A Niche Sharing Scheme-based Co-evolutionary Particle Swarm Optimization Algorithm for Flow Shop Scheduling Problem

Bin Jiao
Electric School, Shanghai Dianji University,
Shanghai, 201306,China
binjiaocn@163.com

and

Shaobin Yan
School of Information Science and Engineering, East China University of Science and Technology
Shanghai,200237,China
Yshaobin123@sina.com

ABSTRACT

By taking advantage of niche sharing scheme,we propose a novel co-evolutionary particle swarm optimization algorithm (NCPSO) to solve permutation flow shop scheduling problem. As the core of this algorithm, niche sharing scheme maximizes the diversity of population and hence improves the quality of individuals. To evaluate the performance of the proposed algorithm, we have use eight Taillard instances with different sizes to extensive experiment and results clearly shown that the solutions found by NCPSO algorithm outperform those by Particle Swarm Optimization (PSO), Genetic Algorithm (GA) and Cooperative Particle Swarm Optimization (CPSO).

Keywords: Co-evolutionary approach, Particle swarm optimization, Niche Sharing Scheme, Flow Shop Scheduling Problem.

1. INTRODUCTION

Since its invention, intelligent optimization algorithm, which is also referred to as evolutionary algorithm, has played an increasingly important role in a wide range of fields, including genetic algorithm, ant colony algorithm, etc... Evolutionary algorithm, as indicated by its name, is inspired and developed by evolution and behavior of animals, namely bionics and has been widely used to deal with the optimization problem in both continuous and discrete domains.

Flow shop scheduling problem (FSSP), which is a complex combinatorial optimization problem with a strong engineering background at present represents approximately a quarter of manufacturing systems and information service facilities in use. As for the permutation FSSP, the goal is to find a job permutation of all the jobs to be processed on several machines so that a specific performance measure is minimized. Thus far, two of the most common measures are the minimization of *makespan* and total flow time, which have been proved to be NP-complete by Garey et al. [1].

Johnson [2] first proposes an optimization algorithm to minimize *makespan* for 2-machine FSSP with n jobs to schedule. From then on, various methods have been developed so as to solve FSSP; however, some of them are only able to cope with small- and moderate-sized problems. Hence, a great deal of effort has been dedicated to obtaining satisfactory solutions to complex FSSP. At the very beginning, the heuristic methods concentrate on settling the *makespan* minimization problem containing Palmer's slope index [3], CDS [4], Gupta's [5] heuristic and NEH [6]. Unfortunately, there is one common shortcoming with the heuristic methods: the pre-defined rules the heuristic methods depend on might not be applicable to some practical problems. As a result, evolution-based algorithms, such as tabu search method[7], simulated annealing algorithm [8], genetic algorithm[9][10], ant colony optimization (ACO) [11] and particle swarm optimization algorithm[12][13][14], have been proposed as a replacement of the heuristic methods to handle the scheduling problems. Yi Zhang et al. [15] propose an HGA (hybrid genetic algorithm) for permutation FSSP with a minimization in total flow time. G.I. Zobolas et al. [16] proposed a hybrid metaheuristic for the minimization of makespan in permutation flow shop scheduling problems. A genetic algorithm for solution evolution and a variable neighborhood search (VNS) to improve the population. The hybridization of a GA with VNS, combining the advantages of these two individual components, is the key innovative aspect of the approach, in which comprises three components: an initial population generation method based on a greedy randomized constructive heuristic. Li and Pan [17] presented a novel hybrid algorithm (TABC) that combines the artificial bee colony (ABC) and tabu search (TS) to solve the hybrid flow shop (HFS) scheduling problem with limited buffers. The objective is to minimize the maximum completion time.

Over the recent years, there have been a number of reported works focusing on the modification PSO and other optimization algorithms to solve continuous optimization problems. Nevertheless, they do not work anymore when used to solve FSSP, and thus far only few algorithms are available for FSSP. Changsheng Zhang et al. [18] propose a hybrid alternate two phase particle swarm optimization (PSO) algorithm called ATPPSO to address FSSP, by taking advantage of the PSO with genetic operators and annealing strategy to minimize *makespan*. Jindong Zhang et al. [19] propose a circular discrete particle swarm optimization algorithm CDPSO instead for FSSP. However, these algorithms have the problem of premature convergence, which is as a consequence of easy trap into a local optimum. In general, niche technology is used in cooperation with other algorithms. Jun Zhang et al. [20] propose a novel adaptive sequential niche particle swarm optimization (ASNPSO) algorithm. By taking advantage of the dynamic niche sharing technique, Xiyu Liu et al. [21] presents a new variation of traditional PSO algorithm. T. Radha Ramanan et al.[22] with the objective of optimizing the *makespan* of an FSSP uses a particle swarm optimization (PSO) approach. Variable neighborhood search (VNS) is employed to overcome the early convergence of the PSO and helps in global search. The shortest maximum completion time was taken as the goal, process industrial production scheduling algorithms based on particle swarm optimization (PSO) was proposed in paper [23], the specific production tasks of propylene oxide (PO) and polyvinyl chloride (PVC) of a chlor-alkali enterprises was taken as the research background, four production scheduling tasks of two products was realized. Gonzalez [24] et al proposed effective neighborhood structures for this problem, including feasibility and non-improving conditions, as well as procedures for fast estimation of the neighbor's quality. These neighborhoods are embedded into a scatter search algorithm which uses tabu search and path relinking in its core. Lei and Guo [25] formulated the problem as a mixed integer linear programming model and develop an effective parallel neighborhood search algorithm. Two-string representation and three neighborhood structures are applied to generate new solutions.

In this paper, we propose a new intelligent optimization algorithm, which is also called co-evolution particle swarm optimization algorithm, based on niche particle swarm optimization (NCPSO). Taking into account co-evolution particle swarm optimization algorithm, a new swarm with niche sharing scheme is designed to cooperate with other swarms. In this niche evolutionary environment, crossover and mutation operations are involved to search optimal solution. In addition, five other swarms are designed to assist each other during the process of best solution search. Therefore, NCPSO algorithm is a paralleling co-evolutionary process.

