A Human-Computer Cooperative Particle Swarm Optimisation Based Immune Algorithm for Layout Design

Fengqiang Zhao[a,b], Guangqiang Li[a,*], Chao Yang[a], Ajith Abraham[c,d], Hongbo Liu[a,c,*]

[a] *School of Information Science and Technology, Dalian Maritime University, Dalian 116026, China*

[b] *College of Electromechanical and Information Engineering, Dalian Nationalities University, Dalian 116600, China*

[c] *Machine Intelligence Research Labs (MIR Labs), Scientific Network for Innovation and Research Excellence, Auburn 98071, WA, USA*

[d] *IT4Innovations - Center of excellence, VSB - Technical University of Ostrava, Ostrava 70833, Czech Republic*

Abstract

Packing and layout problems have wide applications in engineering practice. However, they belong to NP (non-deterministic polynomial)-Complete problems. In this paper, we introduce human intelligence into the computational intelligent algorithms, namely particle swarm optimisation (PSO) and immune algorithms (IA). A novel human-computer cooperative PSO-based immune algorithm (HCPSO-IA) is proposed, in which the initial population consists of the initial artificial individuals supplied by human and the initial algorithm individuals are generated by a chaotic strategy. Some new artificial individuals are introduced to replace the inferior individuals of the population. HCPSO-IA benefits by giving free rein to the talents of designers and computers and contributes to solving complex layout design problems. The experimental results illustrate that the proposed algorithm is feasible and effective.

*Corresponding author.

Email addresses: fqzhao2002@yahoo.com.cn (Fengqiang Zhao), gqlimail@163.com (Guangqiang Li), yangchaoneu@sina.com (Chao Yang), ajith.abraham@ieee.org (Ajith Abraham), lhb@dlut.edu.cn (Hongbo Liu)

Keywords: Human-computer Cooperation, Genetic Algorithms, Immune Function, Particle Swarm Optimization, Hybrid, Layout

1. Introduction

Packing and layout problems [1, 2] deals with how to put objects into a limited space reasonably under given constraints. These constraints include the requirements for equilibrium, stability, connectivity and adjacent states. Some methods [1, 3] are presented, such as mathematical programming and criterion methods, heuristic algorithms, graph theory, expert systems, swarm intelligence and natural laws but it is still difficult to solve the problems satisfactorily.

Unlike those traditional methods, the swarm intelligent algorithms illustrated more superior performances in recent years [4, 5, 6, 7]. They are particularly fit for solving medium or large-scale problems [8] but for the complex packing and layout problems, there exist some defects, such as premature convergence and slow convergence rate. In this paper, we introduce some novel strategies into our hybrid PSO-based immune algorithm (PSO-IA). In addition, a intelligent machine or an algorithm is powerful in numerical calculation and repetitive operations but lack of experience and inspiration, which are just the strong points of human beings. There are very differences in nature between computer and human, which also mean that they should deeply depend on each other when solving practical complex problems. The idea of man-machine synergy (or called as human-computer cooperation) was originated by Lenat and Feigenbaum [9]. It is regarded as a promising approach to solve complex engineering problems [10, 11]. According to this idea, taking into account the intractable nature of the packing and layout problems and their importance, we further propose a novel human-computer cooperative PSO-based immune algorithm (HCPSO-IA).

2. Hybrid PSO-based Immune Algorithm

In this Section, we present a hybrid algorithm PSO-IA with some improvement strategies, which include immunity principle, chaotic initialization, new PSO update operators, arithmetic-progression rank-based selection with pressure as well as a multi-subpopulation evolution based on PGA [12].

2.1. Arithmetic-progression rank-based selection with pressure

In genetic algorithms, rank-based selection model focuses on the numerical size relations rather than the specific numerical differences among individual fitness values. A probability assignment table has to be preset. But there is no deterministic rule for design of the table. It is difficult to make the selection probabilities of individuals adaptively changed along with evolution process [13, 14]. We introduce arithmetic-progression rank-based selection with pressure based on the mathematical concept of interpolation method.

There is one independent parameter in this operator, selection pressure α. It denotes the ratio of the maximal individual selection probability P_{max} to the minimal one P_{min} within a generation, i.e. $P_{max} = \alpha P_{min}$. It numerically shows the superiority that the better individuals are reproduced into the next generation during selection operation and it is changeable along with algorithm evolution. In the early stage, lesser α can maintain population diversity and prevent the algorithm from premature convergence, while in the late stage, greater α can benefit accelerating convergence. Let $\alpha = f(K)$, K denotes the generation number. Assume that α_{max} and α_{min} denote the maximum and minimum of selection pressure respectively, then

$$\alpha = \frac{(K-1)(\alpha_{max} - \alpha_{min})}{K_{max} - 1} + \alpha_{min} \tag{1}$$

where K_{max} is the maximal generation number set in algorithm. And our numerical experiments and statistical analyses show that α_{max} and α_{min} may be chosen in the interval [6, 15] and [1.5, 5] respectively [15].

To calculate the selection probability of every individual, we arrange all the individuals within a population in descending order based on their fitness values. Let Ind_i represent the ith individual within a population as well as F_i and P_i represent its fitness and selection probability respectively. There exist Ind_i $(i = 1, 2, \cdots, M)$ and $F_i > F_{i+1}$ $(i = 1, 2, , \cdots, M-1)$. M is the population size. Suppose that the selection probability values of all the individuals form an arithmetic progression. Its first term and last term are $P_1 = P_{max} = \alpha P_{min}$ and $P_M = P_{min}$ respectively. Obviously, the sum of all the individual selection probability is 1, i.e. subtotal of arithmetic progression as follows:

$$\begin{aligned} S_M &= [(P_1 + P_M) \cdot M]/2 \\ &= [(\alpha P_M + P_M) \cdot M]/2 \\ &= 1 \end{aligned} \tag{2}$$

We get

$$P_M = 2/[M(1+\alpha)] \tag{3}$$

Therefore we obtain the common difference of the arithmetic progression

$$\begin{aligned} \Delta &= (P_1 - P_M)/(M-1) \\ &= [P_M(\alpha - 1)]/(M-1) \\ &= [2(\alpha-1)]/[M(1+\alpha)(M-1)] \end{aligned} \tag{4}$$

And there exists

$$\begin{aligned} P_i &= P_1 - (i-1)\Delta \\ &= \alpha P_M - (i-1)\Delta \end{aligned} \tag{5}$$

Substituting Equs. (3) and (4) into Equ. (5), it is easy to find that

$$P_i = \frac{2\alpha \cdot (M-i) + 2(i-1)}{M \cdot (\alpha+1)(M-1)} \quad i = 1, 2, \cdots, M \tag{6}$$

In the process of selection, we firstly reproduce the best individual of current generation and have its copy in the next generation directly based on the elitist model, and then figure out selection probabilities of all individuals according to Equ. (6) and finally generate the remaining $M-1$ individuals of the next generation by fitness proportional model. Compared with traditional rank-based selection, the advantage of proposed selection operator is that it can conveniently change the selection probabilities of individuals by changing selection pressure.

2.2. Antibody concentration and immune selection

Immunity-based algorithms that originated in 1990's have many good characteristics [16]. They can embody immune memory, extraction and inoculating efficient antibodies as well as antibody inhibition and promotion mechanism in the biological immune systems. So the evolutionary algorithms based on immunity [17, 18] can effectively prevent premature, accelerate convergence rate. In this section, we introduce immune principle into parallel genetic algorithms and put forward some improvements as follows:

- We adopt the simple and easy Euclidean distance to calculate affinities between antibodies (i.e. individuals) for convenient to engineering design.

- We present correction formula for calculating individual concentration and the immune selection operator based on above proposed arithmetic-progression rank-based selection with pressure.

- We propose the individual migration strategy according to the immune memory mechanism between subpopulations in hybrid algorithm (For details, see Section 2.4).

