resource consumption and degraded circuit performances. From the viewpoint of cybernetics and computer science, the cellular differentiation and substitution process of bio-inspired self-repairing hardware can be converted into dynamic placement problems on a reconfigurable system. Current systems can only generate some predefined fault-free placements from a finite number of initial placements. The aim of this paper was to achieve high-quality placements from arbitrary initial placements. Based on P systems, an analysis has been made on the limitations of current systems and an improved computing system has been developed to achieve the ergodic property. Its computational power has been verified by a constructive proof. Moreover, centering on the problem how to improve the placement quality on a distributed platform, the optimization model, task allocation, optimization strategy, and membrane optimization algorithm have been designed and developed. The optimization performances were verified and the calculation amount was exhibited by experiments. Finally, it indicated by comparison that the proposed approach would reduce the resource consumption and maintain good circuit performances.

INDEX TERMS Bio-inspired self-repairing hardware, dynamic placement optimization, P systems, computing model, membrane optimization algorithms.

I. INTRODUCTION

Bio-inspired self-repairing hardware is a distributed self-adaptive system, characterized by fault-tolerant ability and environment adaptivity [1]. Compared with Triple Modular Redundancy, it is considered as an alternative and more promising fault tolerant approach for some special-purpose electronic devices that have to withstand harsh conditions and meet stringent requirements on reliability [2], [3]. The basic idea of such systems is to construct an array made up of many decentrally organized and interacting electronic cells (eCells). As the basic ''building blocks'' in analogy to cells in multicellular organisms, eCells can implement functional differentiation and change connections according to the stored configuration information (genome) [4], [5]. Fault recovery is generally achieved by deactivating the fault eCell and activating another spare one for differentiation and substitution [6]. However, in contrast to the self-healing processes of organisms, the positions of eCells are fixed in the array and no eCell can be created, moved or disappeared. As a result, the cellular differentiation and substitution method suffers from several difficulties.

1) A large number of spare eCells are initially arranged in places adjacent to working ones for substitution, leading to large resource consumption.
2) A predefined point-to-point substitution method can hardly make any flexible adjustment according to the changing environment, leading to inefficient consumption of spare eCells and degraded circuit performances.

From the standpoint of cybernetics and computer science, bio-inspired self-repairing hardware can be considered as a reconfigurable system, which supports a dynamic arrangement of function blocks on the eCell array in response to the interference caused by faults. In this framework, the substitution process is a dynamic placement method where the placement evolves by a series of transitions at run time [7]. Current systems can only generate some predefined fault-free

The associate editor coordinating the review of this manuscript and approving it for publication was Qi Zhou.

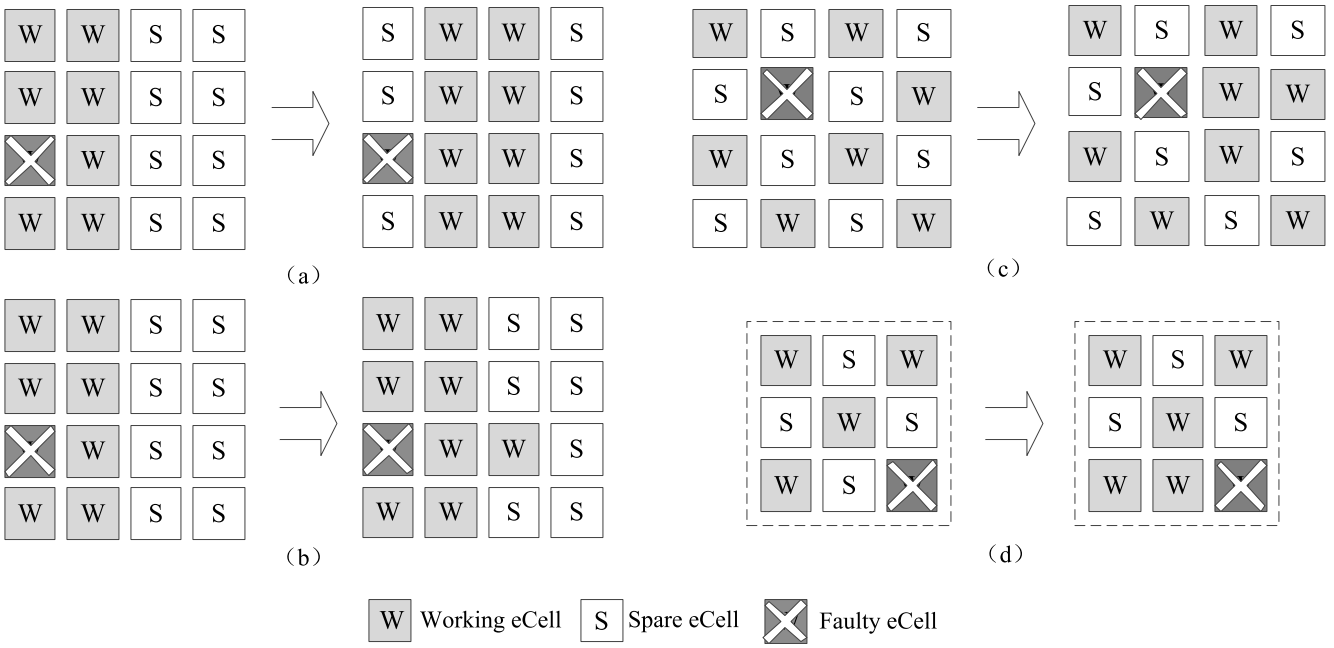


FIGURE 1. Dynamic placement methods with different initial placements (a,b –Embryonics's dynamic placement method, c – Yang's dynamic placement method, d – Szasz's dynamic placement method).

placements from a finite number of initial placements. However, it may lose efficiency when the arrangement of working and spare eCells is changed in the initial placement or when high-quality placements are needed in the self-repairing process.

As a result, this paper makes a research on dynamic placement optimization for bio-inspired self-repairing hardware. The final goal is to achieve high-quality fault-free placements from arbitrary initial placements. In mathematics, placement optimization is generally considered as a kind of NP-hard combination optimization problems [8]. In order to solve such problems, the computing system should have enough computational power to generate feasible solutions and an optimization algorithm is needed to select an optimal solution. Unfortunately, due to the predefined point-to-point substitution method, many feasible placement results cannot be generated on the current hardware platform, so the feasible solution space is limited. Moreover, large-scale iterative calculating, which is often used in FPGA offline placement optimization algorithms [9], is limited by online computing resources. Therefore, a computing model based on P systems was proposed to simulate dynamic placement processes and to analyze the computational power of hardware platforms [10]. An improved distributed computing system, called a mesh P system with bidirectional sliding rules, was developed to extend the feasible solution space. Its computational power has been verified by a constructive proof. Moreover, centering on the problem how to improve the placement quality, the optimization model, task allocation, optimization strategy, and membrane optimization algorithm have been designed and developed. The optimization performances were verified and the calculation amount was exhibited by experiments. The following parts are organized as follows. Section II focuses on computing model and systems. Section III explores the optimization model, optimization strategy, and membrane optimization algorithm. Section IV exhibits experimental results. Section V concludes the paper.

II. COMPUTING MODEL AND SYSTEMS

In this section, a computing model based on P systems is proposed to simulate dynamic placement and analyze the limitation of computational power of the current hardware platforms. Then, an improved distributed computing system, called a mesh P system with bidirectional sliding rules, is developed to achieve the ergodic property.