The rest of the paper is organized as follows. In section 2, we briefly describe the FSSP. Section 3 and section 4 introduce particle swarm optimization, co-evolution algorithm and the principle of NCPSO algorithm. We apply the proposed NCPSO algorithm to the flow shop scheduling optimization in section 5. Finally, we draw a conclusion in section 6.

2. FORMULATION OF FSSP

Let the 4-tuple $<n, m, P, Obj>$ denote a FSSP, where n jobs $J = \{J_1, J_2, \ldots, Jn\}$ are to be processed on m machines $M = \{M_1, M_2, \ldots, M_m\}$, P indicates that only permutation schedules are considered and Obj is an objective function, describing the performance measure by which the schedule is to be evaluated. For instance, $<n, m, P, C_{max}>$ and $<n, m, P, F>$ are two FSSPs that minimize the *makespan* $C_{\max}$ and minimize the total flow time F, respectively.

In *FSSP*, each job J_i is passed on to m machines sequentially, following the ordering $M_1, M_2, \cdots, M_m$, so as to execute m different operations on these machines. In other words, in order to run the r-th operation for the job J_i, we forward J_i to the r-th machine M_r and then perform task on M_r with fixed processing time $T(r,i)$, $1 \le r \le m$ and $1 \le i \le k$. Notice that all the jobs are processed according to the order of a pre-defined schedule, which uniquely represents a permutation of jobs. Moreover, at any time, one machine is only allowed to process less than one job and also one job is only allowed to perform on less than one machine. The maximum completion time of the permutation schedule is given by

$$\begin{aligned} C(1,1) &= T(1,1) \\ C(1,i) &= C(1,i-1) + T(1,i) \\ C(r,1) &= C(r-1,1) + T(r,1) \\ C(r,i) &= \max(C(r,i-1), C(r-1,i)) + T(r,i) \end{aligned} \quad (1)$$

Where $1 \le r \le m$, $1 \le i \le k$, $T(r,i)$ stands for the execution time of the r-th operation of the i-th job J_i on the machine M_r, $C(r,i)$ denotes the maximum running time that J_i requires on M_r. And $C(m, n)$ represents *makespan*.

Each job has a specified processing order through all the machines with the corresponding processing time on each machine. This order is called machine sequence. The scheduling problem is to find out the best operation sequences on all machines in order to minimize the *makespan*. In this case, the *makespan* implies the criterion to be optimized.

3. DEPICT OF PARTICLE SWARM OPTIMIZER AND COOPERATIVE CO-EVOLUTION THEORY

3.1 Overview of PSO

Particle Swarm Optimization (PSO) is an evolutionary computation technique proposed by Kennedy and Eberhart [26] in the mid 1990s. Different from other algorithms, PSO is simple and easy to implement because no operators such as crossover and mutation exist. It was enlightened by the natural biologic phenomenon that a flock of birds attempt to find food through its own position as well as experience gained from others. More specifically, PSO is such an evolutionary computation technique that it works based on individual improvement plus population cooperation and competition. PSO regards the population and each individual in the population as swarm and particle, respectively. Regarding the status of a particle over the search space, it is generally characterized with its position and velocity, which are adjusted according to the flying experience of the particle as well as of its companions. Let $X_i = (x_{i1}, x_{i2}, \cdots, x_{id})$ and $V_i = (v_{i1}, v_{i2}, \cdots, v_{id})$, respectively, denote the position and the velocity of the i-th particle in an d-dimensional search space. Also, let $P_i = (p_{i1}, p_{i2}, \cdots, p_{id})$ and $P_g = (p_{g1}, p_{g2}, \cdots, p_{gd})$, respectively, stand for the best previously visited position of the i-th particle and the best individual of the whole swarm. The fitness value of each particle is evaluated according to the objective function. During the iterations, the velocity and position are repeatedly updated based on.

$$v_{id}(k+1) = v_{id}(k) + c_1 r_1 (p_{id}(k) - x_{id}(k)) + c_2 r_2 (p_{gd}(k) - x_{id}(k)) \quad (2)$$

$$x_{id}(k+1)=x_{id}(k)+v_{id}(k+1),(i=1,2,\cdots,m,d=1,2,\cdots,d) \quad (3)$$

where k is the iteration number, the variables c_1, c_2 are two learning factors, usually $c_1 = c_2 = 2$, defining the moving range for a particle and r_1, r_2 are two numbers randomly taken from the uniform distribution with the support (0, 1), that is, $r_1 \sim U(0,1)$ and $r_2 \sim U(0,1)$.

3.2 Principle of Cooperative Co-evolution Theory

Co-evolution mechanism which was first introduced by German mycologist, Anton de Bary in 1879 and is also referred to as symbiosis includes three main categories: mutualism (both species benefit by the relationship), commensalism (one species benefits while the other species is not affected), and parasitism (one species benefits and the other is harmed) [27]. Rong-Hwa Huang etc in paper [28] researched on the flow shop with multiprocessor scheduling problem (FSMP), and develops an improved particle swarm optimization heuristic to solve it. Additionally, designs an integer programming model to perform effectiveness and robustness testing on the proposed heuristic.

By contrast, the latter as for cooperative co-evolution, in natural ecosystems, almost all species own appetence to interact with other species to improve the survival cooperatively.

Consider a population with M particles, each of which is represented by an n-dimensional vector. After dividing each vector into λ parts S_j ($j=1,\cdots,\lambda$), then we obtain an ecosystem with λ sub-swarms $\{A_1, A_2, \cdots, A_\lambda\}$. Assume that H_j and H_{jg} are the current position and the previously best position of the sub-swarm A_j respectively and also that $S_j x_i$, $S_j p_i$ and $S_j p_g$ are respectively the i-th particle's current position, i-th particle's previously best position of parts S_j of sub-swarm A_j and the previously best position of sub-swarm A_j. According to the cooperative method, H_{ji} ($S_1 p_g, \cdots, S_{j-1} p_g, S_j x_i, S_{j+1} p_g, \cdots, S_\lambda p_g$) represents a new complete vector of each particle of sub-swarm A_j. At the same time, it reflects the cooperative method.