2.2.1. Antibody affinity and antibody concentration

Here Antibodies are exactly individuals. They have the same concept and all represent solutions of a given problem. Antibody affinity ay_{vw} defined as follows indicates similar extent between antibody v and antibody w.

$$ay_{vw} = 1/[1 + H(2)] \tag{7}$$

The range of ay_{vw} is within $(0, 1]$. If the value ay_{vw} is higher then the antibody v is more similar with antibody w. At present, $H(2)$ in last formula is mostly calculated by average information entropy formula based on antibody v and w. In fact, as above stated, antibody affinity denotes similar extent between antibodies. In other words, $H(2)$ represents the distance between two antibodies. It can be calculated by average information entropy and also can be calculated by other methods, if two conditions are satisfied. One is $H(2) \geq 0$, and $H(2) = 0$ indicates that the genes of two antibodies are exactly the same. The other is that greater differences between genes of two antibodies can lead to greater value of $H(2)$. In order to simplify the calculation and be easy for real-number coding and engineering realization, we adopt Euclidean distance to calculate affinities.

Let antibody $\mathbf{v} = (v_1, v_2, \ldots, v_n)$ and antibody $\mathbf{w} = (w_1, w_2, \ldots, w_n)$, then

$$H(2) = \sqrt{\sum_{i=1}^{n}(v_i - w_i)^2} \tag{8}$$

If M denotes the population size, concentration c_v of antibody v in its population is usually defined as follows presently.

$$c_v = \frac{1}{M}\sum_{w=1}^{M} ay_{vw} \tag{9}$$

Obviously there exists $c_v \in (0, 1]$.

In order to avoid oscillation during the later period of proposed algorithm and facilitates algorithm convergence, antibody concentration c_v has to tend to 1 ultimately along with increase in the value of generation number K. Therefore, we present a correction for Equ. (8) as follows:

$$C_v = (\frac{1}{M}\sum_{w=1}^{M} ay_{vw})^{(1-\frac{K}{K_{max}})\beta} \tag{10}$$

where β is a system parameter and usually set $\beta = 0.5$.

Apparently antibody affinity and concentration can be regarded as a kind of representation of population diversity in immune algorithms. And the above relevant formulas are given for the sake of convenience of engineering application.

2.2.2. Immune selection operator

The procedure for immune selection of the proposed algorithm can be described as follows:

- Calculate the fitness value and concentration of every antibody (individual) in the population, i.e. F_v and $c_v,\ v = 1, 2, \ldots, M$;
- Calculate the adjusted fitness F'_v of every antibody in the population and there exist $F'_v = F_v/c_v,\ v = 1, 2, \ldots, M$;
- Generate the next population based on adjusted fitness values by above-stated arithmetic-progression rank-based selection with pressure.

Compared with the traditional selection operators, the above immune selection can reflect self regulation function of antibody inhibition and promotion in immune systems. Namely the antibodies with greater fitness values and lower concentration will be promoted and their survival probabilities become larger. On the contrary, the antibodies with lower fitness values and higher concentration will be inhibited and their survival probabilities become smaller. Consequently proposed immune selection can effectively maintain population diversity and prevent proposed algorithm from premature convergence.

2.3. Improved adaptive crossover and mutation

To prevent genetic algorithms from premature convergence effectively as well as protect superior individuals from untimely destruction, the concept of adaptive crossover and mutation is proposed by Srinivas and Patnaik [19]. According to these operators, crossover and mutation rate of the best individual among a population are both zero. It may lead to rather slow evolution in the early stage. To avoid its occurrence, it's better to let the individuals possess due crossover and mutation rates, whose fitness values are equal or approximate to the maximal fitness. Therefore, our crossover P_c and mutation rates P_m are presented as follows:

$$P_c = \begin{cases} k_1 \exp[\frac{(F_{max}-F')}{F_{max}-F_{avg}}(\ln k_3 - \ln k_1)] & F' \geq F_{avg} \\ k_3 & F' < F_{avg} \end{cases} \tag{11}$$

$$P_m = \begin{cases} k_2 \exp[\frac{(F_{max}-F)}{F_{max}-F_{avg}}(\ln k_4 - \ln k_2)] & F \geq F_{avg} \\ k_4 & F < F_{avg} \end{cases} \tag{12}$$

where F_{max} and F_{avg} denote the maximal and average fitness of current population. F' denotes the greater fitness of the two individuals that take part in crossover operation. F denotes the fitness of the individual that take part in mutation operation. k_1, k_2, k_3, k_4 are constants. And there exist $0 < k_1, k_2, k_3, k_4 \leq 1.0, k_1 < k_3, k_2 < k_4$.

2.4. Multi-subpopulation evolution

Simulating the varied and colorful biological communities in nature, we classify all the subpopulations of proposed algorithm into four classes (named Class A, B, C and D) according to their crossover and mutation rates (P_c and P_m). To be simple and convenient, we suppose that there is only one subpopulation within every class. Their parametric features are shown in Table 1.

We can see that the fitness values of initial individuals of Class A subpopulation are the minimal among those of subpopulations of the four classes. But this subpopulation has the highest P_c and P_m, so it is easier for it to explore the new parts of the feasible field and enhance the possibility of discovering global optima. As well as, it can guard against premature convergence. The initial individuals of Class C subpopulation are with relatively greater fitness values. Because this subpopulation has relatively smaller P_c and P_m, it is

Table 1: Parametric features of four classes of subpopulations

Subpopulation	Class A	Class B	Class C	Class D
Crossover rate	$k_1 = 0.8$ $k_3 = 1.0$	$k_1 = 0.5$ $k_3 = 0.8$	$k_1 = 0.2$ $k_3 = 0.5$	$k_1 = 0.1$ $k_3 = 0.2$
Mutation rate	$k_2 = 0.3$ $k_4 = 0.4$	$k_2 = 0.2$ $k_4 = 0.3$	$k_2 = 0.1$ $k_4 = 0.2$	$k_2 = 0.05$ $k_4 = 0.1$
Initial fitness	Minimal	Medium	Greater	Maximal

easier for it to keep the stability of individuals. The function of Class C subpopulation is mainly to consolidate local search. Class B subpopulation is a transitional subpopulation. Its P_c and P_m, as well as the initial fitness values are between those of subpopulations of the above-stated two classes. Class D subpopulation is also called memory subpopulation for it corresponds to memory cells in immune systems. It is made up of the initial individuals with the maximal fitness values among those of subpopulations of the four classes. In the process of evolution, this subpopulation saves the superior individuals obtained by other three subpopulations. At the same time, Class D subpopulation is also evolving itself. But its P_c and P_m are the lowest. The function of Class D subpopulation is to simulate the immune memory function and keep the stability and diversity of the superior individuals. After chaos initialization, the algorithm arranges all the generated individuals according to their fitness values. The initial individuals with the maximal fitness are allocated to Class D subpopulation; the initial individuals with relatively greater fitness are allocated to Class C subpopulation; the initial individuals with the minimal fitness are allocated to Class A subpopulation; the rest of initial individuals are allocated to Class B subpopulation.

The individual migration strategy based on immune memory between subpopulations of the hybrid algorithm is as follows. At intervals of given migration cycle, the algorithm copies the current best individuals in Class A, B and C subpopulations and remembers (saves) them into Class D subpopulation, then update this memory subpopulation (eliminate the inferior individuals from it) and keep the same subpopulation size. Meanwhile, simulating inoculation, the algorithm selects some individuals from the memory subpopulation and makes them migrate to Class A, B and C subpopulations

respectively. The migration individuals will replace the inferior individuals of the three subpopulations respectively as well. This migration strategy can accelerate the convergence rate of the algorithm. In addition, we set a generation control parameter, denoted by K_m. When generation number K is multiples of K_m, the algorithm merges all the subpopulations together and arrange all the individuals according to their fitness values. Then it reallocates individuals to every subpopulation respectively according to their fitness values.

2.5. PSO update operators

2.5.1. Basic theory of particle swarm optimization

Kennedy and Eberhart [20] presented the idea of particle swarm optimization (PSO), in which each particle represents a potential solution. There are mainly two forms of PSO at present, i.e. global version and local version.

