A Hybrid Mode Membrane Computing Based Algorithm with Applications for Proton-Exchange Membrane Fuel Cells

Jinhui Zhao [1,*], Wei Zhang [2], Tianyu Hu [3], Ouguan Xu [4], Shengxiang Yang [5] and Qichun Zhang [6]

[1] Zhejiang-Belarus Joint Laboratory of Intelligent Equipment and System for Water Conservancy and Hydropower Safety Monitoring, College of Electrical Engineering, Zhejiang University of Water Resources and Electric Power, Hangzhou 310018, China
[2] Fair Friend Institute of Intelligent Manufacturing, Hangzhou Vocational & Technical College, Hangzhou 310018, China; zhw618@hzvtc.edu.cn
[3] College of Mechanical and Electrical Engineering, China Jiliang University, Hangzhou 310018, China; s1801081103@cjlu.edu.cn
[4] College of Information Engineering, Zhejiang University of Water Resources and Electric Power, Hangzhou 310018, China; xuog@zjweu.edu.cn
[5] School of Computer Science and Informatics, De Montfort University, Leicester LE1 9BH, UK; syang@dmu.ac.uk
[6] Department of Computer Science, University of Bradford, Bradford BD7 1DP, UK; q.zhang17@bradford.ac.uk
* Correspondence: jhzhao2009@zju.edu.cn

Abstract: Membrane computing is a branch of natural computing, which has been extended to solve various optimization problems. A hybrid mode membrane-computing-based algorithm (HMMCA) is proposed in this paper to solve complex unconstrained optimization problems with continuous variables. The algorithmic framework of HMMCA translates from its distributed cell-like membrane structure and communication rule. A non-deterministic evolutionary programming method and two computational rules are applied to enhance the computational performance. In a numerical simulation, 12 benchmark test functions with different variables are used to verify the algorithmic performance. The test results and comparison with three other algorithms illustrate its effectiveness and superiority. Moreover, a case study on a proton exchange membrane fuel cell (PEMFC) system parameter optimization problem is applied to validate its practicability. The results of the simulation and comparison with seven other algorithms demonstrate its practicability.

Keywords: membrane computing; nondeterministic evolutionary programming method; benchmark functions; proton exchange membrane fuel cell

MSC: 68T20

Citation: Zhao, J.; Zhang, W.; Hu, T.; Xu, O.; Yang, S.; Zhang, Q. A Hybrid-Mode Membrane-Computing-Based Algorithm with Applications for Proton-Exchange Membrane Fuel Cells. *Mathematics* **2023**, *11*, 3054. https://doi.org/10.3390/math11143054

Academic Editor(s): Gaige Wang

Received: 28 May 2023
Revised: 26 June 2023
Accepted: 28 June 2023
Published: 10 July 2023

1. Introduction

As a branch of natural computing, membrane computing (also called P systems) was presented by Paun in 2000. It is a class of parallel and distributed computing models [1,2]. The computing models of membrane computing (MC) are abstracted from the structures and functions of living cells, the organization and cooperation in tissues and neural nets, etc. The membrane structure, rules and objects are three essential ingredients of MC (P systems). In a P system, the objects (multi-sets described in mathematics) are processed in a nondeterministic, maximally parallel manner, according to their given local reaction rules in distributed membranes, or compartmentalized by the membranes. Then, transitions of objects operated by rules, or passed between different system configurations of membranes, can be obtained [2,3]. At present, the three main types of P systems are cell-like, tissue-like and neural-like [4]. The research into MC has revealed that, whether cell-like, tissue-like or neural-like, whether with symbol objects or string objects, whether working in the generative or accepting modes, or whether with a certain combination of

features, all of these classes of systems are universal [2–4], and have reached the computational power of a Turing machine [3]. For example, it has been proved that spiking neural P systems are Turing complete [4], and the computational power of cell-like P systems exploiting some type of "time acceleration" is theoretically capable of trespassing the Turing barrier [5,6]. The theory of MC has been applied to various fields, such as modeling in medicine and biology [7], modeling of metapopulations [8], linguistics in computer science [9], segmentation in computer vision [10], membrane controllers [11], etc.

Inspired by the powerful computational power of MC, more and more optimization algorithms have been studied, based on the theory of using MC for solving complicated optimization problems with a single objective or multiple objectives, constrained or unconstrained, and continuous or discrete variables. Distinguished from other biologically inspired algorithms, membrane-computing-based algorithms are not only analogous to the natural processes in living cells, but also imitate the ideas, parallel-distributed computing models and nondeterministic computational rules of MC. The first "cell-like" membrane algorithms (MAs) were proposed by Nishida to solve the traveling salesman problem [12], which dynamically evolved membranes by creating or destroying them according to certain possibilities, a genetic algorithm and a simulated annealing algorithm, distributed in nested membranes as sub-algorithms. Another "cell-like" algorithm inspired by MC adopted a static nested membrane structure, and three rules were taken from genetic algorithms and inspired by MC [13]. A quantum-inspired evolutionary algorithm based on P systems with a static "cell-like" membrane structure was proposed for solving knapsack problems [14]. The bio-inspired algorithm based on membrane computing (BIAMC), and hybrid method based on membrane computing (HBS), were proposed for solving unconstrained and constrained problems with continuous variables [15,16]. In BIAMC, a static cell-like nested membrane structure and three new rules that simulated the Golgi apparatus of eukaryotic cells were applied. In HBS, an SQP rule cooperated with an improved BIAMC in a variable manner to solve complicated constrained problems [16]. A novel P-system-based optimization algorithm (BIPOA) was proposed for solving parameter estimation problems, where a nested membrane structure with three new rules was applied [17]. The micro-charge field effect P systems optimization algorithm (MFE-POA), as a deeper exploration of the standard P systems optimization algorithm (POA), was proposed to solve the model parameter estimation of two types of PV cell models and multi-cell PV modules [18]. Multi-objective optimization algorithms, based on tissue P systems for optimal control of complex systems, have also been investigated [19,20]. An algorithm, called the membrane computing with harmony search algorithm (MC-HSA), in which an active membrane dissolving P system was designed and the harmony search algorithm was embedded in the P system, was proposed to quickly select a subset of potential disease-causing genes [21]. A semantic and intelligent focused crawler based on the semantic vector space model (SVSM), and the membrane computing optimization algorithm (MCOA), were proposed for their semantic understanding and intelligent learning abilities. The MCOA method was used to optimize four weighted factors based on evolution and communication rules [22]. An optimization numerical spiking neural P system (ONSN P systems or ONSNPS) was proposed to solve continuous constrained optimization problems, in which a guider algorithm is introduced to finish individuals' crossover and selection, and a random mechanism was used for production function selection to achieve updated parameters [23].

The reported results show that these P systems' optimization algorithms are readily available and are superior to their compared algorithms, in terms of global searching, speed and accuracy. However, novel computational rules derived from MC or living cells are inadequate, and their computational rules from other algorithms carry their own properties, such as rules from GA or DE, etc. Compared with other MC-based algorithms for solving problems with continuous variables, BIAMC works much more effectively, and is distinctive in its algorithmic framework and computational rules, such as 100-dimensional optimization problems. A hybrid mode membrane-computing-based algorithm

(HMMCA) is presented to enhance the computational efficiency of BIAMC when the complexity of problems is increased, such as with high dimensions and highly nonlinear and inseparable variables.

The remainder of the paper is organized as follows. HMMCAs are described in Section 2. Section 3 reports the numerical experiment. A case study on optimizing fuel cell model parameters is presented in section 4. Section 5 concludes this paper.