The best position of every sub-swarm of each particle is updated based on

$$\begin{cases} H_{ji}(S_1 p_g, \cdots, S_{j-1} p_g, S_j x_i, S_{j+1} p_g, \cdots, S_\lambda p_g), \\ f(H_{ji}(S_1 p_g, \cdots, S_{j-1} p_g, S_j x_i, S_{j+1} p_g, \cdots, S_\lambda p_g) \le L \\ H_{ji}(S_1 p_g, \cdots, S_{j-1} p_g, S_j p_i, S_{j+1} p_g, \cdots, S_\lambda p_g), \\ f(H_{ji}(S_1 p_g, \cdots, S_{j-1} p_g, S_j x_i, S_{j+1} p_g, \cdots, S_\lambda p_g) \ge L \end{cases} \quad (4)$$

Where,

$$F = f(H_{ji}(S_1 p_g, \cdots, S_{j-1} p_g, S_j p_i, S_{j+1} p_g, \cdots, S_\lambda p_g).$$

By contrast, using

$$H_{jg} = \mathrm{argmin} f(H_{ji}(S_1 p_g, \cdots, S_{j-1} p_g, S_j p_i, S_{j+1} p_g, \cdots, S_\lambda p_g)) \\ ,1 \le j \le \lambda, 1 \le i \le s \quad (5)$$

The best position of every sub-swarm can be found..

4. THE PROPOSED COOPERATIVE CO-EVOLUTION PARTICLE SWARM OPTIMIZER

4.1 Introduction to sharing scheme

Sharing scheme is a widely used niche technique which modifies fitness landscape by reducing the payoff in densely populated regions [29]. Enrico Sciubba and Federico Zullo in paper [30] consider a set of species feeding on the same energy resources. The balance equations for the allocation of such resources among the species result in a set of non linear differential equations describing the dynamics of each population. The paper address the important question of optimal exploitation of the incoming energy resource at the species- and ecological niche level: more specifically, after a formal definition of the energy effectiveness of the conversion for the overall system and for each species. For each individual, its fitness value is modified associated with other individuals using the sharing function.

4.2 Niche sharing scheme in NCPSO algorithm

Niche-based sharing scheme refers to that the fitness value of each individual in a population is adjusted according to sharing function, so at to reflect analogical degree between one individual and another. After adjustment, the scheme proceeds by choosing the adjusted fitness value, ensuring the diversity of population during the evolution process. The afore-mentioned sharing function defines analogical degree between two individuals in the form of scientific value, denoted as $S(d(x_i, x_j))$. Here, $d(x_i, x_j)$ refers to a certain relationship between two individuals x_i and x_j. The bigger the value of sharing function is, the more analogical the individuals in the population are. The sharing function $S(d(x_i, x_j))$ is given by:

$$S(d(x_i,x_j)) = \begin{cases} 1-\dfrac{d_1(x_i,x_j)}{\varepsilon_1} & d_1(x_i,x_j)<\varepsilon_1, d_2(x_i,x_j)\ge\varepsilon_2 \\ 1-\dfrac{d_2(x_i,x_j)}{\varepsilon_2} & d_1(x_i,x_j)\ge\varepsilon_1, d_2(x_i,x_j)<\varepsilon_2 \\ 1-\dfrac{d_1(x_i,x_j)d_2(x_i,x_j)}{\varepsilon_1\varepsilon_2} & d_1(x_i,x_j)<\varepsilon_1, d_2(x_i,x_j)<\varepsilon_2 \\ 0 & others \end{cases} \quad (6)$$

where $d_1(x_i, x_j)$ denotes the Euclidean distance between the encodings of the individuals x_i and x_j and $d_2(x_i, x_j)$ stands for the fitness distance between x_i and x_j.

In order to measure the analogical degree of an individual in a population, sharing degree, denoted as S_i , is defined. In this paper, we calculate S_i by summing up all the sharing function values between the individual and others. Mathematically, S_i is defined as follows:

$$S_i = \sum_{j=1, j\neq i}^{M} S(d(x_i, x_j)), i = 1, 2, \cdots, M \tag{7}$$

where M represents the population size.
Finally, using

$$f_i^{'} = \frac{f_i}{S_i}, i = 1, 2, \cdots, M \tag{8}$$

We can obtain the new fitness values of all individuals.
The principal idea of the niche-based sharing scheme for finding the optimal solution is to maximize the diversity within a population through adjusting the fitness values of all.

4.3 Pseudo-code of NCPSO Algorithm

The pseudo-code of NCPSO algorithm - is given as follows:

Begin
i =1 // the current generation
Initialize(pop) //generate initial population
Generate $pop0$($pop0$=pop) // this population as the niche evolution population
F($pop0$) //calculate fitness value
$F0^1_{best}$ and $p0^1_{best}$ =find(F($pop0$)) //find the best solution and the individual
Generate Sub-swarm1, Sub-swarm2, Sub-swarm3. //the process is shown in *figure 1*
F(Sub-swarm1),F(Sub-swarm2), F(Sub-swarm3) //calculate fitness values of every Sub-swarm
$F1^1_{best}$ and $Sub1^1_{best}$ =find(F(Sub-swarm1)) //with the same way $F2^1_{best}$ and $Sub2^1_{best}$, $F3^1_{best}$ and $Sub3^1_{best}$ are obtained.
i=2
While i<MAX_GEN
Begin $pop0$ //niche evolution environment
F'=N(F) //adjust the original fitness values
//N() is the function of niche based on sharing mechanism
$pop0_{current\_gen}$ =Genetic Operation($pop0$) //conduct Genetic Operation via F' and generate new population
$F0^i_{best}$ and $p0^i_{best}$ =find(F($pop0^i$))
If $F0^i_{best} < F0^1_{best}$
$p0^1_{best} = p0^i_{best}$
End
End $pop0$
Begin Sub-swarm1
Generate Sub-swarm1^i //the process is shown in *figure 1*
Sub1 =PS(Sub-swarm1^i)
//conduct particle swarm's updating equation with *equation(2),(3)*
$F1^i_{best}$ and $Sub1^i_{best}$ =find(F(Sub1))
If $F1^i_{best} < F1^1_{best}$
$Sub1^1_{best} = Sub1^i_{best}$
End
If $F0^i_{best} < F1^i_{best}$
$Sub1^1_{best} = p0^i_{best}$
End
End Sub-swarm1
Begin Sub-swarm2
Generate Sub-swarm2^i //the process is shown in *figure 1*
Sub2 =PSO(Sub-swarm2^i)
//conduct particle swarm's updating equation with *equation(2),(3)*
$F2^i_{best}$ and $Sub2^i_{best}$ =find(F(Sub2))
If $F2^i_{best} < F2^1_{best}$
$Sub2^1_{best} = Sub2^i_{best}$
End
If $F0^i_{best} < F2^i_{best}$
$Sub2^1_{best} = p0^i_{best}$
End
End Sub-swarm2
Begin Sub-swarm3
Generate Sub-swarm3^i //the process is shown in *figure 1*
Sub3 =PSO(Sub-swarm3^i)
//conduct particle swarm's updating equation with *equation(2),(3)*
$F3^i_{best}$ and $Sub3^i_{best}$ =find(F(Sub3))
If $F3^i_{best} < F3^1_{best}$
$Sub3^1_{best} = Sub3^i_{best}$
End
If $F0^i_{best} < F3^i_{best}$
$Sub3^1_{best} = p0^i_{best}$
End
End Sub-swarm3
S^i =Min($F0^i_{best}$, $F1^i_{best}$, $F2^i_{best}$, $F3^i_{best}$) //find the best solution
i=i+1
End
End

5. NCPSO ALGORITHM FOR FSSP

5.1 Encoding

Thus far, a large number of optimization algorithms have been employed to solve the continuous problem. However, FSSP is a combinatorial problem with solution being in discrete space, so initial schemes are not applicable in the case of FSSP. This, consequently, require us to propose an appropriate representation for FSSP.

Till now, various encoding methods have been proposed, based on permutation, job and precedence etc. In this paper, we choose the operation permutation-based encoding method.

Considering a FSSP with n jobs working on m machines, a n dimensional space is taken as the search space and the position of each particle is represented as a vector with n (real) components. In order to be consistent with the operation permutation sequence of FSSP, we first convert the n components in each vector into n integers, ranging from 1 to n according to a sort program. Each integer here represents the name of a job. For a better understanding of the proposed scheme, we here give an example. Assume that 5 machines are scheduled to process 10 jobs. First, initial solution with real numbers is generated randomly, Xi = [0.7373, 0.4799, 0.7806, 0.9984, 0.1751, 0.9657, 0.8703, 0.5454, 0.3095, 0.1750], and then it is encoded into a set of integers(10, 5, 9, 2, 8, 1, 3, 7, 6, 4) by sorting the 10 real numbers in Xi in ascending order. More specifically, 0.1750 is the smallest number among the ten float numbers, so it is ranked 1 with the initial order 10. In the same way, the rank values of the remaining numbers are assigned.

We have conducted the experiments on the benchmark problems proposed by Taillard (1993), with m = 5,10,20 and n = 20,50,100,200. There are 10 instances for each problem size and 110 problem instances in total.

5.2 Performance evaluation for NCPSO algorithm optimizing FSSP

Using eight flow shop instances with different sizes, we in this section compare NCPSO to PSO, GA and CPSO. The metric measure used for comparison is defined as the average relative percentage deviation (ARD) in *makespan* with respect to the best known solutions by Taillard and is given by

$$ARD = \sum_{i=1}^{R} \left(\frac{C_i - C_{best}}{C_{best}}\right) / R \qquad (9)$$

Where R the running time is C_i denotes the *makespan* obtained for the i -th running and C_{best} is the known minimum *makespan* for the problem or the lowest known upper bound for Taillard's instances.

Through equation (9), the value of ARD is associated with the difference between the solution searched by algorithm and the best solution. The smaller the ARD is, the more efficient the algorithm is. From Table 1, we observe that, for all, Taillard's instances, the ARD of solutions found by NCPSO algorithm are consistently smaller than those resulting from PSO, GA and CPSO. Therefore, NCPSO algorithm is more robust and efficient than others.

5.3 Experimental results

To demonstrate the performance of NCPSO in flow shop, we have used eight instances with different sizes (Taillard, 1990) that were selected from practical data in the experiments. Regarding the parameter setting, we have used R=10, the population size, abbreviated as *PS*, *PS* = 50 and the maximum generation *GEN* = 800 for the proposed method. The values of parameters for GA and NCPSO are P_c =0.8, P_m =0.05. Also, the learning factors $c_1 = c_2 = 2$ and the weight parameter w =0.6 have been used for PSO, CPSO and NCPSO. Table 2 lists the experimental results, and from this table we have noticed that it is particularly challenging to deal with some instances using simple methods.

In Table 2, the best solutions found by NCPSO are highlighted using bold face and the solutions that are comparable to OS are highlighted in italic. After comparing the data in this table, it is easy to see that the proposed NCPSO algorithm outperforms PSO and CPSO. For the instances with size 5×20, 5×50 and 5×100, the NCPSO algorithm finds not only the better solutions, but also the optimal solutions in most cases. This clearly indicates that NCPSO works quite well for the problems with m=5.

Notice that, as Figure 2 shows, the solutions by genetic algorithm (GA) are consistently worse than those by PSO, CPSO and NCPSO algorithm, so we do not list its solutions in table 2, taking into account space saving.

Figure 2 compares the solution searching process of different algorithms for eight scheduling problems with distinct sizes. As demonstrated by these figures, GA obviously achieves inferior performance that other algorithms. In terms of convergence speed, our method converges quicker than PSO, GA and CPSO. Although the figures for ta001b0, ta033b0 and ta063b0 show that CPSO algorithm is able to find the optimal solutions, our NCPSO method requires less running time. Moreover, we can see, from the figures for ta051b0 and ta071b0, that NCPSO provides more powerful capability to overcome the premature issue.