With regard to global version of PSO, in the n-dimensional search space, M particles are assumed to consist of a population. The position and velocity vector of the ith particle are denoted by $\boldsymbol{X}_i = (x_{i1}, x_{i2}, \ldots, x_{in})^T$ and $\boldsymbol{V}_i = (v_{i1}, v_{i2}, \ldots, v_{in})^T$ respectively. Then its velocity and position are updated according to the following equations.

$$v_{id}^{k+1} = w \cdot v_{id}^{k} + c_i \cdot rand() \cdot (p_{id}^{k} - x_{id}^{k}) + c_g \cdot rand() \cdot (p_{gd}^{k} - x_{id}^{k}) \tag{13}$$

$$x_{id}^{k+1} = x_{id}^{k} + v_{id}^{k+1} \tag{14}$$

where $i = 1, 2, \ldots, M$; $d = 1, 2, \ldots, n$; k and $k+1$ are iterative numbers. $p_i = (p_{i1}, p_{i2}, \ldots, p_{in})^T$ is the best previous position that ith particle searched so far and $p_g = (p_{g1}, p_{g2}, \ldots, p_{gn})^T$ is the best previous position for whole particle swarm. $rand()$ denotes a uniform random number between 0 and 1. Acceleration coefficients c_i and c_g are positive constants (usually $c_i = c_g = 2.0$) [21]. w is inertia weight and it showed that w decreases gradually along with iteration can enhance entire algorithm performance effectively [22]. It is usually set limitation to a particle velocity. Without loss of generality, assume that relevant following intervals are symmetrical. There exists $v_{id}^{k} \in [-v_{d,max}, +v_{d,max}]$. $v_{d,max}$ $(d = 1, 2, \ldots, n)$ determine the resolution with which regions between present position and target position are searched.

In local version of PSO, particle i keeps track of not only the best previous position of itself, but also the best position $p_{li} = (p_{li,1}, p_{li,2}, \ldots, p_{li,n})^T$ attained by its local neighbor particles rather than that of the whole particle

swarm. Typically, the circle-topology neighborhood model is adopted [23]. Its velocity update equation is

$$v_{id}^{k+1} = w \cdot v_{id}^{k} + c_i \cdot rand() \cdot (p_{id}^{k} - x_{id}^{k}) + c_l \cdot rand() \cdot (p_{li,d}^{k} - x_{id}^{k}) \quad (15)$$

where c_l is also an acceleration coefficient, usually set $c_i = c_l = 2.0$. And its position update equation is same as that of the global version of PSO. Compared with global version of PSO, local version of PSO has a relatively slower convergence rate but it is not easy to be stuck to local optima.

2.5.2. Operator Realization

In PSO-IA, different modes of PSO update operators are introduced into different subpopulations. Global mode PSO update operator is introduced into Class D subpopulation in order to accelerate the convergence of its individuals to global optima. Average mode PSO update operator is introduced into Class C subpopulation so as to help its individuals to consolidate local search around discovered superior solutions. Random mode and synthesis mode PSO update operator are introduced into Class A and B subpopulation respectively. The former matches the function of exploring solution space of Class A subpopulation and helps to prevent algorithm from premature convergence. The latter matches the function of Class B subpopulation and gives consideration to the balance of exploration and exploitation in solution space.

Every mode PSO update operator has the same position update equation, see Equation (14). But their velocity update equations are different. Global mode update operator is on the basis of global version PSO completely. Synthesis mode update operator integrates global version PSO with local version PSO together. In its individual velocity update equation, see Equation (16), three best positions are chased, i.e. the best position an individual visited so far, the best position attained by its local neighbor particles and the best position obtained so far by the whole population.

$$\begin{aligned} v_{id}^{k+1} =& w \cdot v_{id}^{k} + c_i \cdot rand() \cdot (p_{id}^{k} - x_{id}^{k}) \\ &+ c_g \cdot rand() \cdot (p_{gd}^{k} - x_{id}^{k}) + c_l \cdot rand() \cdot (p_{li,d}^{k} - x_{id}^{k}) \end{aligned} \quad (16)$$

According to relevant reference [24, 25], we set $c_i = c_g = 1.5$ and $c_l = 1.1$ in synthesis mode update operator.

In random mode update operator, the neighborhood N_i of particle i is composed of s particles. Apart from particle i itself, the other $s - 1$ particles are randomly selected from the whole population. In this mode update

operator, particle i keeps track of the best previous position of itself and the best position attained within its random neighborhood N_i. In the broad sense, random mode PSO can be regarded as a special kind of local version PSO. Merely its topology structure of neighborhood is dynamic and stochastic. Therefore it helps to explore solution space thoroughly and prevent from premature convergence. We usually set $s = int(0.1 \sim 0.15M)$ and $int(\cdot)$ denotes round-off function.

As for average mode PSO update operator, we first arrange the best position of every particle $\boldsymbol{p}_i$ $(i = 1, 2, \ldots, M)$ in descending order according to their corresponding fitness, then select the front u best positions (here denoted by $\boldsymbol{p}_{gj} = (p_{gj,1}, p_{gj,2}, \ldots, p_{gj,n})^{\mathrm{T}},\ j = 1, 2, \ldots, u$), and change velocity of particle i based on its own best previous position p_i and average of $\boldsymbol{p}_{gj}$ $(j = 1, 2, \ldots, u)$, i.e. $\bar{\boldsymbol{P}}_g = (\bar{P}_{g1}, \bar{P}_{g2}, \ldots, \bar{P}_{gn})^T$, as follows.

$$v_{id}^{k+1} = w \cdot v_{id}^{k} + c_i \cdot rand() \cdot (p_{id}^{k} - x_{id}^{k}) + c_a \cdot rand() \cdot (\bar{p}_{gd}^{k} - x_{id}^{k}) \tag{17}$$

$$\bar{p}_{gd}^{k} = \frac{1}{u} \cdot \sum_{j=1}^{u} p_{gj,d} \tag{18}$$

Similarly c_a is an acceleration coefficient, we set $c_i = c_a = 2.0$ here. Usually $1 \leq u \leq int(0.15M)$ and this mode PSO will reduce to global version PSO if $u = 1$. According to characteristics of different subpopulations, we set the range of w and $\mathbf{V}_{max}$ of every mode of update operator in Table 2. Based on adaptation idea [24], we let w and k (coefficient of maximal velocity) decrease linearly along with evolution from their maximal values to the minimal values.

2.6. Hybrid strategy

To further improve the local search ability of the algorithm, we hybridize another complex method with the proposed algorithm. Complex method [26] possesses relatively fast local convergence rate and doesn't involve derivative information. For computational efficiency, the hybrid algorithm have to give full play to the global search ability of immune algorithm in the early stage, while to the local search ability of complex method in the late stage. Therefore in PSO-IA, we set parameter K_s, randomly select N_s individuals to form initial complex shape and search C_s turns by complex method at intervals of K_s generations. To enhance the local search ability of PSO-IA in the late stage and accelerate convergence rate, N_s and C_s are set in direct proportion to generation number in our algorithm.

Table 2: Relevant settings of every mode of PSO update operator for all classes of subpopulations

Update operator	Random mode	Synthesis mode	Average mode	Global mode
Subpopulation	Class A	Class B	Class C	Class D
Inertia weight w	$w_{max} = 1.5$ $w_{min} = 1.0$	$w_{max} = 1.1$ $w_{min} = 0.6$	$w_{max} = 0.7$ $w_{min} = 0.4$	$w_{max} = 0.6$ $w_{min} = 0.3$
Mutation rate	$k_{max} = 1.0$ $k_{min} = 0.7$	$k_{max} = 0.7$ $k_{min} = 0.4$	$k_{max} = 0.5$ $k_{min} = 0.2$	$k_{max} = 0.3$ $k_{min} = 0.1$

2.7. The procedure of proposed PSO-IA

Flow chart of the proposed hybrid PSO-based immune algorithm (PSO-IA) is shown in Figure 1.

