Send Orders for Reprints to reprints@benthamscience.ae

The Open Automation and Control Systems Journal, 2014, *6*, 813-822

A New Hybrid PSO-based Genetic Algorithm and its Application to Layout Problems

Fengqiang Zhao[1,2,*], Guangqiang Li[1,3], Jialu Du[1], Chen Guo[1], Tao Li[1], Xinwen Fu[3] and Rubo Zhang[2]

[1]*College of Information Science and Technology, Dalian Maritime University, Dalian 116026, China*

[2]*College of Electromechanical & Information Engineering, Dalian Nationalities University, Dalian 116600, China*

[3] *Department of Computer Science, University of Massachusetts Lowell, Lowell, MA 01854, United States*

Abstract: Layout problems belong to NP-Complete problems theoretically. They are concerned more and more in recent years and arise in a variety of application fields such as the layout design of spacecraft modules, plant equipments, platforms of marine drilling well, shipping, vehicle and robots. The algorithms based on swarm intelligence are relatively effective to solve these kind of problems. But usually there still exist two main defects, i.e. premature convergence and slow convergence rate. To overcome them, a new improved hybrid PSO-based genetic algorithm (HPSO-GA) is proposed on the basis of parallel genetic algorithms (PGA). In this algorithm, chaos initialization, hybrid strategy and multi-subpopulation evolution based on improved adaptive crossover and mutation are adopted. The proposed interpolating rank-based selection with pressure can prevent the algorithm from premature in the early stage and benefit accelerating convergence in the late stage as well. And more importantly, in accordance with characteristics of different classes of subpopulations, different modes of PSO update operator are introduced. It aims at making full use of the fast convergence property of particle swarm optimization (PSO). An example of layout problems shows that HPSO-GA is feasible and effective.

Keywords: Genetic algorithms, Particle swarm optimization, Hybrid methods, Layout.

1. INTRODUCTION

Layout problems [1, 2] are to study how to put objects into limited space reasonably under constraints (for example, no interference, increasing space utilization ratio). In complex problems (e.g. the layout design of spacecraft modules and plant equipments), some extra behavioral constraints should be taken into consideration, such as the requirements for equilibrium, connectivity and adjacent states. These kinds of problems are often faced in many engineering fields and they are of great importance. They usually directly affect some performance indices of design, such as reliability and economy. Since these problems belong to NP-Complete problems, it is quite difficult to solve them satisfactorily.

Relevant references [1, 3, 4] summarized the common methods for solving layout problems, including mathematical programming and criterion methods, heuristic algorithms, graph theory, expert systems and algorithms based on swarm intelligence and natural laws. According to the algorithm trend and solution quality, it shows that the robust universal algorithms based on swarm intelligence and natural laws are of advantage. And they are particularly fit to solve medium or large-scale complex layout problems, compared with other traditional methods [5]. But there still exist some defects with regard to themselves, such as premature convergence and slow convergence rate. To overcome them, some measures are taken and a new hybrid PSO-based genetic algorithm (HPSO-GA) is proposed based on PGA [6]. It aims at solving layout problems more effectively.

2. HYBRID PSO-BASED GENETIC ALGORITHM (HPSO-GA)

The experimental oil is from a platform in Bohai. The OSD is the FuKen-Ⅱ OSD from Qingdao Huahai Environmental Protection Industry Co., Ltd, which is composed mainly of high purity oil-based and non-ionic surface active agents (oxidized lipids), coupler and penetrating agent. The emulsification is that: 30s, > 60%; 10min, > 20% with the using range of 20% -70% in the oils. Experimental sample is a kind of mixture of the OSD and the oil. The quality ratio of the OSD and the oil (expressed by m_D/m_O) is 0, 0.1, 0.2, 0.4 and 0.6. The GC is the Shimadzu Co. 2010 gas chromatography.

2.1. Chaos Iinitialization

The purpose of adopting chaos initialization is to improve the quality of initial individuals. Chaos is a nonlinear

1874-4443/14

phenomenon, which extensively exists in nature [7]. Chaos systems possess the characteristics, such as randomness, ergodicity and sensibility to initial conditions [8]. By means of these characteristics, we can initialize population superiorly. The basic idea of chaos initialization can be stated as follows. First of all, generate the same number of chaos variables as many as decision variables. Then introduce chaos into decision variables and map the ergodic range of chaos variables onto the definition ranges of decision variables. Here we select the following Logistic mapping as chaos generator.

$$Z_{k+1}= f(\mu, Z_k)= \mu Z_k(1- Z_k)\ k=0,1,2, \ldots \quad (1)$$

where μ is a control parameter and the system is in chaotic state when μ=4.

The concrete procedure of chaos initialization is as follows. Assume that the number of decision variables is n. Firstly assign n original values Z_{i0} (i=1, 2,…, n) to Z_k in formula (1), which are all between 0 and 1. So it can generate n different sequences of chaos variables, i.e. $\{Z_{ik}, i=1, 2,\ldots, n\}$. Then introduce every chaos variable into its corresponding decision variable by (2).

$$x_{ik}= a_i+ (b_i- a_i)\ Z_{ik} \quad (2)$$

where b_i and a_i are the upper and lower bounds of decision variable x_i respectively.