6. CONCLUSION

As it is well known, intelligent algorithm plays a significant part in both continuous and discrete optimization problems. We have proposed a cooperative co-evolution intelligent algorithm to solve flow shop scheduling problem, based on niche technology. Eight typical flow shop scheduling instances with forty problems have been used to evaluate the performance of our NCPSO algorithm in searching the best solution. As demonstrated by extensive experimental results, the NCPSO algorithm, when used to cope with flow shop scheduling problem, is more efficient and effective than prior methods. . Compared to PSO, GA and CPSO algorithms, the proposed algorithm achieves improved convergence speed and also is capable of finding the better solutions. As a direction for future research, we employ sharing scheme of niche and crowding scheme in different manners and meanwhile take into consideration some other strategies to advance the algorithm for improved performance. Another future work of ours is to apply the proposed algorithm to more complex scheduling problems, such as multi-objectives and multi-scheduling problems with uncertainty.

7. ACKNOWLEDGEMENT

This work was supported in part by Shanghai Municipal Science and Technology Commission (10JC1405800), Key Discipline of Shanghai Municipal Education Commission (J51901), the Shanghai Natural Science Foundation (14ZR1417300) and Climbing Peak Discipline Project of Shanghai Dianji University (15DFXK01).

8. REFERENCES

[1] Garey MRD, Johnson DS, Sethi R. The complexity of flowshop and job shop scheduling. Mathematics of Operations Research 1976;1:117–29.

[2] Johnson SM. Optimal two- and three-stage production schedules with setup times included. Novel Research Logistics Quarterly 1954;1:61–8.

[3] Palmer DS. Sequencing jobs through a multistage process in the minimum total time: a quick method of obtaining a near optimum. Operations Research Quarterly 1965;16:101–7.

[4] Campbell HG, Dudek RA, Smith ML. A heuristic algorithm for the n-job, m-machine problem. Management Science 1970;16:B630–7.

[5] Gupta JND. A functional heuristic algorithm for the flow-shop scheduling problem. Operations Research 1971;22:39–47.

[6] Nawaz M, Enscore Jr E, Ham I. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega—International Journal of Management Science 1983;11:91–5.

[7] Grabowski J, Wodecki M. A very fast tabu search algorithm for the permutation flowshop problem with makespan criterion. Computers and Operations Research 2004;31:1891–909.

[8] Ogbu FA, Smith DK. The application of the simulated annealing algorithm to the solution of the n/m/Cmax flowshop problem. Comput Operations Research 1990;17:243–53.

[9] Wenbin Xie, Chris Hicks and Pupong Pongcharoen. A Multiple Criteria Genetic Algorithm Scheduling Tool for Production Scheduling in the Capital Goods Industry. International Journal of Engineering and Technology Innovation, vol. 4, no. 1, 2014, pp. 18-29.

[10] Biesinger, B; Hu, B; Raidl, G. A hybrid genetic algorithm with solution archive for the discrete -centroid problem. OURNAL OF HEURISTICS. Vol: 21: 3, JUN 2015. Page: 391-431.DO I: 10.1007/s10732-015-9282-5.

[11] Rajendran C, Ziegler H. Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. European Journal of Operational Research 2004;155:426–38.

[12] I-Hong Kuo, Shi-Jinn Horng, Tzong-Wann Kao, Tsung-Lieh Lin, Cheng-Ling Lee, Takao Terano and Yi Pan, An efficient flow-shop scheduling algorithm based on a hybrid particle swarm optimization model. expert Systems with Applications 36 (2009) 7027–7032.

[13] Lian, Z., Gu, X., & Jiao, B. (2008). A novel particle swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan. Chaos, Solitons and Fractals, 35, 851–861.

[14] Chansareewittaya, S; Jirapong, P. Power transfer capability enhancement with multitype FACTS controllers using hybrid particle swarm optimization. ELECTRICAL ENGINEERING. JUN 2015 Vol: 97: 2.Page: 119-127. DOI: 10.1007/s00202-014-0317-y.

[15] Yi Zhang, Xiaoping Li et al.Hybrid genetic algorithm for permutation flowshop scheduling problems with total flowtime minimization,European Journal of Operational Research 196 (2009) 869–876.

[16] G.I. Zobolas, C.D. Tarantilis, and G. Ioannou, Minimizing makespan in permutation flow shop scheduling problems using a hybrid metaheuristic algorithm, Computers & Operations Research 36 (2009) 1249–1267.

[17] Li, JQ; Pan, QK. Solving the large-scale hybrid flow shop scheduling problem with limited buffers by a hybrid artificial bee colony algorithm. INFORMATION SCIENCES. SEP,20 2015: Vol. 316 P: 487-502.DOI: 10.1016/j.ins.2014.10.009.

[18] Changsheng Zhang, Jiaxu Ning et al,A hybrid alternate two phases particle swarm optimization algorithm for flow shop scheduling problem,Computers & Industrial Engineering 58 (2010) 1–11.

[19] Jindong Zhang, Changsheng Zhang et al.The circular discrete particle swarm optimization algorithm for flow shop scheduling problem,Expert Systems with Applications (2010) 1–8.

[20] Jun Zhang, De-Shuang Huang A novel adaptive sequential niche technique for multimodal function optimization. Neurocomputing 69 (2006) 2396–2401.

[21] Xiyu Liu, Hong Liu Particle swarm optimization based on dynamic niche technology with applications to conceptual design. Advances in Engineering Software 38 (2007) 668–676.

[22] T. Radha Ramanan, Muhammed Iqbal and K. Umarali. A particle swarm optimization approach for permutation flow shop scheduling problem. Int. J. Simul. Multisci. Des. Optim. 2014, 5, A2.

[23] Shuzhi Gao, Liangliang Luan and Xianwen Gao. Chlor-alkali industry production scheduling algorithm research based on adaptive weight PSO. Journal of Chemical and Pharmaceutical Research, 2014, 6(5):34-41.