2. Hybrid Mode Membrane Computing Based Algorithm

2.1. Problem Formulation

Research on non-determinism versions of MC computing models was expanded several years ago [24]. Inspired by the research and its result, a non-determinism evolutionary programming method (NEP), a method of nondeterministic computational probability of rules, a nondeterministic abstraction rule and a modified crossover mode of rewriting rule are proposed and applied to enhance the computational performance of the bio-inspired algorithm based on MC.

2.2. Membrane Structure and Operational Rules

The membrane structure and communication rule, non-determinism evolutionary programming and two modified computational rules of HMMCA are given as follows. The membrane structure is shown in Figure 1.

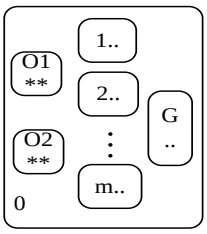

Figure 1. Membrane structure of HMMCA. Membrane 0 is the surface membrane. O_1 and O_2 are two element membranes, where membrane O_1 keeps initial objects generated in order by the surface membrane and membrane O_2 dynamically keeps evolved objects in order by each parallel element membrane. '*****' denote objects. The series 1, …, m is the serial numbers of parallel element membranes, where several rules are located, such as the rewriting rule, pairing rule, selection rule. '·' denotes objects. G denotes the quasi-Golgi element membrane in which the target indication rule, transition rule, non-deterministic abstraction rule and selection rule are located.

The surface membrane randomly generates initial objects and a communication object, and contains all inner membranes and outputs results. The computational rules are distributed in parallel element membranes, and the evolved objects are finally sent out or deleted from the quasi-Golgi or parallel element membrane due to the termination candition. Each object comprises a candidate solution vector of problems and other parameters calculated, e.g., an objective function value, function evaluation times, etc.

Without loss of generality, minimization is consideration in this paper. The new rules used in HMMCA are summarized as follows.

1. Communication rule: Objects and communication objects are controlled by this rule to transfer in inner membranes. n_o numbered objects kept in membrane O_1 or O_2 are sent into parallel element membranes by this rule. It sends out n_o rewritten objects to membrane O_2 or are deleted. Moreover, it sends a communication object built in a parallel element membrane into the quasi-Golgi according to the activating condition or the next parallel element membrane, or from the quasi-Golgi to a parallel element membrane.

2. Crossover mode of rewriting rule:

$$\boldsymbol{X}_i, \boldsymbol{X}_k \underset{p_{cm}}{\mapsto} \boldsymbol{X}_i, \boldsymbol{X}_k \tag{1}$$

Step 1:

$$\boldsymbol{X}_i, \boldsymbol{X}_k \to \boldsymbol{X}_i, \boldsymbol{X}_k$$

$$where, \begin{cases} \boldsymbol{X}_i, \boldsymbol{X}_k \to \boldsymbol{X}_i{}', \boldsymbol{X}_k{}' \\ \boldsymbol{X}_i{}', \boldsymbol{X}_i, \boldsymbol{X}_k{}', \boldsymbol{X}_k \to \boldsymbol{X}_i, \boldsymbol{X}_k \\ \boldsymbol{X}_i{}' = \eta \times \boldsymbol{X}_i + (1-\eta) \times \boldsymbol{X}_k, \\ if\ f(\boldsymbol{X}_i{}') \le f(\boldsymbol{X}_i)\ \text{ then }\ \boldsymbol{X}_i = \boldsymbol{X}_i{}' \\ else \\ if\ f(\boldsymbol{X}_i{}') \le f(\boldsymbol{X}_{i+1})/f(\boldsymbol{X}_{i-1})\ \text{ then} \\ \quad \boldsymbol{X}_{i+1}/\boldsymbol{X}_{i-1} = \boldsymbol{X}_i{}' \\ if\ f(\boldsymbol{X}_i{}') \le f(\boldsymbol{X}_k)\ \text{ then }\ \boldsymbol{X}_k = \boldsymbol{X}_i{}' \\ \boldsymbol{X}_k{}' = (1-\eta) \times \boldsymbol{X}_i + \eta \times \boldsymbol{X}_k \\ if\ f(\boldsymbol{X}_k{}') \le f(\boldsymbol{X}_i)\ \text{ then }\ \boldsymbol{X}_i = \boldsymbol{X}_k{}' \\ else \\ if\ f(\boldsymbol{X}_k{}') \le f(\boldsymbol{X}_{i+1})/f(\boldsymbol{X}_{i-1})\ \text{ then} \\ \quad \boldsymbol{X}_{i+1}/\boldsymbol{X}_{i-1} = \boldsymbol{X}_k{}' \\ if\ f(\boldsymbol{X}_k{}') \le f(\boldsymbol{X}_k)\ \text{ then }\ \boldsymbol{X}_k = \boldsymbol{X}_i{}' \\ \eta \in [0,1];\ i = 1, \dots, n_{co}; k = n_{co}+1, \dots, n_o \end{cases}$$

Step 2:

$$\boldsymbol{X}_i, \boldsymbol{X}_{i+1} \to \boldsymbol{X}_i, \boldsymbol{X}_{i+1}$$

$$where, \begin{cases} \boldsymbol{X}_i, \boldsymbol{X}_{i+1} \to \boldsymbol{X}_i{}', \boldsymbol{X}_{i+1}{}' \\ \boldsymbol{X}_i{}', \boldsymbol{X}_i, \boldsymbol{X}_{i+1}{}', \boldsymbol{X}_{i+1} \to \boldsymbol{X}_i, \boldsymbol{X}_{i+1} \\ \boldsymbol{X}_i{}' = \eta \times \boldsymbol{X}_i + (1-\eta) \times \boldsymbol{X}_{i+1}, \\ \boldsymbol{X}_{i+1}{}' = (1-\eta) \times \boldsymbol{X}_i + \eta \times \boldsymbol{X}_{i+1} \\ \eta \in [0,1];\ i = 1, \cdots, n_o - 1; \end{cases}$$

p_1, p_i and η are random numbers between 0 and 1. p_{cm} is the probability of crossover mode. i and k are objects indices. $\mathbf{X}_i$ denotes the solution vector in object i, and $\mathbf{X}_i'$ denotes its rewritten vector. n_o and n_{co} are the number of objects in a parallel membrane and a communication object.

When the crossover mode operates, the weighted arithmetic mean calculation performs between the top n_{co} and other n_o-n_{co} objects. Firstly, generate $n_{co} \times (n_o - n_{co})$ uniformly distributed variables p_1. If and only if p_1 is less than the probability of crossover mode p_{cm}, object $\mathbf{X}_i$ and $\mathbf{X}_k$ are replicated and rewritten as described in step 1. When rewritten ones $\mathbf{X}_i'$ or $\mathbf{X}_k'$ are less than the initials or the other in top n_{co}, the rewritten object replaces it. Otherwise, $\mathbf{X}_i'$ and $\mathbf{X}_k'$ are deleted. After the first step, sequentially, n_o rewritten objects are rewritten by the arithmetical crossover between every two neighboring copies of the objects. Similarly, generate n_o-1 uniformly distributed variables p_i. When p_i is less than p_{cm}, two neighboring copies of the object $\mathbf{X}_i$ and $\mathbf{X}_{i+1}$ are rewritten. When the rewritten $\mathbf{X}_i'$ or $\mathbf{X}_{i+1}'$ are less than the initials, the rewritten object replaces the initial. Or else, they are deleted.

3. Nondeterministic abstraction rule (Algorithm 1).

$$\mathbf{X_1}, \mathbf{X_2} \underset{p_{na}}{\longmapsto} \mathbf{X'}, \mathbf{X_2} \tag{2}$$

The main steps are summarized as follows.