For a given k, decision vector $\boldsymbol{X}_k=(x_{1k}, x_{2k},\ldots, x_{nk})^{\mathrm{T}}$ represents a solution (an individual) to the problem. Along with increase in the value of k, we can obtain a series of initial individuals. Finally, we calculate the fitness of every obtained individual, select superior individuals to form initial population and divide it into several initial subpopulations.

2.2. Interpolating Rank-based Selection with Pressure

In traditional rank-based selection operator of genetic algorithms, a probability assignment table should be preset. But there is no deterministic rule for design of the table. And it is difficult for traditional rank-based model to make the selection probabilities of individuals adaptively changed along with evolution process. So some research works have been devoted to the improvement of traditional rank-based selection for these years [9, 10]. In this paper, based on the mathematical concept of interpolation method, we introduce interpolating rank-based selection with pressure and its relevant formulas. It can overcome the above-stated shortcomings of traditional rank-based selection operator.

Parameter Decision

There are three control parameters in this kind of selection. They are selection pressure, distribution of interpolation points corresponding to individuals and probabilistic interpolating function.

Selection pressure α denotes the ratio of the maximal individual selection probability $P_{\max}$ to the minimal individual selection probability $P_{\min}$ within a generation, i.e. $P_{\max}=\alpha P_{\min}$. This parameter numerically shows the superiority that the better individuals are reproduced into the next generation during selection process and it is changeable along with the evolution process of the algorithm. Because the fitness values of individuals within a population are usually not much different from one another in the final stage of genetic algorithms and traditional proportional selection model can't assign higher selection probability values to superior individuals, it usually takes a long time to converge to final results for genetic algorithms. However the proposed concept of changeable selection pressure α can overcome this difficulty effectively. In the early stage, lesser selection pressure α can maintain population diversity and prevent the algorithm from premature convergence. While in the late stage, greater selection pressure α can benefit accelerating algorithm convergence.

Let $\alpha=f(K)$, K and f denote the generation number and an increasing function respectively. f can be multiple forms and we adopt linear increasing function for the sake of simplicity. Let $\alpha_{\max}$ and $\alpha_{\min}$ denote the maximum and minimum of selection pressure respectively, then

$$\alpha=\frac{(K-1)(\alpha_{\max}-\alpha_{\min})}{K_{\max}-1}+\alpha_{\min} \quad (3)$$

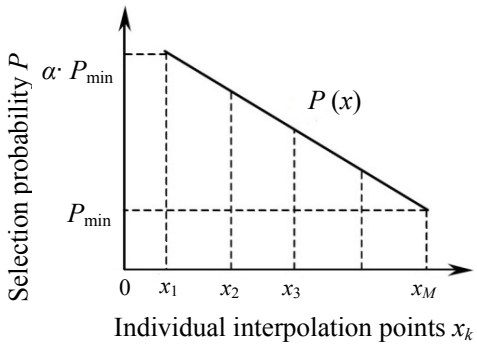
where $K_{\max}$ is the maximal generation number set in algorithm. And our numerical experiments show that $\alpha_{\max}$ and $\alpha_{\min}$ may be chosen in the interval [6, 15] and [1.5, 5] respectively [11].

To calculate the selection probability of every individual, we should arrange all the individuals within a population in descending order based on their fitness values at first. And then determine every interpolation point corresponding to every individual. Interpolation points can be denoted by $x_{k+1}=x_k+ h_k$, k=1, 2, …, M-1, where M is the population size. x_k is the kth interpolation point corresponding to the kth individual within the descending order arrangement. h_k is the step size of interpolation. If $h_k=c$ (k=1, 2, …, M-1), c is a constant, then the distribution of individual interpolation points is equidistant. Otherwise, it is inequidistant. The concrete distribution types should be determined according to the requirement of actual computational condition. For example, under the circumstances of the same α and $P(x)$ (see the next paragraph), comparing the equidistant distribution shown in Fig. (**1a**) to the inequidistant distribution with more compact ends shown in Fig. (**1b**), we know that the latter lays more emphasis on the function of the superior individuals with greater fitness values.

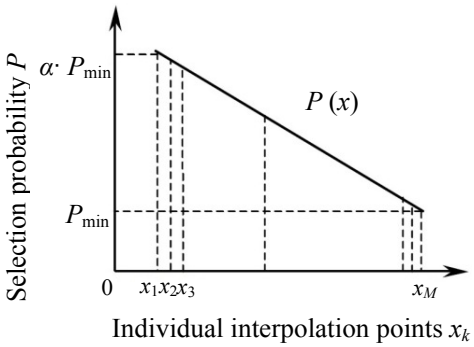
$P(x)$ is called probabilistic interpolating function and it is a decreasing function. The selection probability of the kth individual is $P_k=P(x_k)$. And there exist $P_{\min}= P(x_M)$ and $P_{\max}=P(x_1)$. $P_{\max}$ and $P_{\min}$ are the maximal and minimal selection probability respectively. $P(x)$ can be linear or nonlinear functions.

Realization Process

Assume that the probabilistic interpolating function is linear and the distribution of individual interpolation points is equidistant. We present derived formulas of calculating selection probabilities of individuals in this case as follows. The relevant formulas in other cases can be derived similarly.