[24] Gonzalez, MA; Vela, CR; Varela, R. Scatter search with path relinking for the flexible job shop scheduling problem. EUROPEAN JOURNAL OF OPERATIONAL RESEARCH. Vol. 245:Page: 35-45.DOI: 10.1016/j.ejor.2015.02.052.

[25] Lei, DM; Guo, XP. A parallel neighborhood search for order acceptance and scheduling in flow shop environment. INTERNATIONAL JOURNAL OF PRODUCTION ECONOMICS. Vol: 165,page: 12-18.DOI: 10.1016/j.ijpe.2015.03.013.

[26] J Kennedy, R Eberhart.: Particle Swarm Optimization[C].In: Proc IEEE Int Conf on Neural Network(1995):1942-1948.

[27] A.E. Douglas, Symbiotic Interactions, Oxford University Press, Oxford, 1994.

[28] Rong-Hwa Huang, Chang-Lin Yang and Chun-Ting Hsu. Multi-objective two-stage multiprocessor flow. Shopscheduling-a subgroup particle swarm optimisation approach. International Journal of Systems Science. DOI:10.1080/00207721.2014.886742.

[29] B. Sareni, Krahenbuhl L, Fitness sharing and niching methods revisited, IEEE Trans. Evolut. Comput. 2 (3) (1998) 97–106.

[30] Enrico Sciubba and Federico Zullo. An exergy-based analysis of the co-evolution of different species sharing common resources. Ecological Modelling Volume 273, 10 February 2014, Pages 277–283.

9. APPENDIX

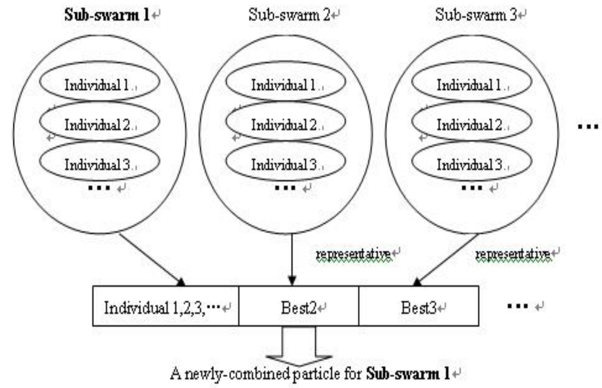
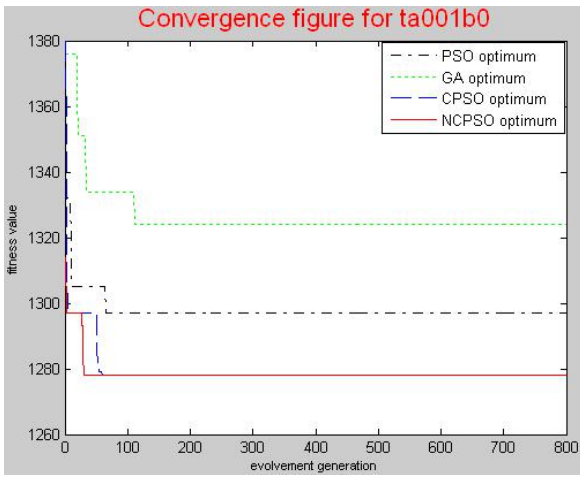
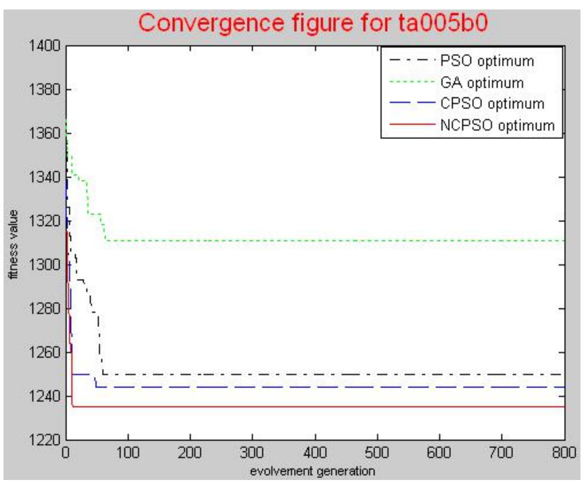
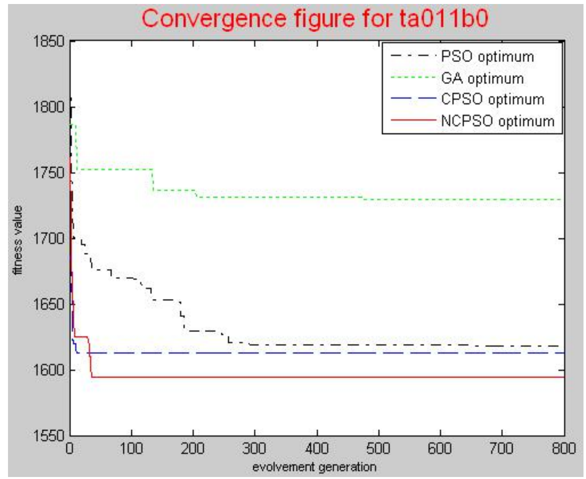
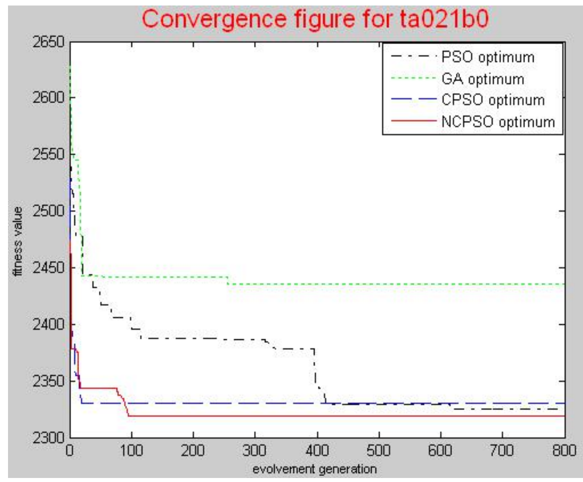