Algorithm 1 Nondeterministic abstraction rule

```
Initialize p
for j = 1: l
if p(j) < p_na
   for k = 1: l
        x = X(1,1: l);
        if k = l
             a = 0;
        Else
             a = max (1, r×l/2) ;
             if k + a >= l | | j + a >= l
                  a = min(l – k, l – j);
             end(if)
        end(if)
        c = X(2, k:k + a);
        x(j: j + a) = c;
        if f(x) < f(X(1))
             X(1, updates);
        end(if)
      end (for)
  end(if)
end (for)
```

p_{na} is the probability of this computational rule. p is a random vector between 0 and 1. j and k are the variables indexes. l is the dimension of a solution vector. llk is a random number between 1 and $l/2$. r is a random number between 0 and 1. $\lfloor\ \rfloor$ is the sign of floor rounded numbers.

2.3. Algorithm Description

Define 1. *One algorithmic cycle consists of applying the computational rules from parallel membrane 1 to parallel membrane m [16].*

The main steps of procedure are summarized as follows (Algorithm 2).

Algorithm 2 Hybrid mode membrane computing

```
Initialize g, nn, tiv, pop1, cpop
mm = mm + 1
pop2 = pop1
for k = 1:C
   for n = 1:N_m
      while (FET≤ fet)
        {
        g = g + 1
        Initialize p_cm, p_mm, p_na, p_t.
        if n =nn(k)
          E = pop2((N_m−1)*N_o + 1:n*N_o,:);
        else
          E = pop1((N_m−1)*N_o + 1:n*N_o,:);
        end(if)
        Crossover mode works on E
        Mutation mode and pairing rule works on updated E
        pop2 updates
        cpop updates
        if g > 2
             if (~mod(mm,2))
               chg = (wide*0.0618/(n*10^(k−1)))*ones(N_co,l)
               target indication rule works on cpop
```

```
25                    cpop updates
26                    transport rule works on cpop
27                    cpop updates
28                    nondeterministic abstraction rule works cpop updates
29                  end(if)
30              end(if)
31              tiv updates;
32            }
33     end (for)
34  end (for)
```

N_r, N_m, N_o, N_{co}, C, mm, FET(*fet*), g, nn, **tiv**, **pop1**, **pop2** and **cpop** denote the amount of repeated trials run once, the number of parallel element membranes, the number of objects sending in each parallel membrane, the sum of objects in a communication object, the algorithmic cycles, an activation parameter for the quasi-Golgi membrane, function evaluation times (the setting value used as a termination condition), the current number of $N_m \times C$, the index of parallel element membranes applied nondeterministic evolutionary program, the matrix used for keeping directions of target indication vectors, the initial candidate objects matrix, the updated candidate objects matrix and the communication object matrix, respectively. p_{cm}, p_{mm}, p_{na} and p_t are the probabilities of crossover mode, mutation mode, non-deterministic abstraction, and transition rules. In HMMCA, p_{cm}, p_{mm}, p_{na} and p_t are assigned randomly in each parallel membrane and each algorithmic cycle.

The non-deterministic evolutionary programming method is actualized by the membrane O_1 and O_2, and communication rule. For each algorithmic cycle, an element membrane is preset randomly to apply evolutionary programming, i.e., besides the communication object, the random preset element membrane receives objects from membrane O_2. Conversely, the remainder element membranes receive objects from membrane O_1. Then, this nondeterministic evolutionary programming is fulfilled after the computing of all algorithmic cycles finishes.

2.4. Computational Complexity

The computational cost of an algorithm is generally to evaluate the CPU time or the times of functions called maximally. For optimization algorithms, the objective functions' evaluation times (FET) are used. The computational cost of HMMCA under the given parameter settings is the summation of each computational rules' cost, which is direct proportion to dimensions of decision variables, probabilities of computational rules, the numbers of algorithmic cycles, parallel element membranes and objects. The computational cost of rewriting rule consumed in the mutation mode is FET $\leq C \times N_m \times N_o \times p_{mm} \times l$, and that of crossover mode is FET $\leq C \times N_m \times p_{cm} \times N_o \times (N_{co} + 1)$. In the quasi-Golgi membrane, the computational cost of non-deterministic abstraction rule is in direct proportion to the square of dimensions of problems and the activating condition of quasi-Golgi membrane, i.e., FET $\leq 0.5C \times N_m \times p_{na} \times l^2$. The rules of target direction and transition are all in direct proportion to the number of objects in communication objects, FET $\leq 1.5C \times N_m \times N_{co}$. Therefore, the worst-case time complexity of HMMCA is $O(l^2)$.

3. Performance Evaluation

3.1. Benchmark Function Optimization

To analyze the merits and drawbacks of the HMMCA, we performed a numerical experiment on 12 benchmark test functions with 30 and 100 variables. The names, variable ranges, parameters and optimal solutions of functions are listed in Table 1. Meanwhile, BIAMC, the evolution strategy with covariance matrix adaptation (CMA-ES) and the genetic algorithm (GAs) were executed for comparison and analyses. CMA-ES, proposed by Nikolaus Hansen and Andreas OsterMeier in 2001, is a powerful evolutionary algorithm using the evolutionary strategy and adaptive covariance matrix method, and has been

used in hundreds of non-linear, non-convex continuous, etc. optimization problems and applications [25].

Table 1. Benchmark functions.

Function	Optimum Solution
f_1: Griewangk	$x_i \in [-600, 600]$, $f_1^*(0, \ldots, 0) = 0$
f_2: Rosenbrock	$x_i \in [-100, 100]$, $f_2^*(1, \ldots, 1) = 0$
f_3: Ackley	$x_i \in [-32.768, 32.768]$, $c_1 = 20$, $c_2 = 0.2$, $c_3 = 2\pi$, $f_3^*(0, \ldots, 0) = 0$
f_4: Weierstrass	$x_i \in [-0.5, 0.5]$, $a = 0.5$, $b = 3$, $k_{max} = 20$, $f_4^*(0, \ldots, 0) = 0$.
f_5: Rastrigin	$x_i \in [-5.12, 512]$, $f_5^*(0, \ldots, 0) = 0$.
f_6: Sphere	$x_i \in [-100, 100]$, $f_6^*(0, \ldots, 0) = 0$.
f_7: Schwefel 2.26	$x_i \in [-500, 500]$, $f_7^*(420.968, \ldots, 420.9687) = 0$.
f_8: Schwefel 1.12	$x_i \in [-100, 100]$, $f_8^*(0, \ldots, 0) = 0$.
f_9: Step	$x_i \in [-100, 100]$, $f_9^*(0, \ldots, 0) = 0$.
f_{10}: Schwefel 2.22	$x_i \in [-100, 100]$, $f_{10}^*(0, \ldots, 0) = 0$
f_{11}: noncontinuous rastrigin	$x_i \in [-5.12, 512]$, $f_{11}^*(0, \ldots, 0) = 0$.
f_{12}: quatic	$x_i \in [-1.28, 1.28]$, $f_{12}^*(0, \ldots, 0) = 0$.

"*" means the global optimal solution.