A. RELATED WORKS

In the early bio-inspired self-repairing hardware such as Embryonics [6] and POEtic tissue [11], all the eCells were arranged in a mesh architecture and every eCell had four nearest neighbors, forming a traditional von Neumann neighborhood. Initially, there existed a sufficient number of spare eCells on the right side of the array. When one eCell turned into fault state, all the functions of the column to which the fault eCell belonged, along with that of its following columns, would synchronously shift by one column to the right. This process could be used repeatedly until all the spare columns were occupied (Fig.1(a)). Another dynamic placement method of Embryonics was conducted by limiting the adjustment range to a single row rather than all the columns on the right side of the fault eCell (Fig.1(b)) [12].

The improvement of initial placement can reduce routing complexity. Lala proposed an initial placement where every working eCell was surrounded by two spare eCells except those at the periphery. Every working eCell could be replaced by any available spare neighbor in the event of fault occurrence [13]. Besides, in the self-repairing system proposed by Yang, the working eCells and spare eCells were arranged in a staggered fashion. Every working eCell had four neighboring spare eCells and it could be substituted by one of the four spare neighbors (Fig.1(c)) [14].

In order to increase communication flexibility, a two-layer architecture was proposed by adding an interconnected switch matrix layer on the top of the interconnected eCell layer. An eCell cluster was generally composed of a switch matrix and a finite number of eCells. In the bio-inspired system proposed by Szasz, every cluster was made up of nine cells, including 5 working eCells and 4 spare ones. When a fault occurs, the fault eCell was deactivated and the fault information was propagated in the whole cluster. Then a available spare eCell responded and replaced the fault one (Fig.1(d)) [15]. In the self-adaptive hardware architecture (SANE), placement information could be transferred among different clusters and the scope of substitution was extended [16].

Heterogeneous fabrics are resource-efficient to some special situations. In Unitronics, the architecture was composed of two different types of eCells: core ones in the middle of the array, surrounded by peripheral ones around its perimeter [17]. Core eCells were used for implementing specific functions and peripheral ones managed the flow of input/output information. In the hierarchical self-repairing architecture proposed by Kim, modular redundancy methods were introduced by adding special redundant eCells to the group of gene-encoded ones [18]. Every working eCell had a corresponding redundancy which can replace its function instantly. If the redundancy was used up, a gene-encoded eCell was immediately configured to the redundancy.

B. MODELING AND ANALYSIS

P systems provide powerful tools to study distributed parallel computing systems. On this basis, a neural P system with swap rules is developed as a universal computing model for simulating dynamic placement of different bio-inspired self-repairing hardware, and a computational power theorem is proposed and proved to analyze the limitation of dynamic placement optimization abilities in current platforms.

1) COMPUTING MODEL

A P system, or membrane computing, is a parallel and distributed computational model [10]. It is obtained by formalizing the structure and functioning of living cells, as well as the organization method in tissues or higher order structures. A basic ingredient of a P system is membrane structure, which contains several membranes with a hierarchical structure as in a cell (represented by an Euler–Venn diagram or a directional labeled unordered tree) or a net structure as in tissues or neural networks. A membrane can be regarded as a separator of two regions. It provides a place for multisets of objects to be processed, and supports a selective communication between the inside and outside regions. The multisets of objects present in different regions constitute the *configuration* of the system. When a P system runs, a set of transformations between different configurations take place by the nondeterministic and maximally parallel application of evolution rules. A sequence of transitions form a computation [19].

In a "purely communicative" P system, symport and antiport rules are generally used when two unstructured symbol objects (a and b) pass through membranes in the same or opposite directions, described in a mathematical form (ab, in), (ab, out) (symport), and $(a, in; b, out)$ (antiport) [20]. Nevertheless, it will lose effectiveness when simulating the trans-membrane transport based on public passages, where the protein passages of adjacent membranes can be automatically closed to avoid the spread of toxic substances and reopen when these substances disappear. In this process, substances are exchanged, if and only if the public passage is open, formalized by a new rule $(i \leftrightarrow j)\,|_p$ (swap rule), representing that all the objects are exchanged, if and only if there is a synapse between membrane i and j and both membranes permit this swap operation. It can be noted that there is no difference in computation whether one membrane has one or more objects, thus all the objects in one non-empty region are regarded as one symbol. There is either one or none symbol in one membrane.

In some cases, there are restrictions on the swap process. Given a synapse between membrane i and j, if it supports a bidirectional transport but only one symbol can pass, the swap operation is achieved when one membrane is empty and another is not, denoted by $(i \rightleftharpoons j)\,|_p$ (bidirectional sliding rule); if it only supports a unidirectional transport from membrane i to j, the swap operation is achieved when membrane j is empty and i is not, denoted by $(i \rightarrow j)\,|_p$ or $(j \leftarrow i)\,|_p$ (unidirectional sliding rule) with an arrow indicating the sliding direction.

Then, we can develop a neural P system with swap rules, of a construct form

$$\Pi = (O, \sigma_1, \cdots, \sigma_m, syn, out) \tag{1}$$

where:

1) O is a finite non-empty alphabet of symbol objects;
2) $\sigma_1, \cdots, \sigma_m$ are membranes, of the form

$$\sigma_i = (c_i, R_i), 1 \leqslant i \leqslant m \tag{2}$$

where:

a) $c_i \in O$ is the symbol of membrane σ_i;
b) R_i is a finite set of swap rules of the general form $(i \leftrightarrow j)\,|_p$, associated with σ_i and σ_j;

3) $syn \subseteq \{1, 2, \cdots, m\} \times \{1, 2, \cdots, m\}$ is the set of links (or *synapses* in P systems) among membranes;

TABLE 1. Transformations from bio-inspired self-repairing hardware to P systems.

Bio-inspired self-repairing hardware	P systems
eCells	membranes
function blocks	symbol objects
routing nets	synapses
function substitution	evolution rules
dynamic placement	computation
placement	calculation output
ergodic property	computational power

4) $out = c_1c_2 \cdots c_m$ is the character string in the terminal configuration, used as calculation output.

In the system Π, computation starts with an initial configuration. At every time unit, if a swap rule $(i \leftrightarrow j)\,|_p$ is used, membrane σ_i and σ_j will exchange their symbols. The use of rules is sequential in membranes, but it is parallel from the perspective of the system. When computation halts, the terminal configuration is considered as the output result. It can be noted that Π obeys the object conservation law. No object is created, modified or disappeared in the computation process.

If an extra restriction is made that every membrane contains no more than one symbol in the initial configuration, Π can be used as a universal computing model to simulate dynamic placement and analyze the limitation of computational power of current hardware platforms (Table 1).