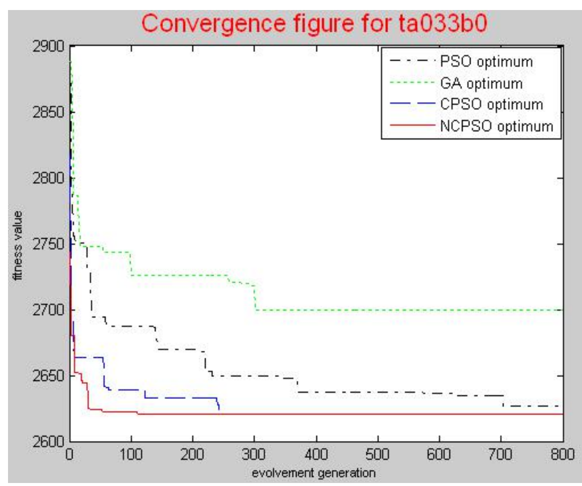
Fig 1. An example for the cooperative method

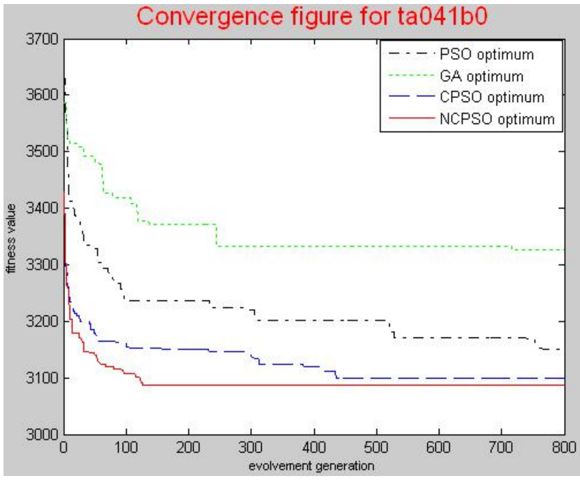


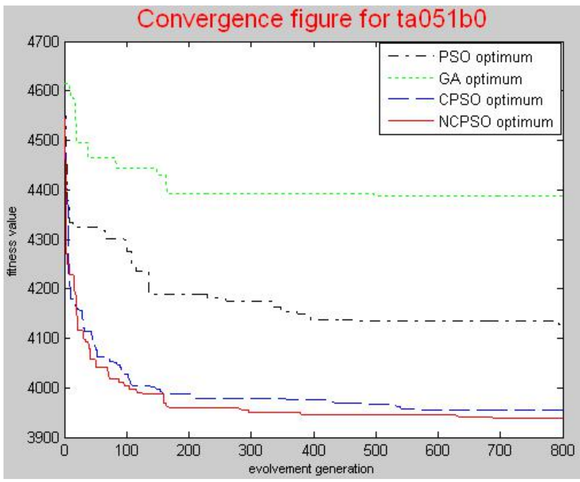


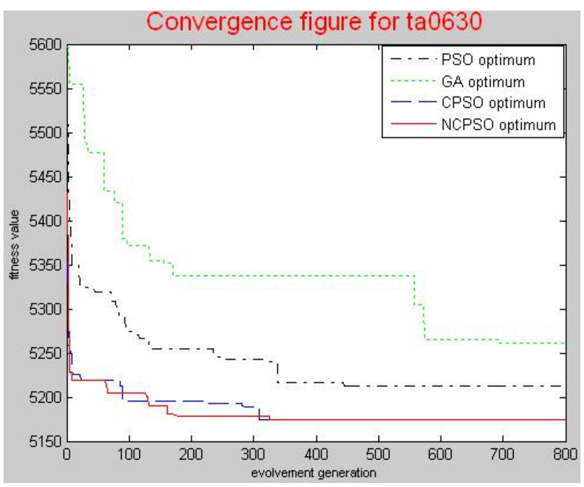


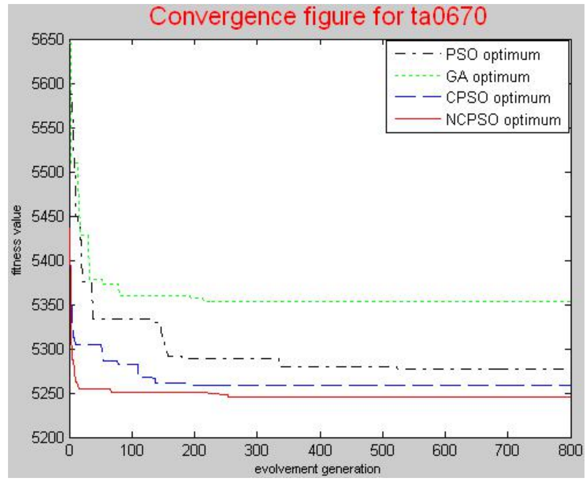












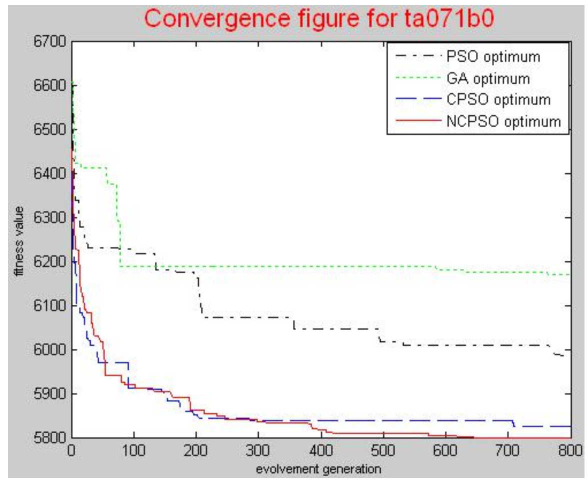


Fig. 2. The optimal curves of PSO, GA, CPSO and NCPSO algorithms for FSSP

TABLE I
COMPARISONS BETWEEN NCPSO, PSO AND GA IN THE AVERAGE RELATIVE DEVIATION

Taillard instance	SIZE	ARD_{PSO}	ARD_{G}	ARD_{CP}	ARD_{NC}
ta001b0	5×20	1.205	3.6933	1.4867	**0.939**
ta011b0	10×20	2.4399	10.3034	2.3009	**1.9848**
ta021b0	20×20	2.4902	7.0091	2.6295	**2.377**
ta031b0	5×50	0.2937	2.7753	0.2423	**0.1542**
ta041b0	10×50	6.894	13.1795	4.4801	**4.1057**
ta051b0	20×50	8.3584	14.5351	3.6519	**3.4805**
ta061b0	5×100	0.6153	1.7987	0.0073	**0**
ta071b0	10×100	4.4714	8.8873	1.7608	**1.227**