3.2. Experiment Setup

The parameter settings of the HMMCA derived from the previous research and trial and method where N_r = 100; N_m = 10; C = 30/ 15/15; the stop condition, FET = 30,000/ 200,000/500,000; N_o = 7; N_{co} = 2; $p_{cm} \geq 0.8$; $p_{mm} \geq 0.8$; $p_t \leq 0.3$; $p_{na} \geq 0.8$; the increment used in mutation mode, $(r \times wide_j)/(ml \times 10^c)$, where $wide_j = x_j^u - x_j^l$, x_j^u and x_j^l are the upper limit and lower limit, respectively, ml and c are the current numbers of the parallel element membrane and algorithmic cycle, the magnitude of target indication vector, $0.0618wide_j/(ml \times 10^{c-1})$; the activating condition of quasi-Golgi, the multiplication of the current value of algorithmic cycle by that of the parallel element membrane is exactly divisible by 2. To implement nondeterministic computing and keep a better computational performance, the upper or lower limits of p_{cm}, p_{mm}, p_t and p_{na} are given, that is, in each element membrane they are randomly set within the upper or lower limits.

Most of the parameter settings of BIAMC followed previous research on BIAMC combined with trial and error: N_r = 100; N_m = 10; C = 30/15/15; FET = 30,000/ 200,000/500,000; N_o = 7; N_{co} = 2; p_{cm} = 1; p_{mm} = 1; p_t = 0.3; p_a = 1; the increment used in mutation mode was $(r \times wide_j)/(ml \times 10^c)$; the magnitude of target indication vector, $0.0618wide_j/(ml \times 10^{c-1})$; the activating condition of quasi-Golgi was that the multiplication of the current value of algorithmic cycle by that of parallel element membrane was exactly divisible by 2.

The MATLAB code of CMA-ES was downloaded from reference [25]. Most of the parameters of CMA-ES were set to default as described in the CMA-ES algorithm. The changed parameters were derived from the trial and error method: the amount of repeated trials, N_r = 100; sigma = $(x^u - x^l)/3$; Options: MaxFunEvals = 30,000/200,000/ 500,000; MaxIter = Inf; TolHistFun = 1/Inf; TolUpX = Inf; TolFun = 1/Inf; TolX = 1/Inf; Restarts = 2; StopFunEvas = 30,000/200,000/500,000; StopOnWarnings = no; StopOnStagnation = off; StopOnEqualFunctionValues = Inf; StopFitness = −Inf.

The ga function in the MATLAB optimization toolbox was applied because of its popularization. In the same way, the parameters of ga were set: the amount of repeated trials, N_r = 100, 'PopulationSize'= 50, 'MutationFcn'= mutation-adaptfeasible, 'CrossoverFraction'=0.9, 'MigrationFraction'=0.2, 'SelectionFcn'=selectionstochunif, 'Cross-overFcn'= crossoverheuristic/crossoverintermediate (used for f_1, f_4, f_9, f_{11}), 'Generations'= 600/5000/10,000, 'StallGenLimit'= 600/4000/10,000, 'Display'= off. Other optional parameters were default.

All the experiments in this paper ran on the same PC with Win 7 64-bit (Inter Core™ i7-2600 processor with 3.40 GHz and 8 GB RAM). The listed data were obtained by executing each algorithm once when MATLAB 2009b was open or when the test functions were changed.

3.3. Result and Discussion

For a compact representation, most test result of functions with 30 and 100 variables were listed in Table 2. In Table 2, A denotes algorithms; mean refers to averages of 100 groups of results; st.dev refers to the standard deviation of 100 groups of result; avg.t. denotes the averages of CPU time consumed in 100 repeated trials; sr is the success ratio that the times of results approaching optimum value (i.e., $|f - f^*| \leq 1 \times 10^{-8}$) to 100; t refers to the result of a *t*-test between HMMCA and the other algorithms. Before, the optional parameters of selection and crossover in GAs and the setting of Restart in CMA-ES were fixed by trial and error while considering sr and st.dev.

Table 2. Result of functions f_1–f_{12} with 30 dimensions calculated by HMMCA, BIAMC, CMA-ES, GAs.

F	A	30					100				
		Mean	st.dev	avg.t	sr	t	Mean	st.dev	avg.t	sr	t
f1	H	4.30×10^{-67}	1.63×10^{-66}	15.83	1	=	2.87×10^{-21}	7.35×10^{-21}	37.99	1	=
	B	1.73×10^{-3}	8.68×10^{-3}	19.57	0.95	≈	1.30×10^{-3}	1.30×10^{-2}	26.19	0.99	≈
	C	1.28×10^{-3}	3.16×10^{-3}	47.22	0.85	+	4.93×10^{-4}	2.24×10^{-3}	223.48	0.95	+
	G	3.76×10^{-1}	3.54×10^{-1}	21.35	0	+	7.24×10^{1}	1.17×10^{1}	85.92	0	+
f2	H	4.99×10^{0}	2.98×10^{0}	18.87	0.79	=	1.26×10^{1}	3.28×10^{1}	37.86	0.85	=
	B	4.10×10^{0}	1.07×10^{1}	17.30	0.07	+	1.88×10^{1}	5.11×10^{1}	33.32	0.33	+
	C	5.98×10^{-1}	1.43×10^{0}	52.17	0.84	−	1.54×10^{0}	2.30×10^{0}	236.22	0.1	+
	G	1.79×10^{2}	3.83×10^{2}	19.01	0	+	5.50×10^{2}	1.07×10^{3}	79.69	0	+
f3	H	6.57×10^{-15}	3.31×10^{-15}	15.23	1	=	2.98×10^{-11}	6.98×10^{-11}	30.34	1	=
	B	7.15×10^{-2}	5.03×10^{-1}	19.24	0.98	≈	1.90×10^{-9}	2.84×10^{-9}	19.95	1	+
	C	4.76×10^{-1}	1.38×10^{0}	19.93	0.89	+	4.20×10^{0}	4.62×10^{0}	205.05	0.19	+
	G	4.38×10^{-1}	1.08×10^{0}	19.19	0	+	1.11×10^{1}	1.78×10^{0}	75.96	0	+
f4	H	7.11×10^{-16}	5.85×10^{-15}	134.34	1	=	1.45×10^{-9}	3.96×10^{-9}	1014.80	0.94	=
	B	2.85×10^{-6}	7.35×10^{-6}	139.17	0.43	+	1.93×10^{-6}	4.96×10^{-6}	1055.84	0.46	+
	C	1.34×10^{0}	2.03×10^{0}	133.00	0.65	+	2.49×10^{1}	4.44×10^{0}	1371.89	0	+
	G	6.08×10^{0}	2.02×10^{0}	140.49	0	+	5.29×10^{1}	4.95×10^{0}	1347.69	0	+
f5	H	2.99×10^{-1}	2.98×10^{0}	18.87	0.99	=	0	0.00×10^{0}	38.84	1	=
	B	0.00×10^{0}	0.00×10^{0}	15.98	1	≈	0	0.00×10^{0}	33.77	1	=
	C	5.93×10^{1}	2.67×10^{1}	46.32	0	+	4.43×10^{2}	6.16×10^{1}	222.14	0	+
	G	9.95×10^{-3}	9.95×10^{-2}	18.84	0.99	≈	6.99×10^{0}	3.59×10^{0}	70.35	0	+
f6	H	9.26×10^{-65}	4.15×10^{-64}	18.35	1	=	3.57×10^{-19}	1.08×10^{-18}	31.68	1	=
	B	2.26×10^{-58}	5.09×10^{-58}	15.34	1	+	5.84×10^{-16}	1.28×10^{-15}	27.55	1	+
	C	2.72×10^{-110}	1.7×10^{-109}	47.27	1	−	1.01×10^{-82}	8.34×10^{-82}	203.93	1	−
	G	1.31×10^{-18}	2.16×10^{-18}	18.87	1	+	4.52×10^{-15}	2.60×10^{-15}	63.97	1	+
f7	H	7.11×10^{1}	5.00×10^{2}	16.34	0.98	=	9.55×10^{-11}	2.59×10^{-12}	34.25	1	=
	B	-3.22×10^{-12}	8.12×10^{-13}	20.69	1	≈	9.76×10^{-11}	1.23×10^{-11}	24.64	1	≈
	C	5.55×10^{3}	7.02×10^{2}	21.33	0	+	1.92×10^{4}	1.40×10^{3}	177.56	0	+
	G	1.82×10^{-14}	4.08×10^{-13}	19.44	1	≈	3.01×10^{-11}	1.04×10^{-11}	65.84	1	+
f8	H	3.30×10^{-63}	1.43×10^{-62}	28.90	1	=	2.41×10^{-15}	2.15×10^{-14}	122.62	1	=
	B	2.77×10^{3}	1.20×10^{3}	27.60	0	+	4.72×10^{4}	1.47×10^{4}	138.54	0	+
	C	4.71×10^{-79}	2.24×10^{-78}	75.20	1	−	6.20×10^{-296}	0	421.73	1	−
	G	2.33×10^{1}	2.29×10^{1}	29.96	0	+	1.39×10^{4}	6.77×10^{3}	160.57	0	+
f9	H	0	0	18.39	1	=	0	0.00×10^{0}	31.91	1	=
	B	0	0	15.56	1	=	0	0	30.24	1	=
	C	0	0	3.07	1	=	0	0	11.84	1	=
	G	1.48×10^{3}	4.77×10^{2}	19.21	0	+	1.11×10^{4}	1.78×10^{3}	73.85	0	+
f10	H	1.04×10^{-32}	2.22×10^{-32}	18.54	1	=	2.01×10^{-9}	3.76×10^{-9}	32.24	0.92	=
	B	3.32×10^{-29}	5.95×10^{-29}	15.70	1	+	1.28×10^{-7}	2.08×10^{-7}	27.19	0.47	+
	C	4.12×10^{-71}	4.12×10^{-70}	51.31	1	−	1.69×10^{-3}	1.11×10^{-2}	212.15	0.07	+
	G	1.76×10^{1}	3.92×10^{1}	18.94	0	+	1.17×10^{3}	6.97×10^{2}	63.12	0	+
f11	H	3.00×10^{-1}	3.00×10^{0}	18.18	0.99	=	2.73×10^{-14}	1.04×10^{-13}	46.32	1	=
	B	1.80×10^{0}	7.16×10^{0}	13.88	0.94	≈	5.00×10^{-14}	2.47×10^{-13}	35.30	1	≈
	C	2.02×10^{1}	4.71×10^{0}	19.64	0	+	1.24×10^{2}	2.58×10^{1}	124.60	0	+