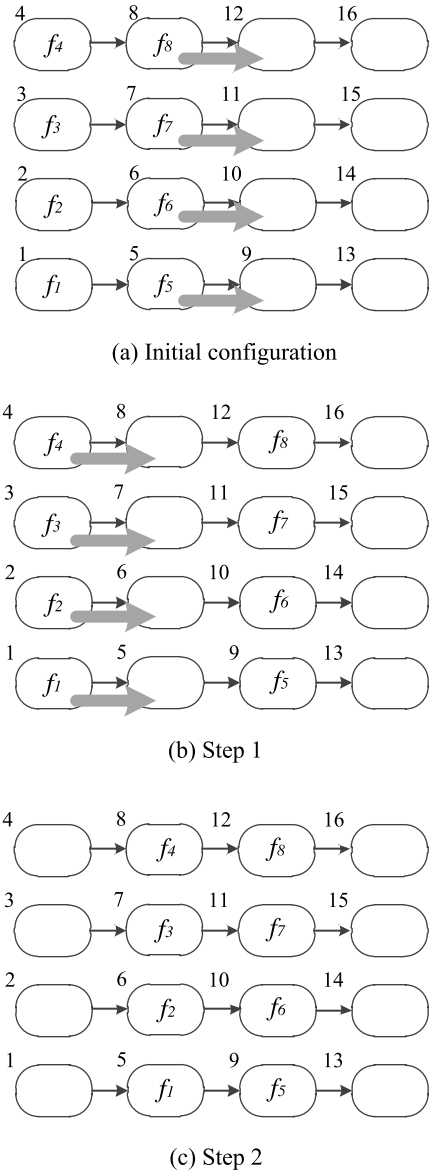
The following example will exhibit this transformation. Let us now consider the dynamic placement process of Embryonics (Fig.1 (a)). It can be simulated by an equivalent computing model

$$\Pi_1 = (O, \sigma_1, \cdots, \sigma_{16}, syn, out) \tag{3}$$

where:

$$\begin{aligned}
O &= \{f_1, f_2, \cdots, f_8\}, \\
\sigma_i &= \left\{f_i, (i \to i+4)\,|_p\right\}, i \in \{1, 2, 3, 4\}, \\
\sigma_i &= \left\{f_i, (i-4 \to i)\,|_p, (i \to i+4)\,|_p\right\}, i \in \{5, 6, 7, 8\}, \\
\sigma_i &= \left\{(i-4 \to i)\,|_p, (i \to i+4)\,|_p\right\}, i \in \{9, 10, 11, 12\}, \\
\sigma_i &= \left\{(i-4 \to i)\,|_p\right\}, i \in \{13, 14, 15, 16\}, \\
syn &= \{(1, 5), (5, 9), (9, 13), (2, 6), (6, 10), (10, 14), \\
&\quad (3, 7), (7, 11), (11, 15), (4, 8), (8, 12), (12, 16)\}, \\
out &= c_1c_2 \cdots c_{16}.
\end{aligned}$$

The structure is seen in Fig.2 (a). When a fault signal occurs in the membrane σ_2, the left three columns are activated and a computation starts. In one step, the rules $(5 \to 9)\,|_p$, $(6 \to 10)\,|_p$, $(7 \to 11)\,|_p$, $(8 \to 12)\,|_p$ are used in a parallel way and four objects are passed from the second column to the third (Fig.2 (b)). In the next step, another four rules $(1 \to 5)\,|_p$, $(2 \to 6)\,|_p$, $(3 \to 7)\,|_p$, $(4 \to 8)\,|_p$ are active and objects of the first column are passed to the second (Fig.2 (c)). Since there is no available rule in the left three columns at the current configuration, the computation halts.



(a) Initial configuration

(b) Step 1

(c) Step 2

FIGURE 2. The dynamic placement process of Embryonics can be simulated by Π_1.

It is worth noting that membranes in Π_1 are connected by unidirectional synapses and form four isolated groups. Objects can only be transported along a unidirectional path Meanwhile, no exchange occurs between different groups. Due to weak computational power, the system Π_1 is not suitable for more complex computing task.

2) COMPUTATIONAL POWER

The computational power of the system Π can be measured by the ergodic property. Given a system Π, if starting from any initial configuration, Π can go through all the possible combinations of configuration character string, Π is ergodic. The ergodic system has the maximal computational power. The following part will investigate the power of the universal model with swap rules and exhibit necessary conditions for the ergodic property.

If all the synapses in a P system support bidirectional transport, the membrane structure can be described by an undirected graph $G = (V, L)$, comprising a set V of vertexes together with a set L of edges [21]. The degree of a vertex, represented by $deg(v)$, is the number of edges that connect to it. If a graph starts from any vertex and can walk along the edge to any other vertex, it is said that the graph is connected. If there exists a closed path, which starts at one vertex, goes along non-repetitive edges and vertexes, and returns to the starting vertex, it can be called a circle [22]. A circle is a special connected graph. As far as connectivity is concerned, there are two lemmas, which will be used in the latter computational power theorem.

Lemma 1: Given a connected graph $G = (V, L)$, if a vertex $v \in V$, of the degree $deg(v) = 1$, together with its own edge $e \in L$, is removed from G, the remaining part $G - v - e$ is still connected.

Lemma 2: Given a connected graph $G = (V, L)$ and its subgraph $H \subseteq G$, H is a circle, if one edge $e \in L(H)$ on H is removed, the remaining part $G - e$ is still connected.

Then the computational power of P systems with swap rules can be exhibited in the following result.

Theorem 1: Given a P system with swap rules Π, if it has a connected membrane structure, Π is ergodic.

One object can be transferred to any membrane in a connected structure. Nevertheless, the movement can change the positions of other objects due to a series of swap operations. It is necessary to protect the objects, which have been in the right place, from adjustments. Here we give a constructive proof.

Proof: Given a P system with swap rules Π, if it has a connected membrane structure, for any initial configuration, there are two steps to generate target configuration character string. In one step, if there exists a membrane σ_i with only one single link and its target symbol is T, T can be transferred to σ_i by swap rules. Then σ_i, together with its link, is removed from the current membrane structure. According to Lemma 1, the remaining membrane structure is still connected. Repeat this step, until all the membranes associated with only one link are removed. Then go to the next step. In the next step, given a membrane structure G with all membranes having more than one links, there exists one circle structure, denoted by $H \subseteq G$. Remove one edge of H. According to Lemma 2, the remaining structure is still connected. Repeat this step, until there exists a membrane with only one link. Then go to the first step. The iteration continues until all the membranes are removed from the structure. At this configuration, the current configuration character string is equal to the target string (Fig.3). Hence, Π is ergodic.

C. DEVELOPMENT OF COMPUTING SYSTEMS

In order to overcome the limitations of current computing systems, we can introduce a new ergodic system for dynamic placement optimization, called mesh P systems with bidirectional sliding rules, of a construct form:

$$\Gamma = (O, \sigma_1, \cdots, \sigma_m, syn, out). \quad (4)$$

It has the same definition with the system Π, except that each membrane is arranged on a rectangular grid and connected with its nearest neighbors in its horizontal and vertical directions (the mesh membrane structure), and all the links support bidirectional sliding rules.

In the system Π, one membrane has to simultaneously play two roles: a requester to send swap requests and an arbiter to select one request, leading to conflicts between swap operations. In the system Γ, however, conflicts can be resolved by the interaction of empty and non-empty membranes. For each sliding task bound to a non-empty membrane σ_n, there are three basic statuses: *waiting*, *ready*, and *executing*. If there is no adjacent empty membrane, σ_n stays in *waiting* status with all the computations paused. When one neighbor σ_e is empty, σ_n turns into *ready* status and is endowed with time slices to send a request to σ_e. If σ_e accepts the swap request, σ_n turns into *executing* status and slides its object to σ_e. If rejected, σ_n returns to the *waiting* status and waits for another empty neighbor. In order to ensure a fair task scheduling, a random selection is made in the arbitration process of empty membranes. Besides, the uncertain moving paths of empty membranes also provides "executing" opportunities for as many sliding operations as possible.