TABEL II

COMPARISON RESULTS OF THE PSO, CPSO AND NCPSO ALGORITHMS

Taillard Problem	size	OS	PSO			CPSO			NCPSO		
			min	**average**	**std.**	**min**	**average**	**std.**	**min**	**average**	**std.**
ta001b0		1278	1297	1297	0	***1278***	1293.2	10.139	***1278***	1289.6	8.4971
ta003b0		1081	***1081***	1109.8	23.5839	***1081***	1092	9.0277	***1081***	1083.6	3.5777
ta005b0	5×20	1235	1250	1250	0	1244	1247.6	3.2863	***1235***	1247	6.7082
ta007b0		1239	**1251**	1253.6	3.7148	**1251**	1255	3.8079	**1251**	1251	0
ta009b0		1230	1236	1248.8	7.3621	***1230***	1246.6	13.5167	***1230***	1244.8	13.0307
ta011b0		1582	1618	1620.6	4.219	1613	1618.4	5.1769	**1594**	1613.4	15.0433
ta013b0		1496	1522	1534	7.6485	1520	1538.2	6.5422	**1517**	1528.6	9.8731
ta015b0	10×20	1419	1455	1463	8.9722	1433	1449.6	14.0996	**1426**	1442.6	12.5419
ta017b0		1484	1493	1509.6	13.6308	1496	1516.8	22.9808	**1486**	1509	12.2219
ta019b0		1593	1625	1633.6	5.1284	1620	1631.8	11.0544	**1617**	1623.6	7.3007
ta021b0		2297	2325	2354.2	21.5801	2330	2357.4	23.8181	**2319**	2351.6	25.9191
ta023b0		2326	2366	2387.6	18.7697	2343	2384.6	24.8254	**2340**	2373	22.6826
ta025b0	20×20	2291	2325	2338.8	12.0706	2319	2343.4	21.3846	**2314**	2331.4	15.5981
ta027b0		2273	2317	2329.8	10.2078	2310	2325.4	19.4499	**2292**	2319.6	9.3117
ta029b0		2237	2275	2304.4	23.1905	2287	2312.4	20.7075	**2268**	2288.2	15.1063
ta031b0		2724	2729	2732	5.6125	***2724***	2730.6	8.9426	***2724***	2728.2	6.3689
ta033b0		2621	2624	2642.4	16.5015	***2621***	2625.2	2.7749	***2621***	2622.6	1.3416
ta035b0	5×50	2863	2864	2880.6	12.9923	***2863***	2863.6	0.5477	***2863***	2863.2	0.4472
ta037b0		2725	2736	2756.6	18.3521	***2725***	2732.8	6.5803	***2725***	2732.4	12.2556
ta039b0		2552	2583	2592.4	11.9917	2564	2567.2	6.6106	**2554**	2559	4.5277
ta041b0		2991	3150	3197.2	34.2885	3100	3125	16.6733	**3086**	3113.8	20.4377
ta043b0		2839	3017	3054.6	21.7555	2937	2959.4	13.3154	**2907**	2945.4	24.5214
ta045b0	10×50	2976	3137	3174.6	29.2882	3055	3085.4	24.5723	**3048**	3066.4	17.3292
ta047b0		3093	3206	3261.4	49.3994	3144	3193.6	36.2119	**3132**	3170.6	25.8902
ta049b0		2897	3040	3085.2	35.8008	2931	2983.4	43.0209	**2925**	2968.8	24.9439
ta051b0		3850	4128	4171.8	35.8845	3956	3990.6	34.1151	**3939**	3984	26.4144
ta053b0		3640	3892	3927	22.3047	3774	3813.4	22.2441	**3768**	3803.6	17.8253
ta055b0	20×50	3610	3878	3958.2	69.5572	3737	3778.4	29.6951	**3726**	3766.2	39.99
ta057b0		3704	3986	4028	43.5833	3828	3864.2	24.6011	**3816**	3857	18.1364
ta059b0		3743	3948	3984.6	21.1967	3941	3977.6	28.6147	**3909**	3940.8	30.9063
ta061b0		5493	5495	5526.8	24.8032	***5493***	5493.4	0.8944	***5493***	5493	0
ta063b0		5175	5212	5224.8	15.2381	***5175***	5190.6	13.6675	***5175***	5188.6	16.8908
ta065b0	5×100	5250	5255	5277.6	22.4678	5255	5255	0	***5250***	5253.6	2.1909
ta067b0		5246	5277	5293	14.6458	5259	5260.4	1.5166	***5246***	5257.8	6.7231
ta069b0		5448	5488	5501.2	12.518	5454	5461.2	11.6106	***5448***	5458.8	5.6526
ta071b0		5770	5986	6028	24.8898	5826	5871.6	38.869	**5800**	5840.8	40.6719
ta073b0		5676	5843	5889.8	34.7448	5722	5759.8	37.2116	**5679**	5729	37.0338
ta075b0	10×100	5467	5752	5796.8	33.922	5539	5578	31.8892	**5535**	5566.6	20.1531
ta077b0		5595	5798	5821.6	17.0529	5654	5680.6	16.2033	**5628**	5667.2	12.3662
ta079b0		5871	6055	6103	46.114	5971	5980	8.124	**5928**	5970	23.622