	G	2.97×10^{1}	6.91×10^{1}	29.70	0	+	2.39×10^{2}	5.27×10^{1}	97.94	0 +
	H	2.90×10^{-4}	2.48×10^{-4}	15.76	0	=	4.86×10^{-4}	2.09×10^{-4}	36.87	0 =
f_{12}	B	9.22×10^{-4}	8.14×10^{-4}	16.87	0	+	4.11×10^{-3}	1.88×10^{-3}	25.16	0 +
	C	2.10×10^{-1}	8.17×10^{-2}	48.57	0	+	4.47×10^{-1}	9.03×10^{-2}	224.94	0 +
	G	1.66×10^{-1}	5.94×10^{-2}	26.17	0	+	5.00×10^{-1}	9.54×10^{-2}	72.63	0 +

Compared the data in Table 2, the computational performance of HMMCA is not satisfactory in terms of robustness because not all sr are 1; nor is that the case for other algorithms. However, HMMCA keeps its high computational performance of speediness, accuracy and robustness in solving 30- and 100-dimensional functions compared its values of sr, avg.t and st.dev. HMMCA is mostly superior to BIAMC in robustness and accuracy considering sr, avg.t, st.dev., and t test result. avg.t of HMMCA is higher than that of BIAMC, although they shared the same stop condition. It was partly caused by much more location in memory occupied and more time spent in the nondeterministic computation for high dimensional problems. CMA-ES is outstanding at approaching optimal solutions, due to its covariance matrix adaptation with evolution strategy, considering its test results for f_2, f_6, f_8, f_9, f_{10} with 30D and 100D, although this computational performance is not so robust. The avg.t of CMA-ES increased more compared with other algorithms. Except for f_6 and f_7, GAs can not approach to optimal solutions, and the dimensions of problems caused it more serious trouble than the other algorithms.

To exhibit the behavior of four algorithms in solving problems with different dimensions, the mean values of f_1, f_2, f_3 and f_{10} with 10 variables together with 30D and 100D are plotted in Figure 2.

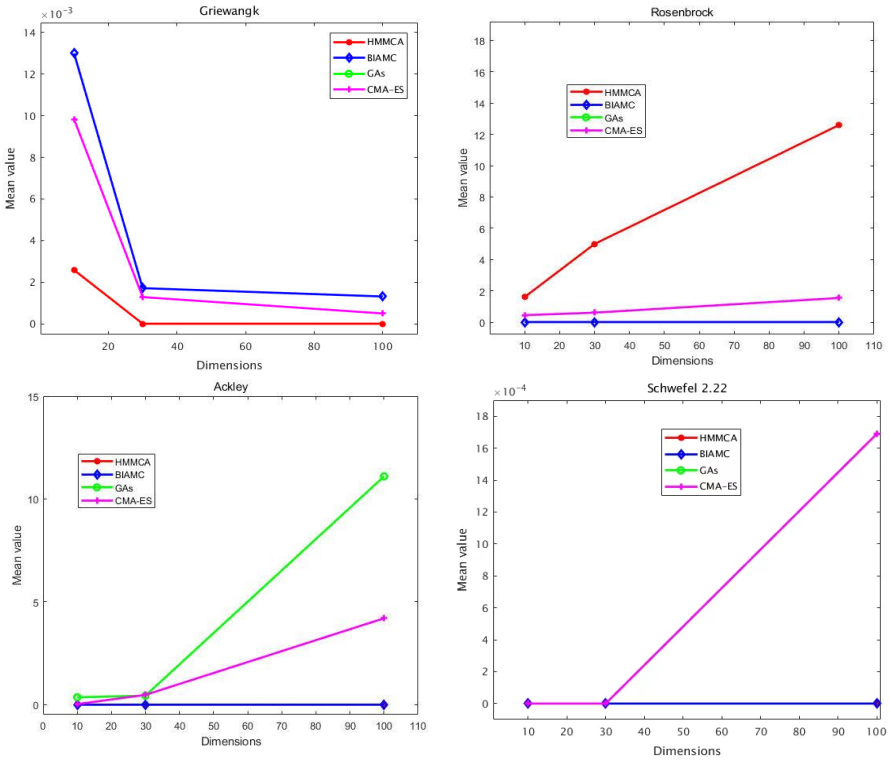

Figure 2. Mean values of f_1, f_2, f_3 and f_{10} from four algorithms.

It is shown from the four sub-graphs in Figure 2 that the mean values tested on the same functions with different dimensions deviate from the optimum on the different dimension. The change of mean curves calculated by four test algorithms reveals their dissimilar computational performance, such as in the sub-graph of f_1 and f_2. Then, the global searching performance of HMMCA is superior to the others, according to the data in Figure 2. Meanwhile, the computational performance of HMMCA is the most robust and highest efficiency considering the result of sr, avg.t, t-test, etc. in Table 2.

4. Industrial Applications

A fuel cell is an electrochemical device that converts the fuel directly and continuously to electricity and heat by using an oxidant (typically oxygen in air), and its by-products are water, very small amounts of nitrogen dioxide and other emissions (depending on the fuel source) [26–28]. Because of quick start-up, low temperature and high power density, etc., Polymer Electrolyte Membrane fuel cell (PEMFC) is widely researched and applied in power distribution systems, e.g., transportation and industries [17,26–28].