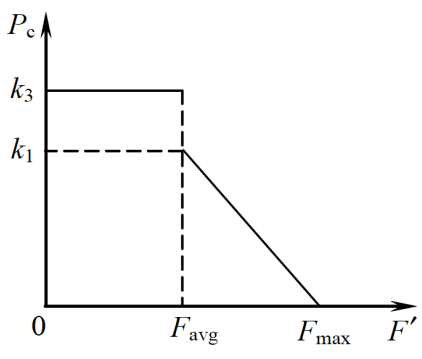


(a) Equidistant distribution

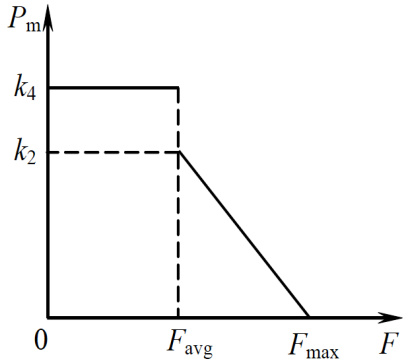


(b) Inequidistant distribution

Fig. (1). Distribution types of individual interpolation points.



(a) Value curve of P_c



(b) Value curve of P_m

Fig. (2). Adaptive crossover & mutation operators by Ref. [12].

As it's shown in Fig. (**1a**), let $\delta_k = P(x_k)\text{-}P(x_{k+1})$, k=1,2, … , M-1. And assume that the difference between P_{max} and P_{min} is $\Delta = (\alpha\text{-}1)P_{min}$. Because $P(x)$ is a linear function and $h_k=x_{k+1}\text{-}x_k=c$ (k=1, 2, …, M-1), δ_k(k=1,2,…, M-1) is a constant, denoted by δ. And there exists $\delta = \Delta /(M\text{-}1)=[(\alpha\text{-}1)P_{min}]/(M\text{-}1)$. Therefore the selection probability of the kth individual is

$$P_k = \alpha P_{min} + [(1-\alpha)P_{min}(k-1)]/(M-1) \quad (4)$$

The sum of all the individual selection probability is 1, i.e.

$$\sum_{k=1}^{M}\left[\alpha \cdot P_{min} + \frac{(1-\alpha)\cdot P_{min}\cdot(k-1)}{M-1}\right] = 1 \quad (5)$$

Therefore we obtain

$$P_{min} = \frac{2}{M(\alpha+1)} \quad (6)$$

Substituting above formula into formula (4), it is easy to find that

$$P_k = \frac{2\alpha\cdot(M-k)+2(k-1)}{M\cdot(\alpha+1)\cdot(M-1)} \quad k=1, 2, \ldots, M \quad (7)$$

In the process of proposed selection operation, we firstly reproduce the best individual of current generation and put its copy into the next generation directly based on elitist model. And then figure out selection probabilities of all the individuals according to formula (7). Finally generate the remaining M-1 individuals of the next generation by fitness proportional model. The best advantage of proposed selection operation is that it can conveniently change the selection probabilities of individuals by changing selection pressure during the evolution process. As a result, the selection operation can be more adaptive to the algorithm run.

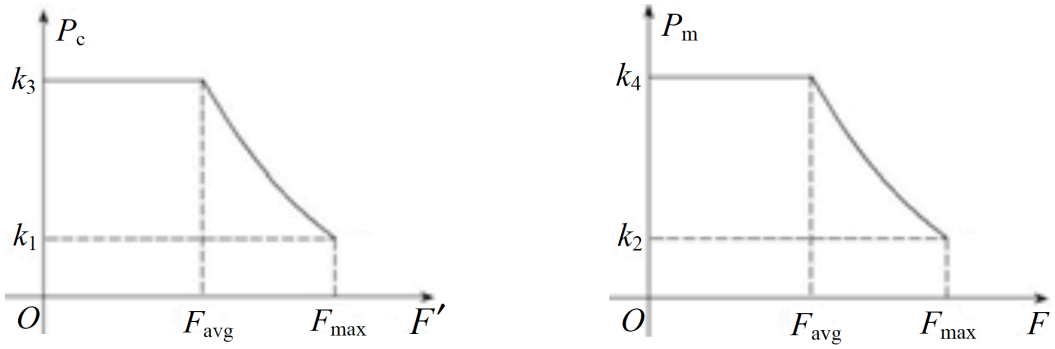
2.3 Improved Adaptive Crossover and Mutation

To prevent genetic algorithms from premature effectively as well as protect superior individuals from untimely destruction, the concept of adaptive crossover and mutation is proposed by Srinivas and Patnaik [12], see (8) and (9) and shown in Fig. (**2**). Here P_c and P_m denote crossover and mutation rate respectively.

$$P_c = \begin{cases} k_1(F_{max}-F')/(F_{max}-F_{avg}), & F' \geq F_{avg} \\ k_3, & F' < F_{avg} \end{cases} \quad (8)$$

$$P_m = \begin{cases} k_2(F_{max}-F)/(F_{max}-F_{avg}), & F \geq F_{avg} \\ k_4, & F < F_{avg} \end{cases} \quad (9)$$

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 $0<k_1, k_2, k_3, k_4\leq 1.0$, $k_1<k_3$, $k_2<k_4$.