Though endowed with a connected structure, Γ can not always have an equivalent computational power to system Π. A famous example is ''14-15 Puzzle'', which originated from the 15 Puzzle [23]. 15 Puzzle is a kind of sliding puzzles, which challenge players to solve the puzzle in a two-dimensional world. Initially, there are 15 pieces placed in a 4×4 square frame with an empty space on the bottom right. Only sliding moves that use the empty space are allowed in

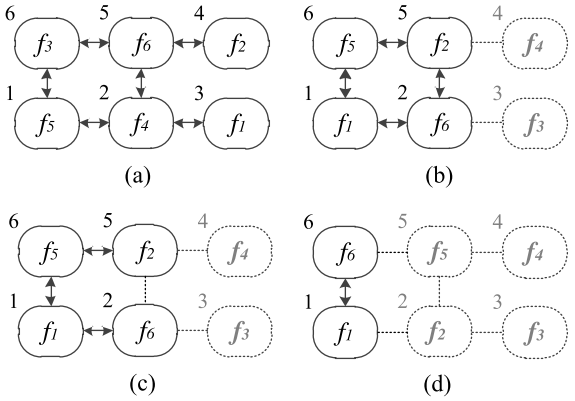


FIGURE 3. An example of the constructive proof. Let us consider a P system Π_2, of the initial character string $f_5f_4f_1f_2f_6f_3$ and the target string $f_1f_2f_3f_4f_5f_6$. (a) The membrane structure and initial configuration; (b) Since the membrane σ_3 and σ_4 have only one link, f_3 and f_4 are transferred to the target membranes and these two membranes, together with their links, are removed from the current membrane structure; (c) Membrane $\sigma_1\sigma_2\sigma_5\sigma_6$ form a circle, so a link (2, 5) is removed; (d) Since the membrane σ_2 and σ_5 have only one link, f_2 and f_5 are transferred to the target membranes. At the current configuration, the character string is $f_1f_2f_3f_4f_5f_6$.

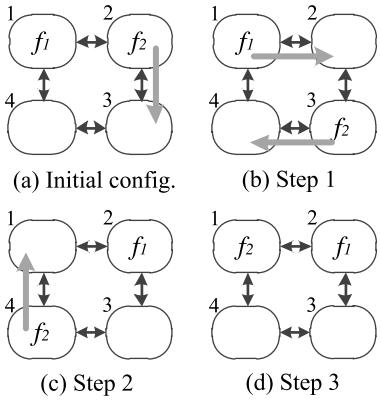


FIGURE 4. Sliding rules can simulate any swap rule in a 2 × 2 mesh structure. For example, in order to simulate the swap rule $(1 \leftrightarrow 2)\,|_p$ in the initial configuration, there are three steps. Step 1: $(2 \rightarrow 4)\,|_p$; Step 2: $(1 \rightarrow 2)\,|_p$, $(3 \rightarrow 4)\,|_p$; Step 3: $(4 \rightarrow 1)\,|_p$.

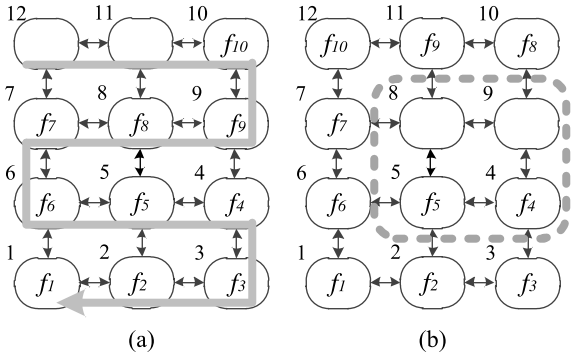


FIGURE 5. Given a system Γ_2 with more membranes, a horizontal swap rule $(4 \leftrightarrow 5)\,|_p$ can be simulated by sliding rules. (a) The initial configuration and a snaking moving path. (b) Empty regions move along the snaking path until membrane σ_4 and σ_5, together with membrane σ_8 and σ_9, construct a 2 × 2 cluster. In this cluster, $(4 \leftrightarrow 5)\,|_p$ can be simulated by sliding rules.

the game. Then it is impossible to get a new configuration with only piece 14 and 15 exchanged.

Experiments have shown that "14-15 Puzzle" can be solved, if there are two empty spaces. Then we will verify these experiments in the computational power theorem of the system Γ.

Theorem 2. Given a mesh P system with bidirectional sliding rules Γ, Γ *has a* $m \times n$ *mesh membrane structure,* $m \geqslant 2, n \geqslant 2$, *if there exists at least two empty membranes,* Γ *is ergodic.*

It has been verified that swap rules can generate arbitrary character string in a connected structure. If sliding rules can simulate swap rules, theorem 2 is proved.

Proof: Let's begin with a P system Γ_1 with a 2 × 2 mesh membrane structure and two empty membranes. As is shown in Fig.4, there are four swap rules at most: $(1 \leftrightarrow 2)\,|_p$, $(2 \leftrightarrow 3)\,|_p$, $(3 \leftrightarrow 4)\,|_p$, $(4 \leftrightarrow 1)\,|_p$. By simple experiments, each swap rule can be simulated by sliding rules. So the theorem is established, when $m = 2$ and $n = 2$.

Then we will verify the conclusion in a P system Γ_2 with a $m \times n$ mesh membrane structure and two empty membranes, $m \geqslant 2, n \geqslant 2$. Given a horizontal swap rule, denoted by $(i \rightarrow j)\,|_p$, $1 \leqslant i, j \leqslant mn$, associated with membrane σ_i and its horizontal neighbor σ_j, the two empty regions can move along a snaking path by sliding rules until they reach to one side of membrane σ_i and σ_j (Fig.5(a)). At this time, the two empty membranes, together with membrane σ_i and σ_j, can construct a 2 × 2 membrane cluster similar to Γ_1 (Fig.5(b)). In this cluster, two symbols can be exchanged with sliding rules. Then, two empty regions return to the initial positions along the same snakelike path. In the current configuration, only membrane σ_i and σ_j exchange their symbols and the others keep unchanged, thus the horizontal swap rule is simulated by sliding rules. In a similar way, vertical swap rules can be simulated by transposing system Γ_2. So the theorem is established, when $m \geqslant 2, n \geqslant 2$.

III. ALGORITHM DESIGN

Limited by online computing resources and distributed architecture, wire length driven placement models and simulated annealing algorithms, which are often used in the offline placement, can hardly be applied on bio-inspired self-repairing hardware. Hence, the optimization model and strategy are developed to improve the quality of placements. The membrane optimization algorithm is proposed to achieve an optimal placement from a fault one.

A. OPTIMIZATION MODEL AND STRATEGY

1) SOME DEFINITIONS

In mathematics, placement is an injective mapping from the object set to the membrane set, denoted by

$$\beta : O \rightarrow \{\sigma_1, \cdots, \sigma_m\}. \tag{5}$$

Correspondingly, the global placement is denoted by $\beta(O)$; the local placement of an object set $F \subset O$ is denoted by $\beta(F)$; the mapping membrane of an object $f_i \in O$ is denoted by $\beta(f_i)$.

Given any mesh or quasi mesh membrane structure μ where a neighbor of one membrane can only be placed on the vertical or horizontal direction, if only translational motion is supported by the coordinate system, μ can be represented by a matrix. The number of row and column vectors is equal to the length and width of the minimum enclosing rectangle of μ. If there is a membrane arranged on one rectangular grid, the corresponding position of the matrix is denoted by 1, otherwise, the position is 0 (Fig.6). Two membrane structure are equal if and only if their matrix representations are the same.