3. Human-computer Cooperative PSO-based Immune Algorithm

According to the idea of human-computer cooperation, we further introduce human intelligence into the algorithm and propose a human-computer cooperative PSO-based immune algorithm (HCPSO-IA) based on above-stated PSO-IA. We describe relevant definitions and operation of HCPSO-IA as follows.

Definition 1. *Artificial individuals*: Swarm-based algorithms operate the population of individuals. Each individual is a representation of a candidate solution to the problems and it is usually a binary string or real number string. An artificial individual in this paper is an individual represents a candidate solution given by designer according to his design experience, a reference template or other available sources (e.g. a sample database).

Definition 2. *Algorithm individuals*: Without the participation of human, the individuals generated by algorithms are called algorithm individuals. It is obviously that the initial individuals obtained by randomized or chaos policy are algorithm individuals. Being in the population and once operated by algorithmic operators, an artificial individual becomes an algorithm individual immediately.

The basic idea of HCPSO-IA is to take artificial individuals as a part of initial population, apply algorithmic operators to the population consisting

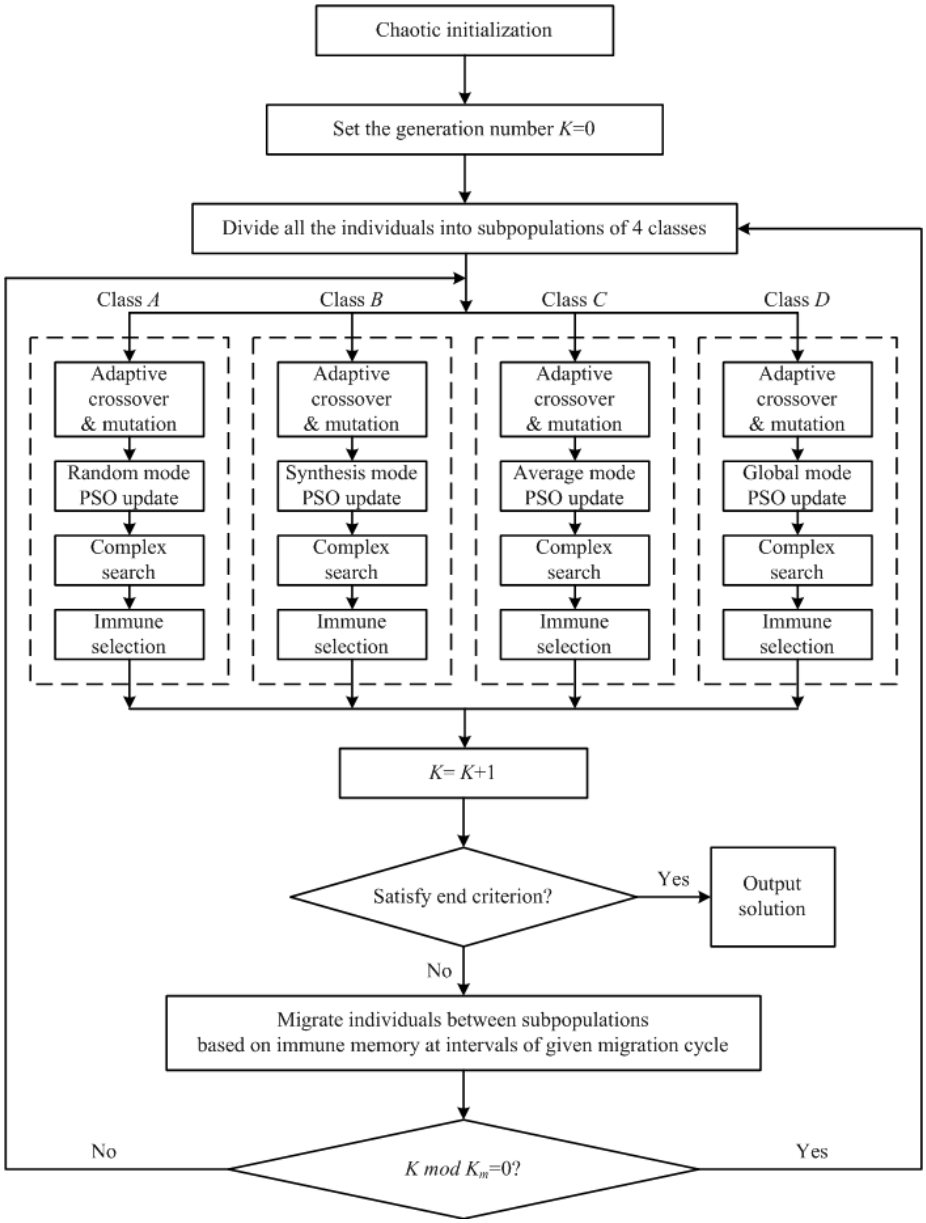


Figure 1: Flow chart of the proposed hybrid PSO-based immune algorithm (PSO-IA)

of artificial individuals and algorithm individuals and introduce new artificial individuals into the population timely during the evolution process.

3.1. Initialization and fitness function

The initial population of HCPSO-IA contains initial artificial individuals provided by designers or sample database and the initial algorithm individuals generated by chaotic initialization method. It deserves to be mentioned that the number of initial artificial individuals $N_m^{(0)}$ should be appropriate. Too many and too few initial artificial individuals can affect the function of the algorithm and human respectively. Based on the trial computation and corresponding result analyses by the mathematical statistics method [15], we suggest that the number of $N_m^{(0)}$ should be $0.25 \sim 0.35M$, M is the population size. After initialization, the algorithm allocates all individuals to four subpopulations according to their fitness values.

Engineering packing and layout design problems are almost complex constrained optimization problems. In this paper, a penalty term is introduced to transform the objective function with constraints into a non-constrained objective function. So the fitness function is further derived from it by exponential scaling as follows.

$$\begin{aligned} F(\boldsymbol{X}) =& exp(f(\boldsymbol{X}) + \sum_{i=1}^{l} \lambda_i g_i(\boldsymbol{X}) u_i(g_i)) \\ u_i(g_i) =& \begin{cases} 0 & \text{if } g_i \text{ satisfies constraint, i.e.} \quad g_i(\boldsymbol{X}) \leq 0, \\ -1 & \text{Otherwise.} \end{cases} \end{aligned} \tag{19}$$

where $F(\boldsymbol{X})$ and $f(\boldsymbol{X})$ are fitness and objective function respectively; $g_i(\boldsymbol{X})$ $(i = 1, 2, \ldots, l)$ are constraint functions and λ_i $(i = 1, 2, \ldots, l)$ are positive penalty factors; l is the number of constraints.

3.2. Human-computer cooperation

We apply above proposed PSO-IA to evolve the population. When generation number K is multiples of K_I (K_I is a given positive integer) or the algorithm is stuck into local optima, new schemes are designed artificially by referring to the superior individuals in population and considering engineering factors in practice. For layout design problems, it is generally can display their layout patterns (or reduced layout patterns) corresponding to

design schemes on output devices (e.g. a computer monitor) by visualization technology. Therefore, it is relatively easy for designers to construct new design schemes. They can directly drag the objects to their ideal positions on computer monitors interactively. Then the new design schemes are encoded and added to the population. The bad individuals can be replaced by the new artificial individuals. Several copies of an artificial individual are added to the population at the same time to ensure that it really works in the following generations. The number of new artificial individuals normally increases with the evolution process so as to take full advantage of the powerful global search ability of PSO-IA in the early stage. Moreover, the new artificial individuals are quite different from the current superior individuals in the early stage in order to contribute to extensively exploring the feasible field of the problem and keeping PSO-IA from premature convergence. While some new artificial individuals are similar to the superior individuals in the late stage in order to enforce the local search around superior individuals and accelerate convergence rate.