(a) Value curve of P_c (b) Value curve of P_m

Fig. (3). Improved adaptive crossover & mutation operators.

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_1=0.5	k_1=0.2	k_1=0.1
	k_3=1.0	k_3=0.8	k_3=0.5	k_3=0.2
Mutation rate	k_2=0.3	k_2=0.2	k_2=0.1	k_2=0.05
	k_4=0.4	k_4=0.3	k_4=0.2	k_4=0.1
Initial fitness	Minimal	Medium	Greater	Maximal

But 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, improved adaptive crossover rate P_c and mutation rate P_m are presented as follows and shown in Fig. (**3**).

$$P_c = \begin{cases} k_1 \exp\left[\dfrac{(F_{max} - F')}{F_{max} - F_{avg}}\left(\ln k_3 - \ln k_1\right)\right], & F' \geq F_{avg} \\ k_3, & F' < F_{avg} \end{cases} \quad (10)$$

$$P_m = \begin{cases} k_2 \exp\left[\dfrac{(F_{max} - F)}{F_{max} - F_{avg}}\left(\ln k_4 - \ln k_2\right)\right], & F \geq F_{avg} \\ k_4, & F < F_{avg} \end{cases} \quad (11)$$

The basic idea of the improved adaptive operators can be described as follows. When the fitness value of an individual is less than the average fitness of the whole population, this individual is assigned greater crossover and mutation rates. It contributes to further exploration of solution space and prevention the algorithm from premature. While when the fitness value of an individual is greater than the average fitness of the whole population, the crossover and mutation rate of this individual decline exponentially with the increase of its fitness value. It can help the algorithm to enforce the exploitation ability and consolidate local search around superior individuals.

2.4. Multi-subpopulation Evolution

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). Suppose that there is only one subpopulation within every class, named class A, B, C and D subpopulation respectively. Their parametric features are shown in Table **1**.

According to their properties of initial fitness as well as crossover and mutation rates, we can see that it is easier for class A subpopulation to explore new parts of solution space and guard against premature. Class C subpopulation is mainly to consolidate local search. Class B subpopulation is a transitional subpopulation. And the function of class D subpopulation is to keep stability and diversity of superior individuals. After chaos initialization, HPSO-GA 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 between subpopulations of HPSO-GA is as follows. At intervals of given migration cycle, HPSO-GA copies the best individuals in class A,

B and C subpopulation and saves them into class D subpopulation, then update class D subpopulation (eliminate the inferior individuals from it) and keep the same subpopulation size. Meanwhile, it selects some individuals from class D subpopulation and makes them migrate to class A, B and C subpopulation respectively. The migration individuals will replace inferior individuals in above subpopulations respectively as well. This migration strategy can accelerate convergence. In addition, we set control parameter K_m. When generation number K is multiples of K_m, HPSO-GA merges all the subpopulations together and arrange all individuals according to their fitness. Then it reallocates individuals to every subpopulation respectively according to their fitness values.

2.5. PSO Update Operators

Basic Theory of Particle Swarm Optimization

Kennedy and Eberhart [13] presented the idea of particle swarm optimization (PSO). In PSO, each particle as an individual in genetic algorithms 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 formulas.

$$v_{id}^{k+1} = w \cdot v_{id}^{k} + c_1 \cdot rand() \cdot \left(p_{id}^{k} - x_{id}^{k} \right) + c_2 \cdot rand() \cdot \left(p_{gd}^{k} - x_{id}^{k} \right) \tag{12}$$

$$x_{id}^{k+1} = x_{id}^{k} + v_{id}^{k+1} \tag{13}$$

where i=1,2,…, M; d=1,2,…, n; k and k+1 are iterative numbers. $\boldsymbol{p}_i = (p_{i1}, p_{i2}, \ldots, p_{in})^T$ is the best previous position that ith particle searched so far and $\boldsymbol{p}_g = (p_{g1}, p_{g2}, \ldots, p_{gn})^T$ is the best previous position for whole particle swarm. *rand*() denotes a uniform random number between 0 and 1. Acceleration coefficients c_1 and c_2 are positive constants (usually c_1=c_2=2.0). w is inertia weight and it showed that w decreases gradually along with iteration can enhance entire algorithm performance effectively [14].

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,…, n) determine the resolution with which regions between present position and target position are searched. If $v_{d,\max}$ is too high, particles may fly past good solutions. While, if it is too small, the algorithm may be stuck to local optima. Suppose that the range of definition for the dth dimension of a position vector is $[-x_{d,\max}, +x_{d,\max}]$, i.e. $x_{id}^{k} \in [-x_{d,\max}, +x_{d,\max}]$. Usually let $\pm v_{d,\max} = \pm k x_{d,\max}$, $0.1 \le k \le 1$.

In local version of PSO, particle i keeps track of not only the best previous position of itself, but also the best position $\boldsymbol{p}_{li} = (p_{li,1}, p_{li,2}, \ldots, p_{li,n})^T$ attained by its local neighbor particles rather than that of the whole particle swarm. Typically, the circle-topology neighborhood model is adopted [15]. Its velocity update formula is

$$v_{id}^{k+1} = w \cdot v_{id}^{k} + c_1 \cdot rand() \cdot \left(p_{id}^{k} - x_{id}^{k} \right) + c_2 \cdot rand() \left(p_{li,d}^{k} - x_{id}^{k} \right) \tag{14}$$

And its position update formula 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.