Given two placements with the same membrane structure, (β_A and β_B), for an object f_i in β_A, if f_i also exists in β_B, the object displacement of f_i between β_A and β_B, denoted by $d(\beta_A(f_i), \beta_B(f_i))$, is calculated by the Manhattan distance between two points $(\beta_A(f_i), \beta_B(f_i))$ in the same matrix

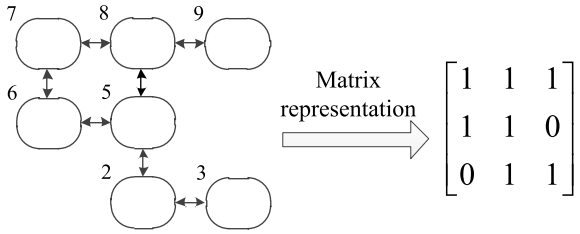


FIGURE 6. Matrix representation of the quasi mesh membrane structure.

representation; if f_i does not exist in β_B, $d(\beta_A(f_i), \beta_B(f_i))$ is equal to the semi-perimeter of the minimum enclosing rectangle.

The distance between two placements (β_A and β_B) can be calculated by the sum of object displacements

$$d(\beta_A, \beta_B) = \sum_{1}^{k} d(\beta_A(f_i), \beta_B(f_i)). \quad (6)$$

2) OPTIMIZATION MODEL

Given a P system Γ, the dynamic placement optimization process starts from an arbitrary initial placement β_{init}, goes through many configuration transitions, and ends with an optimal placement β_{opt} that is close to the best placement β_{tar} in a fault-free system. Since objects can not be arranged in fault membranes, β_{opt} is generally not equal to β_{tar}, but they are close enough for β_{opt} to inherit optimization performances of β_{tar}.

The system optimization model can be denoted by

$$\min d\left(\beta_{opt}(O), \beta_{tar}(O)\right) \quad (7)$$

subject to

$$\beta_{opt}(O) \in U(\beta_{tar}(O)) \quad (8)$$

where $U(\beta_{tar}(O))$ is the neighborhood of $\beta_{tar}(O)$.

Furthermore, system Γ has natural parallelism and the overall optimization task should be divided into many subtasks to be executed in different membranes [24], [25]. The task allocation method in this paper can be described in the following way. Given an object set $O = \{f_1, f_2, \cdots, f_k\}$, O can be divided into k parts $O = \cup_{i=1}^{k} O_i$, each of which, denoted by O_i, contains at least two elements where one element is f_i. Then, the optimization task can be correspondingly divided into k subtasks, each of which establishes a mapping from one subset of O to a few membranes.

Similarly, the optimization model of one subtask O_i can be denoted by

$$\min d\left(\beta_{opt}(O_i), \beta_{tar}(O_i)\right) \quad (9)$$

subject to

$$\beta_{opt}(O_i) \in U(\beta_{tar}(O_i)) \quad (10)$$

where $U(\beta_{tar}(O_i))$ is the neighborhood of $\beta_{tar}(O_i)$.

3) OPTIMIZATION STRATEGY

Placement optimization is a kind of NP-hard combination optimization problem. It takes superpolynomial time of the input size to achieve the exact solution. However, for online applications, it is important to get a solution in a given time, even though the solution may not be the best [26]. Hill climbing is an optimization technique that belongs to the local search family. It can speed up the searching process. Starting with arbitrary feasible solution, hill climbing can generate iteratively incremental improvements by moving from current solution to a better one. It keeps running when no further improvements are possible or a given deadline is reached [27].

Given a membrane together with its object f_i, it should decide whether f_i in another membrane is better than that in its own one, according to the following equation

$$\delta = d(\beta_{curr}(O_i), \beta_{tar}(O_i)) - d(\beta_{next}(O_i), \beta_{tar}(O_i)) \quad (11)$$

where $\beta_{curr}(O_i)$ and $\beta_{next}(O_i)$ are the current and next placements of the subtask O_i, respectively. If $\delta > 0$, two membranes exchange their objects, otherwise, the system maintains the current placement.

B. MEMBRANE OPTIMIZATION ALGORITHM

From macro perspectives, system Γ works as a synchronous system and there are two stages in our membrane optimization optimization algorithm. The first stage starts when one non-empty membrane turns into the fault state. In this stage, all the empty membranes converge towards the fault membrane until the fault one transmits its object to another fault-free one. In the second stage, objects change their positions between different membranes and configurations are evolved by iterative improvements. The second stage ends when exceeding the maximum number of iterations. In order to reduce the amount of calculation and speed up the searching process, hill climbing strategy is used in the algorithm.

From micro perspectives, the computation is made up of a number of sliding operations. There are three kinds of membranes: non-empty membranes, fault empty membranes, and fault-free empty membranes. In one sliding process, the non-empty membrane sends a request to one of its fault-free empty neighbors. If its neighbor responds that request, the non-empty membrane will transmit its object to its neighbor. If rejected, it will wait for the next cycle. In this process, fault empty membranes do not participate in any activities. Then we will discuss the decision process of fault-free empty membranes and non-empty membranes and the membrane optimization algorithm is shown in Algorithm1.

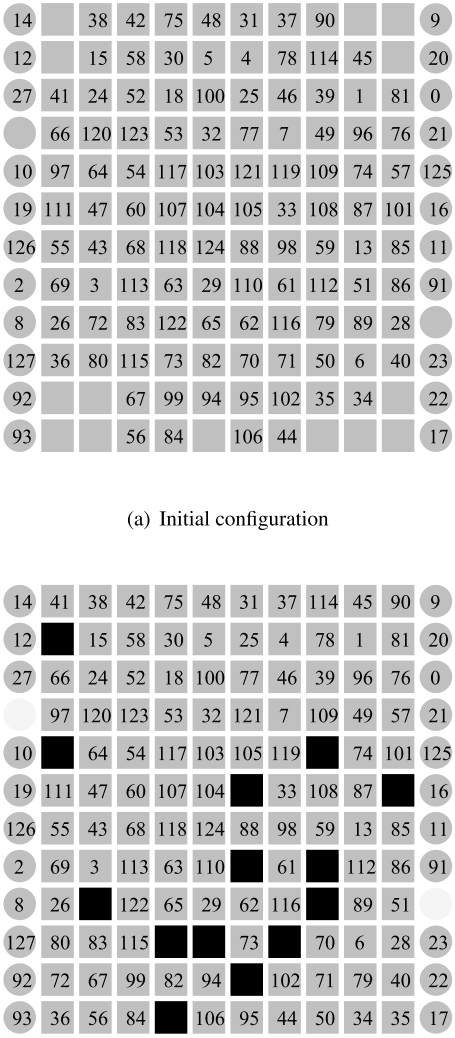
Given a fault-free empty membrane σ_j, if it receives several swap requests, σ_j will randomly choose one swap candidate among all the requests and carry out the swap operation with this candidate.

Given a non-empty membrane σ_i associated with a symbol f_x, it needs some extra information to make a decision. In the first stage, σ_i chooses a fault-free empty neighbor on the opposite direction of the fault membrane σ_f. For example,

Algorithm 1 Membrane optimization algorithm

Initialization *rounds* is the number of iterations, an integer, initially 0.
For each membrane σ_i, run these codes synchronously, when a membrane turns into fault non-empty state:
while $rounds < threshold_r$ **do**
  Step 1 the message function can be defined as:
  if σ_i *is non-empty* **then**
    acquire local configuration information;
    choose a fault-free empty neighbor;
    send Request messages;
  end
  else
    wait until receiving Request messages;
    randomly choose a Request;
    establish a lock with that neighbor;
  end
  Step 2 the swap function can be defined as:
  if *locked* **then**
    swap objects with its locked neighbor;
    unlock with its neighbor;
  end
  Step 3 the stage transformation function is defined as:
  if *swapped and fault* **then**
    broadcast message;
  end
  if *broadcast* **then**
    go to the second stage;
  end
  $rounds = rounds + 1$;
end

if σ_f is on the upper right side of σ_i, σ_i will choose a fault-free empty neighbor on its lower or left side; if σ_f is itself σ_i, σ_i will choose a fault-free empty neighbor on any side. In the second stage, σ_i should decide whether its symbol f_x in another membrane is better than that in its own one and select the best place with the minimum δ in the equation 11.