Assume that the number of added artificial individuals linearly increases along with the algorithm run. In the late stage of the cooperative algorithm, the added number is a constant and after rounding and taking the actual knowledge level of designers into consideration, we have:

$$N_m^{(k)} = \begin{cases} int\{\frac{1.25\lambda M[(b-a)K+acK_{max}-b]}{cK_{max}-1} - 2.5\lambda + 2\} & 1 \le K \le cK_{max} \\ int\{1.25\lambda(bM-2)+2\} & cK_{max} \le K \le K_{max} \end{cases} \tag{20}$$

where $N_m^{(k)}$ is the number of artificial individuals advised to be added to the population in Kth generation, and $int(\cdot)$ denotes round-off function. K_{max} is the maximal generation number. M is the population size. a, b and c are control parameters and usually $a = 0.05 \sim 0.10$, $b = 0.35 \sim 0.45$, $c = 0.80 \sim 0.95$. Parameter $\lambda \in [0, 1]$ represents knowledge level of the designer. The more λ is close to 1, the higher knowledge level the designer has. λ can be evaluated by fuzzy synthetic evaluation [27].

It should be noted that at every interaction time, the above-mentioned adding artificial individuals and replacing algorithm individuals can be operated only within one subpopulation or several subpopulations at the same time. It depends on the actual situation and designers' experience. The relationship between human and algorithmic operations in our HCPSO-IA can be illustrated as Figure 2, where the real-time monitoring of human includes the adjustment of relevant parameters and other necessary control. The basic

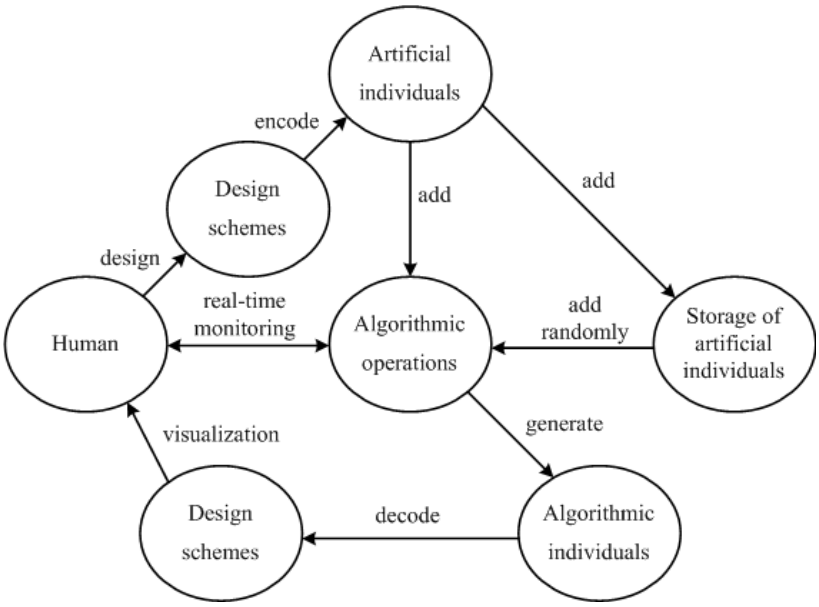


Figure 2: Relationship between human and algorithmic operations

procedure of HCPSO-IA is illustrated in Algorithm 1.

4. Numerical Experiments

4.1. Problem 1

The engineering background of this problem P_1 is the packing and layout design of printed circuit boards (PCB) and plant equipments. Assume that there are n objects named $A_1, A_2, \ldots, A_n$ and the weight between A_i and A_j is $w_{ij}, i, j = 1, 2, \ldots, n$. Try to locate each object such that the value of expression $S + \lambda_w Q$ of a layout scheme is as small as possible and the constraints of no interference between any two objects are satisfied. Here S is the area of enveloping rectangle of a layout scheme. λ_w is a weight factor and Q is the sum of the products of d_{ij} multiplied by w_{ij}, i.e.

$$C = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij} w_{ij} \tag{21}$$

where d_{ij} is the distance between object A_i and A_j. w_{ij} may possess different meanings in different engineering problems. For example, in the packing

Algorithm 1 Human-computer Cooperative PSO-based Immune Algorithm

1: Initialize population. Set $K = 0$. Let S^* be the best individual of initial population. Here S^* denotes the best individual up to now. Ensure that the whole population consists of artificial individuals and algorithm individuals and their numbers are $N_m^{(K)}$ and $N_a^{(K)}$ respectively in the Kth generation. Set parameter K_I represents the interaction cycle.

2: Apply the proposed PSO-IA to the population and evolve it to the next generation. Increase K by 1 .

3: Evaluate each individual. If the best individual of current generation is better than S^*, then assign it to S^*. And if S^* satisfies the end criterion, then go to Step 7.

4: If K is not multiples of K_I or there is no break request from designers, go to Step 2.

5: Design $N_m^{(K)}$ artificial individuals by referring to S^* and the superior individuals of current generation, and taking engineering factors into consideration.

6: Replace some algorithm individuals by new artificial individuals and ensure population size is unchanged, go to Step 2.

7: Output the optimal solution .

and layout design problems of PCB, w_{ij} denotes the connectivity between integrated devices. While in the packing and layout design problems of plant equipments, w_{ij} denotes the adjacent requirement between equipments.

Suppose that (x_i, y_i) is the coordinates of the center of the object A_i. The mathematical model is given by

$$\begin{aligned} &Find\ \mathbf{X} = (x_i, y_i)^T,\ i \in \{1, 2, \ldots, n\} \\ &\quad min f(\mathbf{X}) = S + \lambda_w Q \\ &\quad s.t.\ intA_i \cap intA_j = \varnothing, \quad i \neq j,\ i, j \in \{1, 2, \ldots, n\} \end{aligned} \tag{22}$$

where $intA_i$ presents the interior of object A_i as above.

15 circular objects are contained in P_1 [25]. Let $\lambda_w = 1$. The radii of objects are $r_1 = r_3 = r_{10} = 12mm$, $r_2 = r_4 = 3mm$, $r_5 = r_{13} = r_{14} = 9mm$, $r_6 = r_{12} = r_{15} = 10mm$, $r_7 = 7mm$, $r_8 = 8mm$, $r_9 = 4mm$, $r_{11} = 6mm$. The weight matrix is

$$W = \begin{pmatrix} 0 & 0 & 0 & 98 & 98 & 0 & 81 & 0 & 92 & 93 & 45 & 61 & 99 & 84 & 27 \\ 0 & 0 & 34 & 0 & 0 & 0 & 93 & 44 & 0 & 0 & 33 & 60 & 0 & 0 & 56 \\ 0 & 34 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 85 & 0 & 65 & 39 & 0 & 50 \\ 98 & 0 & 0 & 0 & 91 & 50 & 5 & 24 & 73 & 0 & 4 & 0 & 0 & 31 & 23 \\ 98 & 0 & 0 & 91 & 0 & 37 & 0 & 16 & 78 & 95 & 0 & 0 & 73 & 32 & 0 \\ 0 & 0 & 0 & 50 & 37 & 0 & 0 & 35 & 0 & 31 & 0 & 0 & 0 & 48 & 0 \\ 81 & 93 & 0 & 5 & 0 & 0 & 0 & 94 & 33 & 34 & 26 & 61 & 0 & 87 & 87 \\ 0 & 44 & 0 & 24 & 16 & 35 & 94 & 0 & 91 & 0 & 0 & 0 & 59 & 39 & 0 \\ 92 & 0 & 0 & 73 & 78 & 0 & 33 & 91 & 0 & 0 & 30 & 0 & 0 & 0 & 0 \\ 93 & 0 & 85 & 0 & 95 & 31 & 34 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 45 & 33 & 0 & 4 & 0 & 0 & 26 & 0 & 30 & 0 & 0 & 0 & 21 & 35 & 2 \\ 61 & 60 & 65 & 0 & 0 & 0 & 61 & 0 & 0 & 0 & 0 & 0 & 56 & 0 & 43 \\ 99 & 0 & 39 & 0 & 73 & 0 & 0 & 59 & 0 & 0 & 21 & 56 & 0 & 1 & 0 \\ 84 & 0 & 0 & 31 & 32 & 48 & 87 & 39 & 0 & 0 & 35 & 0 & 1 & 0 & 0 \\ 27 & 56 & 50 & 23 & 0 & 0 & 87 & 0 & 0 & 0 & 2 & 43 & 0 & 0 & 0 \end{pmatrix} \tag{23}$$