PSO has been applied to many fields and results are satisfactory [16]. It is easy to be implemented and has quite fast convergence rate among evolutionary algorithms. But it also has the limitations such as low precision and premature. Noticed that genetic algorithms and PSO are both based on swarm intelligence and can match each other fairly well. To make full use of fast convergence property of PSO and global convergence ability of genetic algorithms, we propose this hybrid algorithm. Specifically, let velocity and position update formulas together serve as a new operator (PSO update). After conventional genetic operation, individuals go on with PSO update operation. It hopes to make hybrid algorithm possess more superior global performance.

Operator Realization

In HPSO-GA, different modes of PSO update operator 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. 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 formula, see formula (13). But their velocity update formulas 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 formula, see formula (15), 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.

$$v_{id}^{k+1} = w \cdot v_{id}^{k} + c_1 \cdot rand() \cdot \left(p_{id}^{k} - x_{id}^{k} \right) + c_2 \cdot rand() \cdot \left(p_{gd}^{k} - x_{id}^{k} \right) + c_3 \cdot rand() \left(p_{li,d}^{k} - x_{id}^{k} \right) \tag{15}$$

where c_3 is the acceleration coefficient of the newly added item. According to Ref. [16, 17], we set c_1 = c_2 =1.5 and c_3=1.1 in synthesis mode update operator.

Table 2. Relevant settings of PSO update operators 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
Coefficient k	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

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. We usually set s=*int*(0.1~0.15M) and *int*(·) 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,…,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,…, u. And change velocity of particle i based on its own best previous position $\boldsymbol{p}_i$ and average of $\boldsymbol{p}_{gj}$ (j=1, 2,…, u), i.e. $\overline{\boldsymbol{p}}_g=(\overline{p}_{g1},\overline{p}_{g2},\ldots,\overline{p}_{gn})^{\mathrm{T}}$, as follows.

$$v_{id}^{k+1}=w\cdot v_{id}^{k}+c_1\cdot rand()\cdot\left(p_{id}^{k}-x_{id}^{k}\right)+c_2\cdot rand()\cdot\left(\overline{p}_{gd}^{k}-x_{id}^{k}\right) \tag{16}$$

$$\overline{p}_{gd}^{k}=\frac{1}{u}\cdot\sum_{j=1}^{u}p_{gj,d} \tag{17}$$

Usually $1\le u\le$ *int*(0.15M) and this mode PSO will reduce to global version PSO if u=1.

We lay emphasis on two parameters in PSO update operator, i.e. inertia weight w and maximal velocity $\boldsymbol{V}_{\max}$. Usually there exist $w\in$ [0.3, 1.5], $\pm v_{d,\max}=\pm kx_{d,\max}$ ($0.1\le k\le 1.0$). If they select greater values, the update operator is more likely to find out new parts of solution space. Otherwise the update operator is good at local search. According to characteristics of different subpopulations, we set the range of w and $\boldsymbol{V}_{\max}$ of every mode of update operator in Table **2**. Based on adaptation idea [14], 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 local search ability of the proposed algorithm, it is necessary to apply hybrid strategy. Taking the matching problem into consideration, we hybridize complex method with proposed algorithm. Complex method [18] possesses relatively fast local convergence rate and doesn't involve derivative information. Allowing for the problem of computational efficiency, the hybrid algorithm should give full play to the global search ability of genetic algorithm in the early stage, while to the local search ability of complex method in the late stage. Therefore in HPSO-GA, 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 HPSO-GA in the late stage and accelerate convergence rate, N_s and C_s are set in direct proportion to generation number in the proposed algorithm.

2.7. The Procedure of Proposed Algorithm

Flow chart of the proposed hybrid PSO-based genetic algorithm (HPSO-GA) is shown in Fig. (**4**).

3. NUMERICAL EXAMPLE

The engineering background of this example is the layout design of printed circuit boards (PCB) and plant equipments. Assume that there are n objects named A_1, A_2, …, A_n and the weight between A_i and A_j is w_{ij}, i, j=1,2,…, n. Try to locate each object such that the value of expression $S+\lambda_w C$ 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 C is the sum of the products of d_{ij} multiplied by w_{ij}, i.e.

$$C=\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}d_{ij}w_{ij} \tag{18}$$

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 PCB layout design problems, w_{ij} denotes the connectivity between integrated devices. While in the 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 for this problem is given by

$$\begin{aligned}&\text{Find } \boldsymbol{X}=(x_i, y_i)^{\mathrm{T}},\ i\in\{1,2,\ldots,n\}\\&\min f(\boldsymbol{X})=S+\lambda_w C\\&\text{s.t. } \mathrm{int}A_i\cap\mathrm{int}A_j=\varnothing\ i\ne j,\ i,j\in\{1,2,\ldots,n\}\end{aligned} \tag{19}$$

where $intA_i$ presents the interior of object A_i.