(a) Initial configuration

(b) Terminal configuration

FIGURE 7. The comparison of initial and terminal placement in the P system Γ. Γ has 10 × 12 membranes (denoted by squares with a grey color for fault-free and black for fault) surrounded by two columns of I/O (denoted by circles). Offline placement optimization algorithms are used for the initial placement of CLB and I/O blocks (denoted by a number, respectively) and the membrane optimization algorithm is used for dynamic placement optimization. (a) Initial configuration. (b) Terminal configuration.

IV. EXPERIMENTAL RESULTS

In this section, optimization performances and the calculation amount of membrane optimization algorithms are verified and analyzed. Dynamic placement performances of the proposed approach are exhibited and identified by a comparison with those of Embryonics, Yang's system, and Szasz's system.

A. FUNCTIONAL VERIFICATION

The benchmark "alu4" from MCNC20 is used as the target circuits to be implemented on the proposed computing system. This benchmark, initially with approximate 934 netlist primitives, is packaged into 106 CLB blocks, 22 I/O blocks, and 685 nets by technology mapping and packing. CLB blocks are logic elements, which can be connected with other resources via interconnects, while I/O blocks are generally located around the perimeter with fixed pads to connect external circuitry. We will use a P system Γ with a 10 × 12 mesh structure to simulate the dynamic placement process of CLB blocks, each of which can be regarded as a symbol object. The initial placement β_{init}, also used as the target placement β_{tar}, is generated by wire length driven simulated annealing algorithms (Fig.7 (a)).

The system Γ starts working when a fault signal is inserted into one randomly selected non-empty membrane. A swap decision is made by a comparison between the target and current local placement in a 5 × 5 rectangular area.

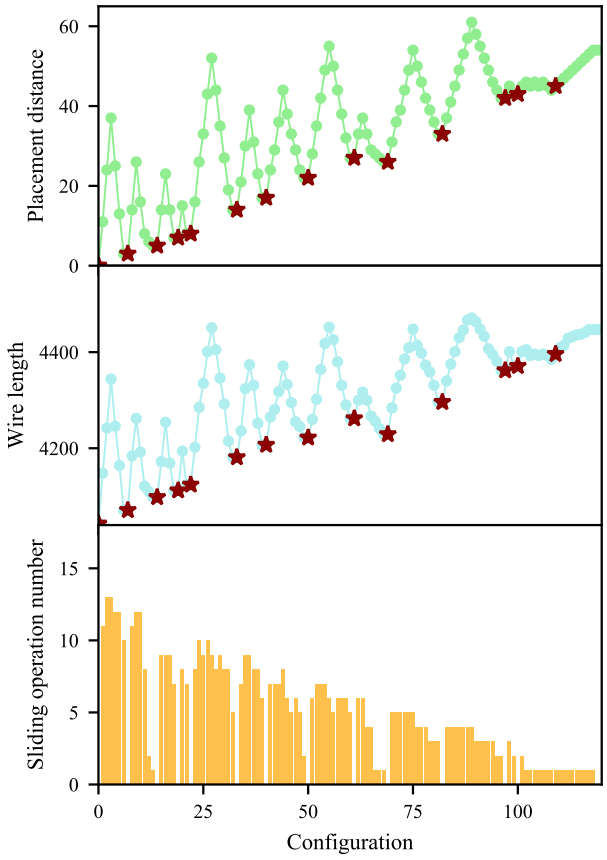


FIGURE 8. The entire computation process of Γ with a comparison of placement distance, wire length, and sliding operation number in every configuration. The pentagram refers to the configuration when a fault signal is inserted into the system.

When the computation halts, another membrane is successively set to fault state and Γ restarts the computing. This process goes on until all the empty membranes are used up (Fig.7 (b)).

There are 14 empty membranes in the initial placement and 14 faults are inserted into the system. The entire computation process is shown in Fig.8. At each configuration, the placement distance (d (β_{curr}, β_{tar})), the total wire length based on the semi-perimeter model, and the number of sliding operations are calculated and counted.

When a fault signal is inserted, the placement difference increases first, then decreases, presenting like a "reverse V" and corresponding to two stages in the membrane optimization algorithm (Fig.8(a)). The local maximum, representing the end configuration of the first stage, changes without regularity. It indicates that fault positions are chosen randomly. The local minimum, representing the end configuration of the optimization stage, shows a gradually increasing trend. It identifies that the optimization performances are related with the number of faults (or the number of empty membranes). The system usually goes through a big drop in the optimization stage, however, in the last three fault insertion experiments, the curve decreases with a very small decline, representing that the optimization performances at these configurations are weak. It indicates that the optimization performances will make a rapid decline when there are only a few empty membranes.

Let us make a comparison between the placement distance and wire length (Fig.8(b)), which are used as the cost value in the proposed approach and offline placement algorithms. These two curves are highly similar (the correlation coefficient is 0.9907), indicating that the placement distance can be used as a measure for circuit performances.

There are 118 configurations and 556 sliding operations in the entire computation process (Fig.8(c)). In one fault insertion experiment, there are nearly 40 sliding operations in average. Compared with offline placement, the optimization can be achieved rapidly with a reduction of the amount of calculation. In addition, the system Γ has parallel power, which is related with the number of empty membranes. The parallelism can reduce the computing time and accelerate the computing speed.

B. PERFORMANCE COMPARISON AND ANALYSIS

In order to exhibit the dynamic placement performance of the proposed approach, denoted by Γ, a comparison is made with Embryonics, Yang's hardware, and Szasz's hardware. Based on neural P systems with swap rules, the equivalent computing systems of different hardware (Π_{Embry}, Π_{Yang}, and Π_{Szasz}, respectively) are designed to guarantee that each system can implement benchmark "alu4" and make a recovery from no less than 14 faults.

In the system Π_{Embry}, a 36×5 membrane structure is used to reduce the consumption of membranes in dynamic placement. All the membranes in the same horizontal direction are connected by unidirectional synapses, forming a chain structure, while isolated in the vertical direction. Symbol objects are initially placed at the left part of the system to maximize fault tolerant ability, and I/O blocks are located on the upper side for convenience of rerouting (Fig.9 (a)).

In the system Π_{Yang}, the number of membranes is twice the number of objects, so a 12×18 mesh membrane structure with unidirectional synapses is used. Symbol objects are initially arranged in different membranes with a staggered fashion and I/O blocks are located on both sides of these membranes (Fig.9 (b)).

System Π_{Szasz} is made up of 24 isolated clusters arranged on a 6×4 rectangular grid with I/O blocks located on both sides of the system. In one cluster containing 3×3 membranes, 4 membranes on four sides are interconnected by bidirectional synapses, and other 5 used for initial object placement connect to these 4 by unidirectional synapses (Fig.9 (c)).