We adopt PGA, PSO-IA and HCPSO-IA to solve the problems respectively. Here the widely used island model, i.e. coarse-grained PGA is employed. To compare the performance of the algorithms objectively, every algorithm we adopt in this paper possesses four subpopulations. And any relevant contents of the three algorithms, such as population size, encoding scheme, fitness function and migration cycle, that may be identical are selected as the same. In the aspect of algorithm initialization, stochastic initialization and chaos initialization are relatively used in PGA and PSO-IA. And as above stated, the initial population of HCPSO-IA contains artificial individuals provided by designers as well as algorithm individuals generated by chaos

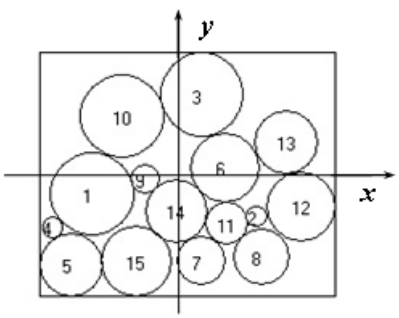
Table 3: Comparison of obtained results of the best layouts by three algorithms of P_1

Algorithms	S/mm^2	Q	t_1/s	t_2/s	t/s
PGA	5996.46	89779.16	29.79		29.79
PSO-IA	5586.97	86861.74	30.82		30.82
HPSO-IA	5410.58	76169.97	16.26	18	34.26

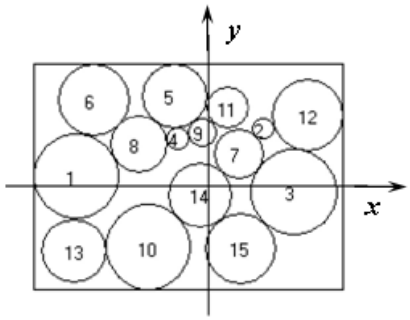
initialization. In the aspect of algorithm operators, PSO-IA and HCPSO-IA have same operators (such as adaptive crossover and mutation, PSO update and immune selection), while proportional model selection, two-point crossover and uniform mutation are used in PGA. The migration strategy of PGA we adopted in this paper is as follows. At intervals of given migration cycle, PGA copies several superior individuals of every subpopulation, sends to another arbitrarily taken subpopulation and replaces the inferior individuals of the subpopulation. It is worth mentioning that the human intervention is applied to help to overcome local minima traps, when we adopt HCPSO-IA to solve two problems. That is to say, when designers consider that the algorithm is stuck into local optima artificial individuals will be added into the algorithm. All computation is performed on a PC with CPU at 2.1GHz and RAM size of 2GB and non-interference requirements are guaranteed to obtain optimal results.

All the three algorithms are run 20 times respectively. The best layouts of P_1 among 20 results are shown in Figure 3. The Comparison of obtained results of the best layouts is given in Table 3. In this table, the time spent on calculation and human-computer interaction is denoted as t_1 and t_2 respectively, so the total cost of time is $t = t_1 + t_2$. The relevant symbols possess the same meanings in the following tables of this paper.

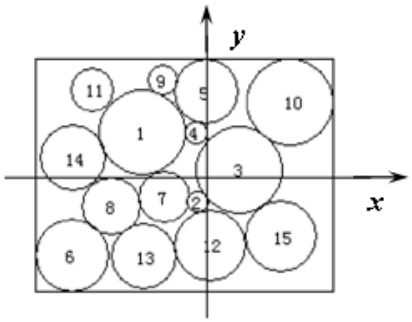
As shown in Table 3, for the best layout by PGA, the area of enveloping rectangle S, the parameter Q and computation time t are $5996.46mm^2$, 89799.16 and 29.79s. In the sense of best results, compared with PGA, PSO-IA reduces S and Q by 6.83% and 3.27%, i.e. from $5996.46mm^2$ to $5586.97mm^2$ and from 89799.16 to 86861.74 respectively. HCPSO-IA further reduces them by 9.77% and 15.16%, i.e. from $5996.46mm^2$ to $5410.58mm^2$ and from 89799.16 to 76169.97 respectively. When obtained $S \leq 5996.46mm^2$ and $Q \leq 89799.16$, PSO-IA takes $21.15s$ and HCPSO-IA takes $24.48s$ (in-



(a) Best layout by PGA



(b) Best layout by PSO-IA



(c) Best layout by HCPSO-IA

Figure 3: The obtained best layout patterns of P_1 by three algorithms

Table 4: Comparison of average values of optimal results by three algorithms of P_1

Algorithms	S/mm^2	Q	K
PGA	6153.83	95739.06	705
PSO-IA	5778.69	87461.15	503
HPSO-IA	5547.53	78034.84	387

cluding $12s$ interaction time). Therefore compared with PGA, to reach the same precision, PSO-IA and HCPSO-IA reduced the cost of time by 29.00% and 17.82% respectively.

Table 4 lists relevant average values of obtained twenty optimal results of P_1 by three algorithms. K represents elapsed generation number for an optimal result. Compared with PGA, on an average, PSO-IA reduces the area of enveloping rectangle S, the parameter Q and elapsed generation number K by 6.10%, 8.65% and 28.65%, i.e. from $6153.83mm^2$ to $5778.69mm^2$, from 95739.06 to 87461.15 and from 705 to 503. HCPSO-IA further reduces them by 9.85%, 18.49% and 45.11%, i.e. from $6153.83mm^2$ to $5547.53mm^2$, from 95739.06 to 78034.84 and from 705 to 387 respectively.

4.2. Problem 2

The engineering background of this problem P_2 is the layout design of satellite modules [28]. There are 12 rectangular objects and 8 circular objects will be located on a bearing plate with radius $R = 1800mm$. Dimensions of all the objects are given in Table 5, in which a_i and b_i are one half of the longer and shorter edge for a rectangular object while r_i is the radius for a circular object, and m_i denotes the mass of every object. The unit of a_i, b_i, r_i is mm, and m_i is Kg. There are three rectangular objects (i.e. No. 18, 19 and 20 objects shown in Figure 4) that are fixed on the plate in advance. The allowable value of static non-equilibrium is $[\delta_J] = 2.5Kg \cdot mm$. Try to locate each object such that these objects highly concentrate on the center of the container and the constraints of no interference and static equilibrium

Table 5: Dimensions and mass of objects to be located

No.	a_i	b_i	r_i	m_i	No.	a_i	b_i	r_i	m_i
1	365	365		4.0	11			250	3.2
2	425	300		4.5	12			340	2.6
3	250	225		3.6	13			130	1.5
4	290	215		3.0	14			90	0.7
5	175	175		2.1	15			150	1.4
6	350	135		3.3	16			360	2.6
7	330	140		2.0	17			160	1.0
8	250	175		1.8	18	300	250		2.0
9	100	80		1.1	19	225	175		1.5
10			450	4.0	20	250	225		1.8

behavior are satisfied. The mathematical model of P_2 is given as follows.

$$\begin{aligned} & Find\ \mathbf{X} = (x_i, y_i, \alpha_i)^T,\ i \in \{1, 2, \ldots, n\} \\ & min\ F(\mathbf{X}) = R^* = maxR_i \\ s.t.\ & intA_i \cap intA_j = \varnothing, \quad i \neq j,\ i, j \in \{1, 2, \ldots, n\} \\ & maxR_i \leq R \qquad i = 1, 2, \cdots, n \\ & J = \sqrt{\left(\sum_{i=1}^{n} m_i x_i\right) + \left(\sum_{i=1}^{n} m_i y_i\right)} \leq [\delta_J] \end{aligned} \tag{24}$$

where (x_i, y_i) is the coordinates of the centroid of the ith object (denoted by A_i) and bearing angle α_i is the orientation it is located in. For a rectangular object, α_i is the included angle between its long side and the positive semi-axis x. It is defined to be positive by anticlockwise within the bound of $[0, \pi]$. In the case of circular objects, α_i equals 0. R_i is the radius of enveloping circle of object A_i. R^* and J denote the radius of enveloping circle and the value of static non-equilibrium of the entire layout scheme respectively. $[\delta_J]$ represents the allowable value of J. $int\ A_i$ presents the interior of object A_i.