Fuel cells generally have three adjacent segments, i.e., the anode, the cathode and the electrolyte that allows charges to move between two poles (shown in Figure 3). Proton-conducting membrane covers two electrodes where two chemical reactions occur (described in Equations (3)–(5)). Only protons (H^+) can pass through the membrane, which are transported from the hydrogen electrode to the oxygen electrode.

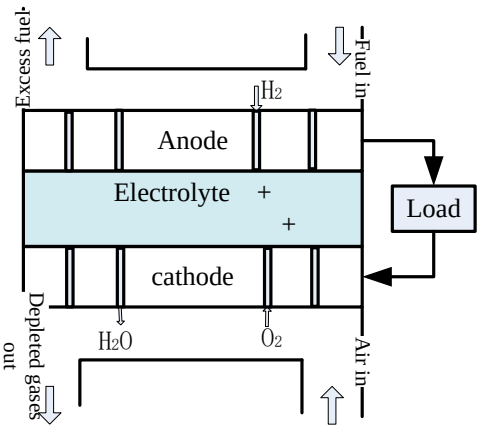

Figure 3. Scheme of a PEMFC.

The electrochemical reactions occurring in PEMFC are as follows [28]:

Anode Reaction:

$$2H_2 \rightarrow 4H^+ + 4e^- \quad (3)$$

Cathode Reaction:

$$4H^+ + 4e^- + O_2 \rightarrow 2H_2O \quad (4)$$

Overall cell electrochemical reaction:

$$2H_2 + O_2 \rightarrow 2H_2O \quad (5)$$

The electricity produced by unit fuel cells is about 0.7 volts [26], so cells can be placed in series or parallel circuits to increase the voltage and current output to supply hundreds of kilowatts designed. Moreover, the performance of fuel cells can be improved by pressurizing the air to higher pressures. These characters have been reflected in the following mathematical model of PEMFC.

4.1. A. PEMFC Stack Model

In this case study, a widely used semi-empirical model of PEMFC for predicting *i-v* character is applied as a parameter optimization problem [17,28,29]. The voltage produced by PEMFC can not attain its theoretically maximum E_{Nernst} (thermodynamic cell potential), which is defined via Nernst equation in expanded form [17,27–29], and resulted from some resistances cause. Three main loss terms in this model: The activation overvoltage V_{act} is caused by activating the anode and the cathode; ohmic loss V_{ohmic} is the resistance to electrons transferred through the collecting plates and carbon electrodes, and protons transferred through the solid membrane; concentration overvoltage V_{con} is caused by the drop in the concentration of oxygen and hydrogen. The fuel cell electrochemical model [17,27–33] is given as follows.

The stack voltage V_{stack} connected by n unit cell:

$$V_{\text{stack}} = nV_{\text{FC}} \tag{6}$$

The output voltage of unit fuel cell V_{FC}

$$V_{\text{FC}} = E_{\text{Nernst}} - V_{\text{act}} - V_{\text{ohmic}} - V_{\text{con}} \tag{7}$$

The thermodynamic cell potential of fuel cell E_{Nernst}:

$$\begin{aligned} E_{\text{Nernst}} &= 1.229 - 8.5\times10^{-4}(T - 298.15) + 4.3085\times10^{-5} \\ &\quad T(\ln(P_{\text{H}_2}) + 0.5\ln(P_{\text{O}_2})) \end{aligned} \tag{8}$$

Three voltage loss terms:

$$V_{\text{act}} = -[\zeta_1 + \zeta_2 T + \zeta_3 T\ln(C_{\text{O}_2}) + \zeta_4 T\ln(i)] \tag{9}$$

$$V_{\text{ohmic}} = i(R_{\text{M}} + R_{\text{C}}) \tag{10}$$

$$V_{\text{con}} = -b\ln(1 - \frac{J}{J_{\max}}) \tag{11}$$

$$\begin{aligned} R_{\text{M}} &= \frac{\rho_{\text{M}} l}{A} = \frac{l}{A}\times \\ &\quad \frac{181.6\left[1 + 0.03(\frac{i}{A}) + 0.062(\frac{T}{303})^2(\frac{i}{A})^{2.5}\right]}{[\lambda - 0.634 - 3(\frac{i}{A})]\exp\left[4.18(\frac{T-303}{T})\right]} \end{aligned} \tag{12}$$

$$\begin{aligned} P_{\text{H}_2} &= 0.5RH_a \times P_{\text{O}_2} \times \\ &\quad \left[\left(\exp(\frac{1.635(i/A)}{T^{1.334}})\times\frac{RH_a \times P_{\text{H}_2\text{O}}}{P_a}\right)^{-1} - 1\right] \end{aligned} \tag{13}$$

$$\begin{aligned} \log_{10}(P_{\text{H}_2\text{O}}) &= 2.95\times10^{-2}(T - 273.15) - 9.18\times10^{-5} \\ &\quad (T - 273.15)^2 + 1.44\times10^{-7}(T - 273.15)^3 - 2.18 \end{aligned} \tag{14}$$

$$P_{\text{O}_2} = P_c - P\text{H}_c \times P_{\text{H}_2\text{O}} - P_{\text{N}_2}\times\exp(\frac{0.291(i/A)}{T^{0.832}}) \tag{15}$$

$$P_{\text{N}_2} = \frac{0.79}{0.21}P_{\text{O}_2} \tag{16}$$

$$C_{\text{O}_2} = \frac{P_{\text{O}_2}}{5.08\times10^6\exp(\frac{-498}{T})} \tag{17}$$

where T is the absolute temperature (K). P_{H_2} and P_{O_2} refer to the partial pressure (atm) of hydrogen and oxygen. ζ_1, ζ_2, ζ_3, ζ_4, R_{C} (Ω) and b (V) are six parameters of cell model. R_{C} is the resistance to transfer of protons through the membranes, and usually considered as constant. R_{M} and R_{C} denote the equivalent membrane resistance inside and the equivalent contact resistance to the outside (Ω·cm^2). b depends on the cell and its operation state. C_{O_2} is the concentration of oxygen in the catalytic interface of the cathode (mol·cm^{-3}). i is the cell operating current (A). J and J_{max} are the actual current density and the maximum current density, $J = \frac{i}{A}$. l is the thickness of the PEM (cm). A is the cell active area (cm^2).

ρ_M is the membrane's specific resistivity (Ω·cm). λ is an adjustable parameter. P_{N_2} and P_{H_2O} mean nitrogen partial pressure at the cathode gas flow channel and the saturation pressure of water vapor. P_a and P_c are the anode and cathode gas flow channel (atm), respectively. RH_a and RH_c are the relative humidity of vapor in the anode and cathode, respectively.

The parameters' definition of semi-empirical mathematical model is essential for approaching to the actual performance of PEMFC. Then, minimizing errors between the output of model and the actual performance is used as the objective function of this parameter optimization problem.

$$\min_{\mathbf{X}} f(\mathbf{X}) = \sum_{j=1}^{N} (V_{stackm} - V_{stack}(\mathbf{X}))^2 \quad (18)$$

where $\mathbf{X}$ is the parameter vector. V_{stackm} is the experimental data of output voltage. V_{stack} is the model output voltage. N is the number of the experimental data.