The initial configurations of four systems are compared in Table 2 with the following items:

1) n_t – total membrane number;
2) n_{init} – number of membranes used for storing objects in the initial placement;
3) n_{sub} – number of membranes used for substitution;
4) n_{fr} – maximal fault recovery number;

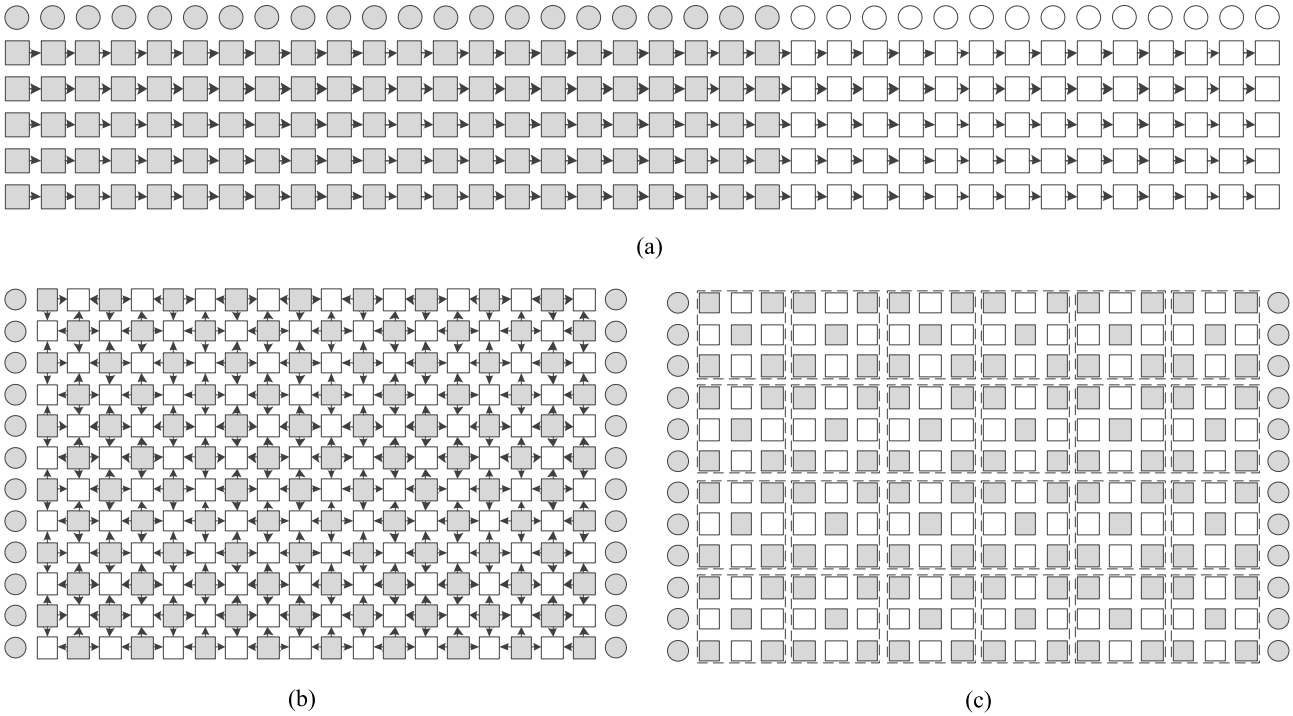


FIGURE 9. Membrane structure and initial placements for (a) Π_{Embry}, (b) Π_{Yang}, and (c) Π_{Szasz}. Membranes are denoted by squares with a grey color for initial placement and white for substitution, and I/O blocks are denoted by circles.

TABLE 2. Comparison of initial configurations between four computing systems.

	Π_{Embry}	Π_{Yang}	Π_{Szasz}	Γ
n_t	180	216	216	120
n_{init}	110	108	128	120
n_{sub}	80	108	88	14
n_{fr}	14	108	88	14
$wire$	5057	5947	5902	4074

5) *wire* – total wire length calculated by the semi-perimeter model.

System Π_{Embry}, Π_{Yang}, and Π_{Szasz} are all based on unidirectional sliding rules. Since moving direction is unidirectional, there is only one single path for object transport and the movement of one object is generally restricted. In some cases, these systems may lose effectiveness, even though there are many remaining empty membranes. Meanwhile, all the membranes are classified by unidirectional sliding rules into two groups: one is on the starting point of sliding rules and used for storing objects in the initial placement; another is on the terminal point and for substituting the fault. Thus, there are a number of empty membranes in the initial placement, leading to large resource consumption.

On the contrary, since two membranes are equal in bidirectional sliding rules, the system Γ can support a compact initial placement with fewer membranes. One object can move to any other membrane in a connected structure and many feasible paths are provided for object transport. So fault

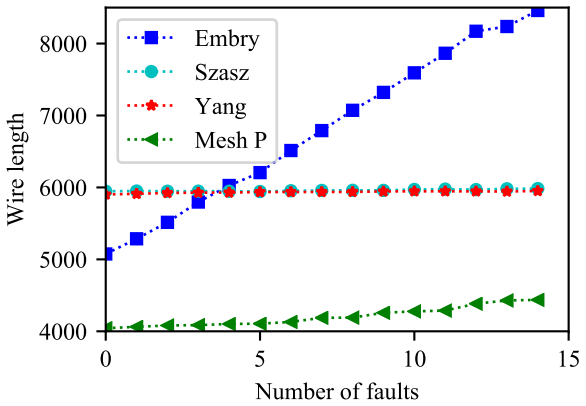


FIGURE 10. Comparison of the wire length between four computing systems (Π_{Embry}, Π_{Yang}, Π_{Szasz}, and Γ).

substitution can be achieved by non-adjacent membranes. The system will, ideally, keep going until all the empty membranes are used up. As a result, the system Γ can reduce the resource consumption and improve design flexibility.

In dynamic placement, 14 randomly selected faults are inserted into different systems. The total wire length is calculated when each fault is recovered. As is shown in Fig.10, the wire length curve of Π_{Embry} has a low starting point, indicating that Π_{Embry} has a compact initial placement. However, the curve shows a rapid increasing trend, leading to poor circuit performances in several self-repairing processes. The wire length curves of Π_{Yang} and Π_{Szasz} are approximately horizontal, representing stable performances in a long term.

Nevertheless, the starting points are high, so these systems have loose initial placements and their resource consumption is large. Compared with these three curves, the wire length curve of Γ has the lowest starting point and increases slightly in the following process. It indicates that Γ can reduce resource consumption and maintain good performances in the long term.

V. CONCLUSION

The purpose of this paper was to study dynamic placement optimization for bio-inspired self-repairing hardware. The investigations concentrated on computing systems and distributed optimization algorithms. A computing model based on P systems was built to simulate dynamic placement of different platforms. The computational power was analyzed to provide necessary conditions for placement optimization. An improved computing system, called a mesh P system with bidirectional sliding rules, was developed for dynamic placement optimization. Its computational power has been verified by a constructive proof. Moreover, an optimization model based on placement distance was developed and discussed. An membrane optimization algorithm based on hill climbing was proposed for a reduction of the calculation amount. Experimental results showed that the proposed approach can reduce resource consumption and maintain good performances.

One concern needing further research is dynamic routing optimization. Also we will continue to improve optimization algorithms and design hardware structure to meet engineering demands.

REFERENCES

[1] Q. Zhuo, Y. Qian, Y. Li, and N. Wang, "Development of configurations for lookup table–based embryonics using graphic mapping: A case study," *Adv. Mech. Eng.*, vol. 7, no. 7, Jun. 2015, Art. no. 168781401559253.

[2] Q. Zheng, J. Cui, W. Lu, H. Guo, J. Liu, X. Yu, Y. Wei, L. Wang, J. Liu, C. He, and Q. Guo, "The increased single-event upset sensitivity of 65-nm DICE SRAM induced by total ionizing dose," *IEEE Trans. Nucl. Sci.*, vol. 65, no. 8, pp. 1920–1927, Aug. 2018.

[3] S. Abba and J.-A. Lee, "Bio-inspired self-aware fault-tolerant routing protocol for network-on-chip architectures using particle swarm optimization," *Microprocessors Microsyst.*, vol. 51, pp. 18–38, Jun. 2017.