We adopt PGA, PSO-IA and HCPSO-IA to solve P_2 respectively. All the three algorithms are run 20 times respectively. The best layouts of P_2 among 20 results by them are shown in Figure 4. The comparison of the

Table 6: Comparison of obtained results of the best layouts by three algorithms of P_2

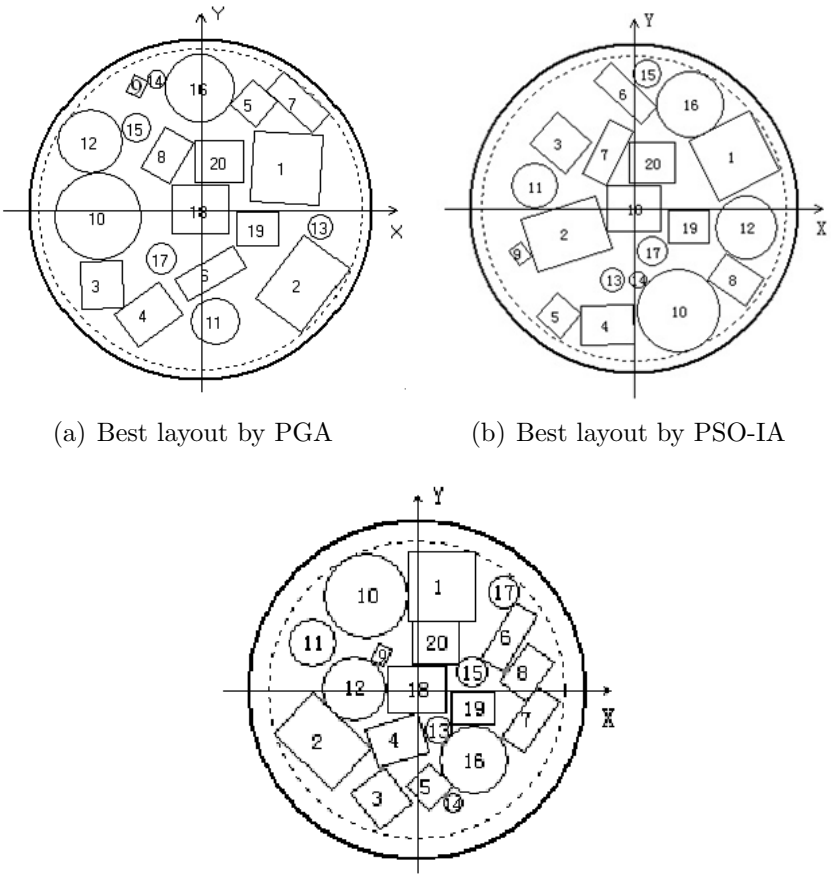
Algorithms	R^*/mm	$J/Kg \cdot mm$	t_1/s	t_2/s	t/s
PGA	1703.57	1.277	182.29		182.29
PSO-IA	1666.27	0.310	201.47		201.47
HPSO-IA	1587.58	0.001	163.65	52	215.65

obtained results of the best layouts is given in Table 6, where R^* and J denote the radius of enveloping circle and the value of static non-equilibrium of a layout scheme respectively. For the best layout by PGA, the radius of the enveloping circle R^*, the value of static non-equilibrium J and computation time t are $1703.57mm$, $1.277Kg{\cdot}mm$ and $182.29s$. In the sense of best results, compared with PGA, PSO-IA reduces R^* and J by 2.19% and 75.72%, i.e. from $1703.57mm$ to $1666.27mm$ and from $1.277Kg \cdot mm$ to $0.310Kg \cdot mm$ respectively. HCPSO-IA further reduced them by 6.81% and 99.92%, i.e. from $1703.57mm$ to $1587.58mm$ and from $1.277Kg \cdot mm$ to $0.001Kg \cdot mm$ respectively. When obtained $R^* \leq 1703.57mm$ and $J \leq 1.277Kg{\cdot}mm$, PSO-IA takes $145.06s$ and HCPSO-IA takes $138.72s$ (including $35s$ interaction time). Therefore compared with PGA, to reach the same precision, PSO-IA and HCPSO-IA reduced the cost of time by 20.42% and 23.90% respectively.

Table 7 lists relevant average values of obtained twenty optimal results of P_2 by the three algorithms. And it shows that compared with PGA, on an average, PSO-IA reduces the radius of the enveloping circle R^*, the value of static non-equilibrium J and elapsed generation number K by 3.77%, 63.59% and 25.70%, i.e. from $1779.75mm$ to $1712.61mm$, from $2.153Kg{\cdot}mm$ to $0.784Kg \cdot mm$ and from 1327 to 986. HCPSO-IA further reduces them by 7.94%, 99.58% and 51.85%, i.e. from $1779.75mm$ to $1638.42mm$, from $2.153Kg \cdot mm$ to $0.009Kg \cdot mm$ and from 1327 to 639 respectively.

5. Conclusions

In order to solve packing and layout problems more effectively, we took several strategies on PGA and presented an improved hybrid algorithm named PSO-IA. These measures involve introducing immunity principle, PSO update operators, hybrid strategy, modified rank-based selection, as well as



(a) Best layout by PGA (b) Best layout by PSO-IA

(c) Best layout by HCPSO-IA

Figure 4: The obtained best layout patterns of P_2 by three algorithms

Table 7: Comparison of average values of optimal results by three algorithms of P_2

Algorithms	R^*/mm	$J/Kg \cdot mm$	K
PGA	1779.75	2.153	1327
PSO-IA	1712.61	0.784	986
HPSO-IA	1638.42	0.009	639

multi-subpopulation evolution based on improved adaptive crossover and mutation. Additionally, in order to contribute to solving layout design problems of complex engineering systems satisfactorily, we further propose a novel human-computer cooperative PSO-based immune algorithm (HCPSO-IA) to realize the idea of man-machine synergy in engineering application. In HCPSO-IA, the artificial individuals supplied by designers properly, together with algorithm individuals, consist of the whole population of the algorithm. Artificial individuals also replace the relevant individuals at an opportune moment during evolutionary process. HCPSO-IA benefits by giving free rein to the talents of designers and computers, and can realize the cooperation of their intelligence at the algorithmic level.

Two problems originated from engineering layout design demonstrate the effectiveness of the proposed algorithms and verify that the attempt is meaningful. According to obtained results, it is clear that PSO-IA and HCPSO-IA depict better performances than traditional parallel genetic algorithms. Some key performance indices, such as envelope area and static non-equilibrium value, have been significantly improved by the proposed two algorithms. By contrast, HCPSO-IA has higher performance than PSO-IA. This demonstrates that the introduction of human knowledge and experience is really beneficial in obtaining better solutions to engineering problems. Experience also illustrate that the more complex a problem is, the proposed algorithms are superior to the traditional algorithms, both in providing quality solutions and computationally more efficient, though HCPSO-IA is not satisfactory for some relatively simple problems in terms of time requirement. The reason lies in that the time spent on human-computer interaction may occupy rather high percentage in the total time cost for solving simple problems. In future, it is worthwhile making attempts to our approaches to solving more packing and layout problems in practice. It deserves to be noted that our methodology is problem-independent and can also be easily applied to other engineering problems.

6. Acknowledgment

The authors would like to thank Jialu Du, Chen Guo and Hongying Hu for their scientific collaboration in this research work. This work is partly supported by the National Natural Science Foundation of China (Grant No. 51079013, 61074053 and 61173035), the Fundamental Research Funds for the Central Universities (Grant No. DMU2011NQ030, DC120101014 and

DC110320), the Program for New Century Excellent Talents in University (Grant No. NCET-11-0861) and the Applied Basic Research Program of Ministry of Transport of China (No. 2011-329-225-390, No. 2012-329-225-070).