4.2. Experimental Data

A set of experimental data was adopted, where four polarization curves for a 250 W fuel cell were recorded in different process conditions accordingly which were measured at (3/5 bar, 353.15 K), (1/1 bar, 343.15 K), (2.5/3 bar, 343.15 K) and (1.5/1.5 bar, 343.15 K) [17,28,29]. The experimental data used in this case study were taken from four polarization curves, altogether 15 *i-v* pairs in each group. Other parameters data and the operation conditions were shown in Table 3. Two group of data measured at (3/5 bar, 353.15 K), (1/1 bar, 343.15 K) were used for estimation, and the left two group of data were used for validation. The upper and lower bounds of parameters were given in Table 4.

Table 3. Stack parameters and the range of operation conditions [17,28,29].

Stack Parameters		Range of Operation Conditions
Number of cells in series	n	24
Inlet Anode pressure	Pa (bar)	3.0–1.0
Cell's effective active area	A (cm)	27
Inlet Cathode pressure	P_c (bar)	5.0–1.0
Nafion 115:5	mil (μm)	127
Stack temperature	T (K)	353.15–343.15
Maximum current density	I_{max} (A cm^{-2})	0.86
Relative humidity in anode	RHa	1
Rated power	(W)	250
Relative humidity in cathode	RHc	1

Table 4. Value range of parameters [17,32,33].

Parametes	ζ_1	ζ_2	ζ_3	ζ_4	R_C	B	λ
Upper bound	−0.80	6.00×10^{-3}	1.00×10^{-4}	-8.00×10^{-5}	9.90×10^{-4}	0.60	24
Lower bound	−1.20	8.00×10^{-4}	3.50×10^{-5}	-3.00×10^{-4}	8.00×10^{-5}	0.01	10

4.3. Optimization and Simulation Results

We applied HMMCA, BIAMC, GAs and CMA-ES to solve this parameter optimization problem.

The parameter settings of the HMMCA were: the amount of repeated trials, $N_r = 100$; the number of parallel membranes, $N_m = 10$; the number of algorithm cycle for one trial, $C = 35$; the stop condition, FET = 30,000; the number of objects sending in each parallel membrane, $N_o = 7$; the sum of objects in a communication object, $N_{co} = 2$; $p_{cm} \geq 0.8$; $p_{mm} \geq 0.8$; $p_t \leq 0.3$; $p_{na} \geq 0.8$; the increment used in mutation mode, $(r \times wide_j)/(ml \times c)$. The magnitude of target indication vector, $wide_j \times 0.618/(ml \times c)$. The activating condition of the quasi-Golgi

was that the multiplication of the current value of algorithmic cycle by that of membranes does divide exactly by 2.

The parameter settings of BIAMC were: N_r = 100; N_m = 10; C = 35; FET = 30,000; N_o = 7; N_{co} = 2; p_{mm} = 1; p_{cm} = 1; p_t = 0.3; p_a = 1. The increment used in mutation mode, $(r \times wide_j)/(ml \times c)$. The magnitude of target indication vector, $wide_j \times 0.618/(ml \times c)$. The activating condition of quasi-Golgi was that the multiplication of the current value of algorithmic cycle by that of membranes was exactly divisible exactly by 2. Most of parameter settings were the same as the subsection of function examination, except the parameters of mutation mode and target indication rule were changed for a faster searching.

The main optional parameter settings of ga were: the amount of repeated trials, N_r = 100, 'PopulationSize'= 50, 'MutationFcn'= mutationadaptfeasible, 'CrossoverFraction'= 0.9, 'MigrationFraction'= 0.2, 'SelectionFcn'= selectionstochunif, 'CrossoverFcn'= crossoverheuristic/crossoverintermediate, 'Generations'= 600, 'StallGen-Limit'= 600, 'Display'= off. Other optional parameters were default.

The changed parameter settings were: the amount of repeated trials, N_r = 100; sigma = min($(x^u - x^l)/3$); Options: MaxFunEvals = 30,000; MaxIter = Inf; TolHistFun= 1/Inf; TolUpX = Inf; TolFun = 1/Inf; TolX = 1/Inf; Restarts = 2; StopFunEvas = 30,000; StopOnWarnings = no; StopOnStagnation = off; StopOnEqualFunctionValues = Inf; StopFitness = -Inf; etc.

All the result in Table 5 were obtained by running them once when MATLAB 2009b opened.

Table 5. The results calculated by HMMCA, BIAMC, GA and CMA-ES.

A	Min	Mean	max	avg.t(s)	st.dev
HMMCA	5.4881	5.8091	6.2466	6.28	1.61×10^{-1}
BIAMC	5.5105	5.8171	6.3642	6.36	1.76×10^{-1}
CMA-ES	5.4639	5.4639	5.4640	15.85	1.00×10^{-5}
GA	5.6624	6.2618	8.3809	9.12	4.31×10^{-1}

Meanwhile, the parameters calculated by four algorithms in [17,28,29] were introduced directly and their square deviations on the estimation data were used for comparison. In 100 groups of parameters calculated by each algorithm, the group of parameters minimized the square deviation on the estimation data (SD1) were list in Table 6. The evaluation of computational cost used in the four algorithms in[17,28,29] was different from this paper. According to the data shown in those references, even if the objective function was evaluated once for a candidate solution in one generation, the minimum FET was 20,000.

Table 6. Comparison with results from [17,28,29].

A	ζ_1	$\zeta_2 \times 10^{-3}$	$\zeta_3 \times 10^{-5}$	$\zeta_4 \times 10^{-4}$	$R_C \times 10^{-4}$	b	λ	SD1
HMMCA	−0.9113	2.8731	7.4531	−1.1482	1.9944	0.0301	12.0815	5.4881
BIAMC	−1.1257	3.2957	6.1573	−1.1207	1.6779	0.0273	11.2850	5.5105
GA	−0.9415	3.0181	7.5145	−1.1196	6.7769	0.0346	13.9963	5.6624
CMA-ES	−1.2000	3.4962	6.0781	−1.1618	0.8000	0.0299	11.9888	5.4639
BIPOA	−0.8016	2.6673	8.1288	−1.2713	0.8000	0.0324	13.5158	5.6053
RGA	−1.1568	3.4243	6.4161	−1.1544	1.4504	0.0343	12.8989	5.5676
HGA	−0.944957	3.01801	7.4010	−1.8800	1.0000	0.02914489	23.0000	11.3400
SGA	−0.94731	3.0641	7.7134	−1.9390	2.7197	0.023981	19.7650	13.1686

It is shown that CMA-ES is superior to HMMCA at solving this seven-dimensional parameter optimization problem considering the explored best parameters with the smallest SD1. However, HMMCA is consistently excellent compared with other performance, such as avg.t, st.dev, mean and the minimal value of objective function listed in Table 5. It can be seen that all the best parameters in Table 6 are close neighbors of the parameters calculated by the CMA-ES algorithm. Perhaps it will be helpful to explore the best parameters to apply other termination conditions or parameters setting for HMMCA and BIAMC.

5. Conclusions

To goal of precisely solving as many types of complicated optimization problems as possible with lowest computational complexity is pursued by all optimization algorithms. Distributed structures, nondeterministic and evolutionary mechanism are generality used for most optimization algorithms. The hybrid mode membrane computing algorithm (HMMCA) was proposed in this paper as a modified algorithm of the bio-inspired algorithm based on membrane computing (BIAMC), which contained a new netted membrane structure together with a communication rule, a nondeterministic computation mechanism of evolutionary programming used for the communication rule, rewriting rule, target indication rule and abstraction rule. The test results on benchmark functions with 30D and 100D, and a seven-dimensional parameter optimization problem of PEMFC revealed its higher efficiency, global searching and robustness of computational performance in general, compared with the test results of BIAMC, CMA-ES and GAs.