Quoted from Ref. [17], 15 circular objects are contained in this example. Let λ_w=1. The radii of objects are r_1=r_3=r_{10}=12 mm, r_2=r_4=3 mm, r_5=r_{13}=r_{14}=9 mm, r_6=r_{12}=r_{15}=10 mm, r_7=7 mm, r_8=8 mm, r_9=4 mm, r_{11}=6mm. The weight matrix is

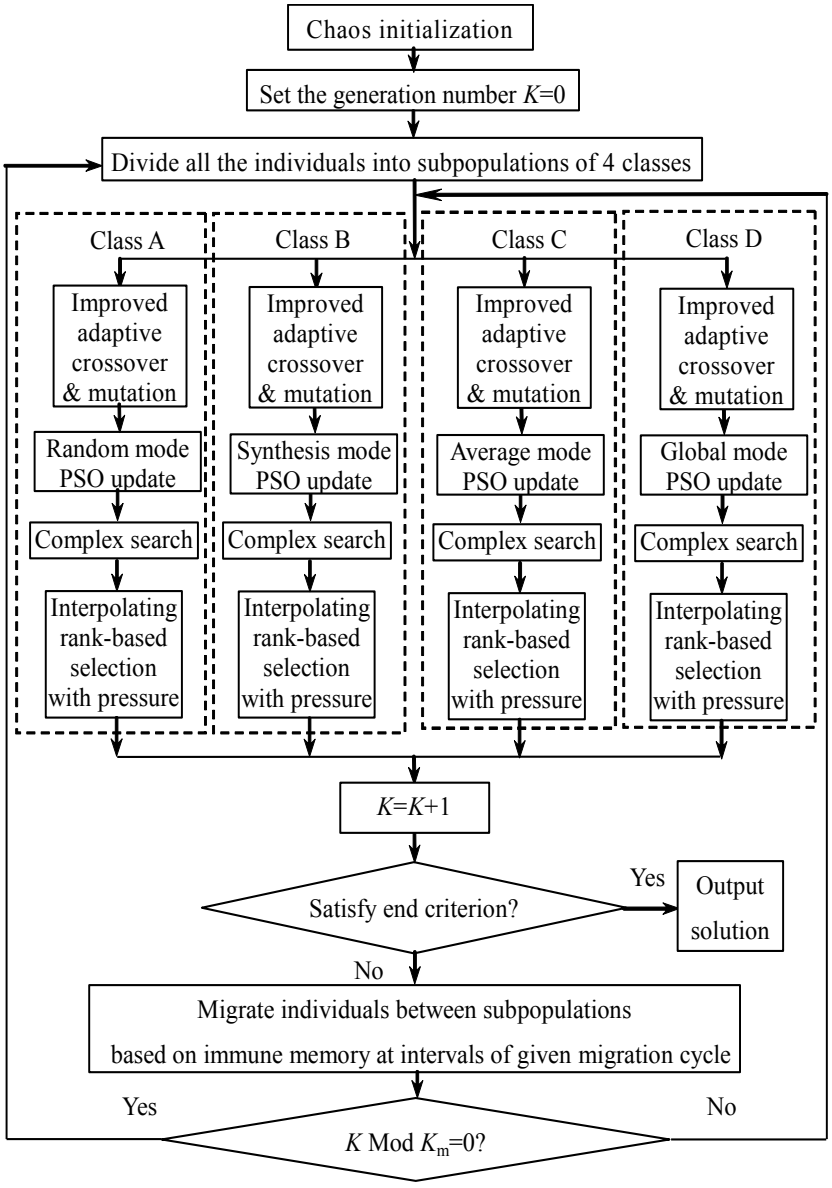


Fig. (4). Flow chart of the proposed hybrid PSO-based genetic algorithm (HPSO-GA).