[4] I. Yang, S. H. Jung, and K.-H. Cho, "Self-repairing digital system based on state attractor convergence inspired by the recovery process of a living cell," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 2, pp. 648–659, Feb. 2017.

[5] T. Wang, J. Cai, and Y. Meng, "A novel embryonics electronic cell array structure based on functional decomposition and circular removal self-repair mechanism," *Adv. Mech. Eng.*, vol. 9, no. 9, Sep. 2017, Art. no. 168781401772008.

[6] D. Mange, E. Sanchez, A. Stauffer, G. Tempesti, P. Marchal, and C. Piguet, "Embryonics: A new methodology for designing field-programmable gate arrays with self-repair and self-replicating properties," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 6, no. 3, pp. 387–399, Sep. 1998.

[7] Z. Qingqi, Q. Yanling, L. Yue, W. Nantian, and L. Tingpeng, "Embryonic electronics: State of the art and future perspective," in *Proc. IEEE 11th Int. Conf. Electron. Meas. Instrum.*, Aug. 2013, pp. 140–146.

[8] H. Liang, S. Sinha, and W. Zhang, "Parallelizing hardware tasks on multicontext FPGA with efficient placement and scheduling algorithms," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 37, no. 2, pp. 350–363, Feb. 2018.

[9] A. Kamaleldin, A. Mohamed, A. Nagy, Y. Gamal, A. Shalash, Y. Ismail, and H. Mostafa, "Design guidelines for the high-speed dynamic partial reconfiguration based software defined radio implementations on Xilinx Zynq FPGA," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2017, pp. 1–4.

[10] G. Paun and G. Rozenberg, "A guide to membrane computing," *Theor. Comput. Sci.*, vol. 287, no. 1, pp. 73–100, Sep. 2002.

[11] Y. Thoma, G. Tempesti, E. Sanchez, and J.-M.-M. Arostegui, "POEtic: An electronic tissue for bio-inspired cellular applications," *Biosystems*, vol. 76, nos. 1–3, pp. 191–200, Aug. 2004.

[12] A. Stauffer and J. Rossier, "Self-testable and self-repairable bio-inspired configurable circuits," in *Proc. NASA/ESA Conf. Adapt. Hardw. Syst.*, Jul. 2009, pp. 155–162.

[13] P. Lala and B. Kumar, "Human immune system inspired architecture for self-healing digital systems," in *Proc. Int. Symp. Quality Electron. Design*, Jun. 2003, pp. 292–297.

[14] I. Yang, S. H. Jung, and K.-H. Cho, "Self-repairing digital system with unified recovery process inspired by endocrine cellular communication," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 6, pp. 1027–1040, Jun. 2013.

[15] C. Szasz and A. Cioloca, "Two-layer coarse-fine-grid network model for bio-inspired computing systems development," in *Proc. 17th Int. Conf. Syst. Theory, Control Comput. (ICSTCC)*, Oct. 2013, pp. 515–520.

[16] J. Soto, J. Manuel Moreno, and J. Cabestany, "A self-adaptive hardware architecture with fault tolerance capabilities," *Neurocomputing*, vol. 121, pp. 25–31, Dec. 2013.

[17] M. Samie, G. Dragffy, A. Popescu, T. Pipe, and J. Kiely, "Prokaryotic bio-inspired system," in *Proc. NASA/ESA Conf. Adapt. Hardw. Syst.*, Jul. 2009, pp. 171–178.

[18] S. Kim, H. Chu, I. Yang, S. Hong, S. H. Jung, and K.-H. Cho, "A hierarchical self-repairing architecture for fast fault recovery of digital systems inspired from paralogous gene regulatory circuits," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 12, pp. 2315–2328, Dec. 2012.

[19] K. G. Subramanian, R. Saravanan, M. Geethalakshmr, P. H. Chandra, and M. Margenstern, "P systems with array objects and array rewriting rules," *Prog. Natural Sci. Mater. Int.*, vol. 17, no. 4, pp. 479–485, Apr. 2007.

[20] B. Song, C. Zhang, and L. Pan, "Tissue-like P systems with evolutional symport/antiport rules," *Inf. Sci.*, vol. 378, pp. 177–193, Feb. 2017.

[21] R. Diestel, "Graph theory," *Math. Gazette*, vol. 173, no. 502, pp. 67–128, 2011.

[22] F. Gavril, "Algorithms for a maximum clique and a maximum independent set of a circle graph," *Networks*, vol. 3, no. 3, pp. 261–273, 1973.

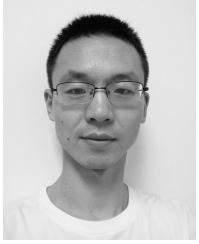
[23] A. Archer, "The 15 puzzle: How it drove the world crazy," *Math. Intelligencer*, vol. 29, no. 2, pp. 83–85, Mar. 2007.

[24] N. Chatterjee, S. Paul, and S. Chattopadhyay, "Task mapping and scheduling for network-on-chip based multi-core platform with transient faults," *J. Syst. Archit.*, vol. 83, pp. 34–56, Feb. 2018.

[25] W. Housseyni, O. Mosbahi, M. Khalgui, Z. Li, L. Yin, and M. Chetto, "Multiagent architecture for distributed adaptive scheduling of reconfigurable real-time tasks with energy harvesting constraints," *IEEE Access*, vol. 6, pp. 2068–2084, 2018.

[26] M. J. Alexander, J. P. Cohoon, J. L. Ganley, and G. Robins, "Placement and routing for performance-oriented FPGA layout," *VLSI Des.*, vol. 7, no. 1, pp. 97–110, Jan. 1998.

[27] A. R. Yıldız, "An effective hybrid immune-hill climbing optimization approach for solving design and manufacturing optimization problems in industry," *J. Mater. Process. Technol.*, vol. 209, no. 6, pp. 2773–2780, Mar. 2009.



LIU XIUBIN was born in Hegang, China, in 1990. He received the B.S. and M.S. degrees from the National University of Defense Technology (NUDT), Changsha, China, in 2013 and 2015, respectively, where he is currently pursuing the Ph.D. degree with the Laboratory of Science and Technology on Integrated Logistics Support. His research interests include health management and self-repairing hardware.

QIAN YANLING received the B.S., M.S., and Ph.D. degrees from the National University of Defense Technology (NUDT), Changsha, China, in 1995, 1998, and 2002, respectively. He is currently a Full Professor with the Laboratory of Science and Technology on Integrated Logistics Support, NUDT. His research interests include integrated logistics support, data-driven prognostics and health management, and system reliability for equipment.

FENG XIANGLI was born in Erdos, China, in 1995. He received the B.S. degree from the College of Mechanical and Electrical Engineering, Central South University, Changsha, China, in 2017. He is currently pursuing the M.S. degree with the Laboratory of Science and Technology on Integrated Logistics Support, National University of Defense Technology. His research interest includes self-repairing hardware.

ZHUO QINGQI was born in Xiamen, China, in 1985. He received the B.S., M.Sc., and Ph.D. degrees from the National University of Defense Technology (NUDT), Changsha, China, in 2008, 2010, and 2015, respectively. His research interest includes failure analysis and health management for equipment.

LI YUE received the M.S. and Ph.D. degrees from the National University of Defense Technology (NUDT), in 1993 and 2007, respectively. He is currently a Full Professor with NUDT. His research interests include condition monitoring and fault diagnosis, bio-inspired self-repairing technology, and reliability testing and evaluating.