Author Contributions: Conceptualization, J.Z. and W.Z.; methodology, J.Z.; software, W.Z.; validation, T.H., S.Y. and Q.Z.; formal analysis, O.X. and S.Y.; investigation, W.Z.; resources, J.Z.; data curation, J.Z.; writing—original draft preparation, J.Z.; writing—review and editing, T.H. and S.Y.; visualization, Q.Z.; supervision, S.Y.; project administration, J.Z.; funding acquisition, O.X. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Joint Funds of the Zhejiang Provincial Natural Science Foundation of China under Grant No. LZJWZ23E090001; Zhejiang Provincial Natural Science Foundation under Grant No. LGG21E050005; Hangzhou Key Scientific Research Program of China under Grant No. 20212013B06

Data Availability Statement: The data are available upon request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Paun, G. Computing with Membranes. *J. Comput. Syst. Sci.* **2000**, *1*, 108-143.
2. Paun, G.; Perez-Jimenez, M. J. Membrane Computing: Brief Introduction, Recent Result and Applications. *Biosystems* **2006**, 1, 11–22.
3. Ciobanu, G.; Paun, G.; Perez-Jimenez, J. M. *Applications of Membrane Computing*; Springer: New York, NY, USA, 2006; pp. 33.
4. Ibarra, O.H.; Paun, G. Membrane Computing: A General View. *Ann. Eur. Acad. Sci.* **200**8, *1*, 83-101.
5. Ionescu, M.; Paun, G.; Yokomori, T. Spiking Neural P Systems. *Fund. Inform.* **2006**, *2-3*, 279- 308.
6. Calude, S.C.; Paun, G. Bio-Steps Beyind Turing. *Biosystems* **2004**, *1-3*, 175-194.
7. Franco, G.; Jonoska, N.; Osborn, B.; Plaas, A. Knee Joint Injury and Repair Modeled by Membrane Systems. *BioSystems* **2008**, *3*, 473-488.
8. Besozzi, D.; Cazzaniga, P.; Pescini, D.; Mauri, G. Modelling Metapopulations with Stochastic Membrane Systems. *Biosystems* **2008**, *3*, 499-514.
9. Csuhaj-Varju, E.; Paun, G.; Vaszil, G. Grammar Systems Versus Membrane Computing: the Case of CD Grammar Systems. *Fund. Inform.* **2007**, *3*, 271-292.
10. Christinal, H.A.; Diaz-Pernil, D.; Real, P. Region-Based Segmentation of 2D and 3D Images with Tissue-Like P Systems. *Pattern Recogn. Lett.* **2011**, *16*, 2206-2212.
11. Buiu, C.; Vasile, C.; Arsene, O. Development of Membrane Controllers for Mobile Robots. *Inform. Sciences* **2012**, *187*, 33-51.
12. Nishida, T.Y. An Application of P System: a New Algorithm for NP-Complete Optimization Problems. in Proceedings of 8th world multi-conference on systems, cybernetics and informations, Orlando, Florida, USA, 2004, pp.109-112.
13. Huang, L.; Wang, N. An Optimization Algorithm Inspired by Membrane Computing. *Lecture Notes in Computer Science* **2007**, *4222*, 49-52.
14. Zhang, G.X.; Gheorghe, M.; Wu, C.Z. A Quantum-Inspired Evolutionary Algorithm Based on P Systems for Knapsack Problem. *Fund. Inform.* **2008**, *1*, 93-116.
15. Zhao, J.; Wang, N. A Bio-Inspired Algorithm Based on Membrane Computing and its Application to Gasoline Blending Scheduling. *Comput. Chem. Eng.* **2011**, *2*, 272-283.
16. Zhao, J.; Wang, N. A Hybrid Optimization Method Based on Membrane Computing. *Ind. Eng. Chem. Res.* **2011**, *3*, 1691-1704.
17. Yang, S.; Wang, N. A Novel P Systems Based Optimization Algorithm for Parameter Estimation of Proton Exchange Membrane Fuel Cell. *Int. J. Hydrgen Energ.* **2012**, *37*, 8465-8476.
18. Yang, S.; Max, N.; Xie S.; Li L.; Zhao, T. Photovoltaic Cell Model Parameter Optimization Using Micro-Charge Field Effect P Systems. *Eng. Appl. Artif. Intel.* **2021**, *104*, 104374.
19. Huang, L.; He, X.X.; Wang, N.; Xie, Y. P System Based Multi-Objective Optimization Algorithm. *Prog. Nat. Sci.* **2007**, *4*, 458-465.
20. Huang, L.; Suh, II H.; Abraham, A. Dynamic Multi-Objective Optimization Based on Membrane Computing for Control of Time-Varying Unstable Plants. *Inform. Sci*. **2011**, *11*, 2370-2391.
21. Tuo, S.; Liu F.; Feng Z.; Li C.; Zhu, Y.; Chen T.; Liu H. Membrane Computing with Harmony Search Algorithm for Gene Selection from Expression and Methylation Data. *J. Membr. Comput.* **2022**, *4*, 293-313.
22. Liu, W.; Gan Z.; Xi T.; Du Y.; Wu J.; He Y.; Jiang P.; ·Liu X.; Lai X. A Semantic and Intelligent Focused Crawler based on Semantic Vector Space Model and Membrane Computing Optimization Algorithm. *Appl. Intell.* **2023**, *53*, 7390–7407.
23. Dong, J.; Zhang, G.; Luo, B.; Rong, H. An Optimization Numerical Spiking Neural P System for Solving Constrained Optimization Problems. *Inform. Sci*. **2023**, *626*, 428-456.
24. Ibarra, O.H. On Deterministic Versus Nondeterminism in P Systems. *Comput. Syst. Sci.* **2006**, *344*, 120-133.
25. CMA-ES in MATLAB. Available online: http://yarpiz.com/235/ypea108-cma-es(accessed on 25 May 2023).
26. Fuel Cell. Available online: http://en.wikipedia.org/wiki/Fuel_cell (accessed on 25 May 2023).
27. Ramakumar, R. Fuel Cell-an Introduction. in Proceeding of IEEE-PES summer meeting, July, 2001, pp. 702-709.
28. Mo, Z. J.; Zhu, X. J.; Wei, L. Y.; Cao, G. Y. Parameter Optimization for a PEMFC Model with a Hybrid Genetic Algorithm. *Int. J. Energy Res.* **2006**, *30*, 585-597.
29. Ohenoja, M.; Leiviska, K. Identification of Electrochemical Model Parameters in PEM Fuel Cells. in Proceeding of Power Engineering, Energy and Electrical Drives International Conference on, March, 2009, pp. 363-368.

30. Ohenoja, M.; Leiviska, K. Validation of Genetic Algorithm Results in a Fuel Cell Model. *Int. J. Hydrgen Energ.* **2010**, *35*, 12618-12625.
31. Mann, R.F.; Amphlett, J.C.; Hooper, M.A.I.; Jensen, H.M.; Peppley,B.A.; Roberge, P.R. Development and Application of a Generalized Steady-State Electrochemical Model for a PEM Fuel Cell. *J. Power Sources* **2000**, *86*, 172-180.
32. Correa, J.M.; Farret, F.A.; Popov, V.A.; Simoes, M.G. Sensitivity Analysis of the Modeling Parameters Used in Simulation of Proton Exchange Membrane Fuel Cell. *IEEE T. Energ. Conver.* **2005**, *20*, 211-218.
33. Askarzadeh, A.; Rezazadeh, A. A Grouping-Based Global Harmony Search Algorithm for Modeling of Proton Exchange Membrane Fuel Cell. *Int. J. Hydrogen Energ.* **2011**, *36*, 5047-5053.