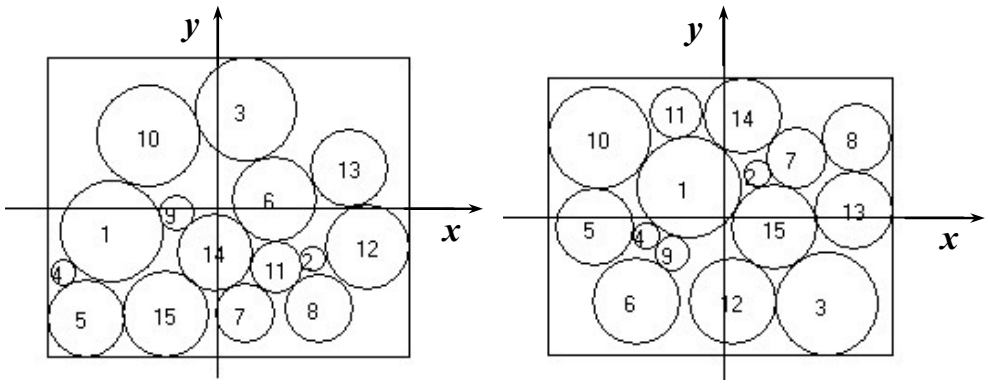
$$W = \begin{pmatrix}
0 & 0 & 0 & 98 & 98 & 0 & 81 & 0 & 92 & 93 & 45 & 61 & 99 & 84 & 27 \\
0 & 0 & 34 & 0 & 0 & 0 & 93 & 44 & 0 & 0 & 33 & 60 & 0 & 0 & 56 \\
0 & 34 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 85 & 0 & 65 & 39 & 0 & 50 \\
98 & 0 & 0 & 0 & 91 & 50 & 5 & 24 & 73 & 0 & 4 & 0 & 0 & 31 & 23 \\
98 & 0 & 0 & 91 & 0 & 37 & 0 & 16 & 78 & 95 & 0 & 0 & 73 & 32 & 0 \\
0 & 0 & 0 & 50 & 37 & 0 & 0 & 35 & 0 & 31 & 0 & 0 & 0 & 48 & 0 \\
81 & 93 & 0 & 5 & 0 & 0 & 0 & 94 & 33 & 34 & 26 & 61 & 0 & 87 & 87 \\
0 & 44 & 0 & 24 & 16 & 35 & 94 & 0 & 91 & 0 & 0 & 0 & 59 & 39 & 0 \\
92 & 0 & 0 & 73 & 78 & 0 & 33 & 91 & 0 & 0 & 30 & 0 & 0 & 0 & 0 \\
93 & 0 & 85 & 0 & 95 & 31 & 34 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
45 & 33 & 0 & 4 & 0 & 0 & 26 & 0 & 30 & 0 & 0 & 0 & 21 & 35 & 2 \\
61 & 60 & 65 & 0 & 0 & 0 & 61 & 0 & 0 & 0 & 0 & 0 & 56 & 0 & 43 \\
99 & 0 & 39 & 0 & 73 & 0 & 0 & 59 & 0 & 0 & 21 & 56 & 0 & 1 & 0 \\
84 & 0 & 0 & 31 & 32 & 48 & 87 & 39 & 0 & 0 & 35 & 0 & 1 & 0 & 0 \\
27 & 56 & 50 & 23 & 0 & 0 & 87 & 0 & 0 & 0 & 2 & 43 & 0 & 0 & 0
\end{pmatrix} \quad (20)$$

To compare the performance of HPSO-GA with that of traditional PGA objectively, we adopt HPSO-GA and the PGA that possesses four subpopulations (same as HPSO-GA) to solve this example respectively and the subpopulation sizes of both algorithms are identical. Moreover, any relevant contents of the two algorithms, such as encoding scheme, fitness function and migration cycle, that may be identical are selected as the same. 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. All computation is performed on PC with CPU at 2.1GHz and RAM size of 2GB.

Both algorithms are calculated 20 times respectively. The best layouts among 20 optimal results by them are in Table **3** and the corresponding best geometric layout patterns are shown in Fig. (**5**). The comparison of obtained results of the

Table 3. The best layouts by two algorithms of the example.

No.	The Best Layout by PGA		The Best Layout by HPSO-GA	
	x_i/mm	y_i/mm	x_i/mm	y_i/mm
1	-24.80	-5.70	-8.03	6.85
2	22.85	-12.11	7.81	9.98
3	6.97	23.00	24.19	-20.64
4	-36.29	-15.28	-18.06	-4.36
5	-30.97	-26.00	-29.91	-2.44
6	13.69	2.08	-20.27	-19.87
7	6.76	-24.90	17.11	13.80
8	24.21	-23.85	31.24	18.62
9	-9.48	-1.18	-11.85	-8.68
10	-16.21	16.85	-28.92	18.57
11	14.12	-14.15	-11.04	24.54
12	35.70	-9.21	2.16	-19.88
13	31.22	9.35	30.54	1.49
14	-0.42	-10.45	4.48	23.72
15	-12.10	-25.00	11.81	-2.47



(**a**) Best layout by PGA (**b**) Best layout by HPSO-GA

Fig. (5). The obtained best layout patterns of the example by two algorithms.

best layouts is given in Table **4**. In Table **4**, ΔS and t denote the interference area and computation time respectively.

As the data presented in Table **4**, for the best layout by PGA, S, C and computation time t are 5996.46mm^2, 89779.16 and 29.79s; for the best layout by HPSO-GA, S, C and t are 5258.63mm^2, 79082.28 and 27.53s. When obtained $S \leq$ 5996.46mm^2, $C \leq$ 89779.16 by HPSO-GA, it takes 22.91s. So in the sense of best results, to reach the same precision, HPSO-GA reduces the cost of time by 23.10% compared with PGA.

Table **5** lists relevant average values of obtained twenty optimal results of the example by two algorithms. In this table, K represents elapsed generation number for an optimal result.

Table **5** shows that compared with PGA, on an average, HPSO-GA reduces the area of enveloping rectangle S, the parameter C and elapsed generation number K by 12.05%, 9.17% and 27.38%, i.e. from 6153.83mm^2 to 5412.23mm^2, from 95739.06 to 86962.57 and from 705 to 512 respectively.

Table 4. Comparison of obtained results of the best layouts by two algorithms of the example.

Algorithms	S/mm^2	C	ΔS/mm^2	t/s
PGA	5996.46	89779.16	0	29.79
HPSO-GA	5258.63	79082.28	0	27.53

Table 5. Comparison of average values of optimal results by two algorithms of the example.