References

[1] J. Cagan, K. Shimada, S. Yin, A survey of computational approaches to three-dimensional layout problems, Computer-Aided Design 34 (8) (2002) 597–611.

[2] I. Jankovits, C. Luo, M. Anjos, A. Vannelli, A convex optimisation framework for the unequal-areas facility layout problem, European Journal of Operational Research 214 (2) (2011) 199–215.

[3] Z. Qian, H. Teng, Algorithms of complex layout design problems, China Mechanical Engineering 13 (8) (2002) 696–699.

[4] J. Schutte, A. Groenwold, A study of global optimization using particle swarms, Journal of Global Optimization 31 (1) (2005) 93–108.

[5] S. Das, A. Abraham, A. Konar, Automatic clustering using an improved differential evolution algorithm, IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 38 (1) (2008) 218–237.

[6] M. Alrashidi, M. El-Hawary, A survey of particle swarm optimization applications in electric power systems, IEEE Transactions on Evolutionary Computation 13 (4) (2009) 913–918.

[7] C. Kolias, G. Kambourakis, M. Maragoudakis, Swarm intelligence in intrusion detection: A survey, Computers & Security 30 (8) (2011) 625–642.

[8] Y. Sun, L. Zhang, X. Gu, A hybrid co-evolutionary cultural algorithm based on particle swarm optimization for solving global optimization problems, Neurocomputing 98 (2012) 76–89.

[9] D. Lenat, E. Feigenbaum, On the thresholds of knowledge, Artificial Intelligence 47 (1) (1991) 185–250.

[10] R. Al-Kasasbeh, M. Alshamasin, F. Ionescu, N. Korenevskiy, Modelling and parameter estimation for operator intelligence in man-machine systems, International Journal of Modelling, Identification and Control 15 (1) (2012) 69–85.

[11] Y. Fu, The study on human computer interaction design method based on unified interactive mode, International Journal of Digital Content Technology and Its Applications 6 (2) (2012) 180–187.

[12] D. Knysh, V. Kureichik, Parallel genetic algorithms: a survey and problem state of the art, Journal of Computer and Systems Sciences International 49 (4) (2010) 579–589.

[13] A. Sokolov, D. Whitley, A. da Motta Salles Barreto, A note on the variance of rank-based selection strategies for genetic algorithms and genetic programming, Genetic Programming and Evolvable Machines 8 (3) (2007) 221–237.

[14] E. Boudissa, M. Bounekhla, Genetic algorithm with dynamic selection based on quadratic ranking applied to induction machine parameters estimation, Electric Power Components and Systems 40 (10) (2012) 1089–1104.

[15] G. Li, Research on theory and methods of layout design and their applications, Ph.D. thesis, Dalian University of technology, Dalian, China (2003).

[16] R. Satheesh Kumar, P. Asokan, S. Kumanan, An artificial immune system-based algorithm to solve linear and loop layout problems in flexible manufacturing systems, International Journal of Product Development 10 (1) (2010) 165–179.

[17] X. Xu, C. Li, Research on immune genetic algorithm for solving the job-shop scheduling problem, The International Journal of Advanced Manufacturing Technology 34 (7) (2007) 783–789.

[18] A. Rami, A. Zeblah, H. Hamdaoui, Y. Massim, An efficient artificial immune algorithm for power system reliability optimisation, International Journal of Power and Energy Conversion 1 (2) (2009) 178–197.

[19] M. Srinivas, L. Patnaik, Adaptive probabilities of crossover and mutation in genetic algorithms, IEEE Transactions on Systems, Man and Cybernetics 24 (4) (1994) 656–667.

[20] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, Vol. 4, IEEE, 1995, pp. 1942–1948.

[21] H. Liu, A. Abraham, M. Clerc, Chaotic dynamic characteristics in swarm intelligence, Applied Soft Computing 7 (3) (2007) 1019–1026.

[22] A. Nickabadi, M. Ebadzadeh, R. Safabakhsh, A novel particle swarm optimization algorithm with adaptive inertia weight, Applied Soft Computing 11 (4) (2011) 3658–3670.

[23] J. Kennedy, R. Mendes, Population structure and particle swarm performance, in: Proceedings of the 2002 Congress on Evolutionary Computation, Vol. 2, IEEE, 2002, pp. 1671–1676.

[24] K. Kameyama, Particle swarm optimization-a survey, IEICE Transactions on Information and Systems 92 (7) (2009) 1354–1361.

[25] G. Li, Evolutionary algorithms and their application to engineering layout design, Postdoctoral research report, Tongji University, Shanghai, China (2005).

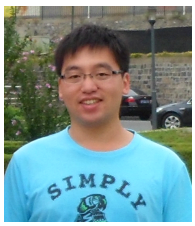
[26] S. Li, L. Ding, L. Zhao, W. Zhou, Optimization design of arch dam shape with modified complex method, Advances in Engineering Software 40 (9) (2009) 804–808.

[27] L. Wu, L. Zhang, J. Zhou, The safety assessment model of stone arch bridge based on fuzzy synthetic evaluation, International Journal of Digital Content Technology and Its Applications 6 (6) (2012) 43–52.

[28] Y. Wang, Y. Shi, X. Wang, H. Teng, Opposition-based cooperative coevolutionary differential evolution algorithm with gaussian mutation for simplified satellite module optimization, Information Technology Journal 11 (2012) 67–75.

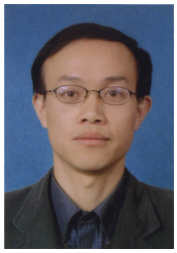
Fengqiang Zhao received his Ph.D. degree in Mechanical and Electronic Engineering from Dalian University of Technology, Dalian, China, in 2006. He is a Lecturer at College of Electromechanical & Information Engineering, Dalian Nationalities University, China. Currently he is also a Postdoctoral Fellow in School of Information Science and TechnologyDalian Maritime University, China. His research interests include intelligent systems and optimization, fault diagnosis and noise control.

Guangqiang Li is an Associate Professor at School of Information Science and Technology, Dalian Maritime University, China. He received his Ph.D. degree from Dalian University of Technology in 2003. During 2004 and 2005, he was a Postdoctoral Fellow in Tongji University, Shanghai, China. His current research focuses on intelligent systems and optimization, evolutionary algorithms, ship motion control and complex layout design.



Chao Yang received his B. Eng. degree in biomedical engineering from Northeast University of China in 2011. Currently, he is a Ph.D. candidate at School of Information Science and Technology, Dalian Maritime University. His research interests include computational intelligence and bioinformatics.

Ajith Abraham is the Director of Machine Intelligence Research Labs (MIR Labs), Scientific Network for Innovation and Research Excellence, WA, USA, with a joint appointment in the IT For Innovations - Center of excellence at VSB - Technical University of Ostrava, Czech Republic. He also holds an Adjunct Professor appointment in Dalian Maritime University, China. He received the Ph.D. degree in Computer Science from Monash University, Melbourne, Australia. His research and development experience includes more than 23 years in the industry and academia. He works in a multidisciplinary environment involving machine intelligence, network security, various aspects of networks, e-commerce, Web intelligence, Web services, computational grids, data mining and their applications to various real-world problems. He has authored/co-authored more than 900 publications (h-index=50+), and some of the works have also won best paper awards at international conferences. He has given more than 60 plenary lectures and conference tutorials in these areas. He serves/has served the editorial board of over 50 International journals and has also guest edited over 40 special issues on various topics. Since 2008, he is the Chair of IEEE Systems Man and Cybernetics Society Technical Committee on Soft Computing and a Distinguished Lecturer of IEEE Computer Society representing Europe (since 2011). More information at: http://www.softcomputing.net.



Hongbo Liu received his three level educations (B. Sc., M. Sc., Ph.D.) at the Dalian University of Technology, China. Currently he is a Professor at School of Information Science and Technology, Dalian Maritime University, with an affiliate appointment in the Institute for Neural Computation, University of California San Diego, USA. His research interests are in system modeling and optimization involving soft computing, probabilistic modeling, cognitive computing, machine learning, data mining, etc. He participates and organizes actively international conference and workshop and international journals/publications.