Algorithms	S/mm^2	C	ΔS/mm^2	K
PGA	6153.83	95739.06	0	705
HPSO-GA	5412.23	86962.57	0	512

CONCLUSION

In order to solve layout problems more effectively, we take several measures on PGA and propose a new improved hybrid algorithm named HPSO-GA. These measures involve introducing chaos initialization, hybrid strategy, interpolating rank-based selection with pressure as well as multi-subpopulation evolution based on improved adaptive crossover and mutation into proposed algorithm. And more importantly, the idea of particle swarm optimization is introduced and PSO update operator can improve the global performance of the proposed algorithm. A numerical example shows that HPSO-GA is feasible and effective for this kind of problems. It is really superior to PGA in accuracy and convergence rate. Our work is expected to provide inspiration and reference for solving engineering layout problems satisfactorily. In addition, because HPSO-GA is a universal algorithm, it also can be adopted to solve other complex engineering optimization problems.

CONFLICT OF INTEREST

The author confirms that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

We would like to express our gratitude to the National Natural Science Foundation of China (No. 51079013, No. 61074053 and No. 61374114), the Fundamental Research Funds for the Central Universities of China (No. DMU2011NQ030, No. DC120101014, No. DC110320), the Applied Basic Research Program of Ministry of Transport of China (No. 2011-329-225-390, No. 2012-329-225-070), the China Scholarship council (No. 201306575010), and the Higher Education Research Fund of Education Department of Liaoning Province of China (No. LT2010013) for financial support of our work.

REFERENCES

[1] J. Cagan, K. Shimada, and S. Yin, "A survey of computational approaches to three-dimensional layout problems," *CAD Computer Aided Design*, vol. 34, no. 8, pp. 597-611, 2002.

[2] A. Jankovits, C. Luo, M. F. Anjos, and A. Vannelli, "A convex optimization framework for the unequal-areas facility layout problem", *European Journal of Operational Research*, vol. 214, no. 2, pp. 199-215, 2011.

[3] Z. Q. Qian, and H. F. Teng, "Algorithms of complex layout design problems", *China Mechanical Engineering*, vol. 13, no. 8, pp. 696-699, 2002.

[4] J. Yang, and J. Yang, "Intelligence optimization algorithms: a survey", *International Journal of Advancements in Computing Technology*, vol. 3, no. 4, pp. 144-152, 2011.

[5] Y. Sun, L. Zhang, and X. Gu, "A hybrid co-evolutionary cultural algorithm based on particle swarm optimization for solving global optimization problems", *Neurocomputing*, vol. 98, pp. 76-89, 2012.

[6] D. S. Knysh, and V. M. Kureichik, "Parallel genetic algorithms: a survey and problem state of the art", *International Journal of Computer and Systems Sciences*, vol. 49, no. 4, pp. 579-589, 2010.

[7] C. P. Silva, "Survey of chaos and its applications", *Proceedings of the 1996 IEEE MTT-S International Microwave Symposium Digest*, San Francisco, CA, pp. 1871-1874, June 1996.

[8] C. Li, J. Zhou, P. Kou, and J. Xiao, "A novel chaotic particle swarm optimization based fuzzy clustering algorithm", *Neurocomputing* , vol. 83, pp. 98-109, 2012.

[9] A. Sokolov, D. Whitley, and A. D. M. Salles-Barreto, "A note on the variance of rank-based selection strategies for genetic algorithms and genetic programming", *Genetic Programming and Evolvable Machines*, vol. 8, no. 3, pp. 221-237, 2007.

[10] E. Boudissa, and M. Bounekhla, "Genetic algorithm with dynamic selection based on quadratic ranking applied to induction machine parameters estimation", *Electric Power Components and Systems*, vol. 40, no. 10, pp. 1089-1104, 2012.

[11] G. Q. Li, "*Research on theory and methods of layout design and their applications*", Ph.D. dissertation, Dalian University of technology, Dalian, China, 2003.

[12] M. Srinivas, and L. M. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms", *IEEE Transactions on Systems, Man and Cybernetics*, vol. 24, no. 4, pp. 656-667, 1994.

[13] J. Kennedy, and R. Eberhart, "Particle swarm optimization," *Proceedings of the IEEE International Conference on Neural Networks*, Perth, Australia, pp. 1942-1948, November 1995.

[14] A. Nickabadi, M. M. Ebadzadeh, and R. Safabakhsh, "A novel particle swarm optimization algorithm with adaptive inertia weight", *Applied Soft Computing Journal*, vol. 11, no. 4, pp. 3658-3670.

[15] K. Jame, and M. Rui, "Population structure and particle swarm performance, *Proceedings of the 2002 Congress on Evolutionary Computation*, Honolulu, HI, USA, pp. 1671-1676, 2002.

[16] K. Kameyama, "Particle swarm optimization: A survey", *IEICE Transactions on Information and Systems*, vol. 92, no. 7, pp. 1354-1361, 2009.

[17] G. Q. Li, "Evolutionary algorithms and their application to engineering layout design", *Postdoctoral Research Report*, Tongji University, Shanghai, China, 2005.

[18] S. Li, L. Ding, L. Zhao, and W. Zhou, "Optimization design of arch dam shape with modified complex method," *Advances in Engineering Software,* vol. 40, no. 9, pp. 804-808, 2009.

Received: November 26, 2014 Revised: January 08, 2015 Accepted: January 21, 2015