

A Novel Particle Swarm Optimization Algorithm for Short-Term Scheduling of Batch Plants with Parallel Units

Jin Zhu¹, Xingsheng Gu¹ and Bin Jiao²

¹ *Research Institute of Automation, East China University of Science and Technology, Shanghai, 200237, China zhjtan@126.com, xsgu@ecust.edu.cn*

² *Electrical Engineering Department, Shanghai DianJi University, Shanghai, 200240, China binjiaocn@163.com*

Abstract: This paper proposed a novel particle swarm optimization (NPSO) algorithm and applied it to short-term scheduling of single-stage batch plants with parallel units to minimization of earliness. The model is formulated as a mixed-integer linear programming (MILP) problem using the continuous-time domain representation. PSO is an optimization technique in real-number spaces, however scheduling of batch plants is a problem in discrete space. So, the improvement of NPSO includes introducing GAs operators for generating particle's flying velocity and position, and some heuristic rules for generating better initialization population. They all have no effect on the optimality of the scheduling problem. Computational experiments show that NPSO are more clearly appropriate than standard GA for scheduling of batch plants with due date constraints, and NPSO becomes more effective after involving heuristic rules.

1. Introduction

Batch plants has been widely practiced since long before the development of the modern chemical industry [1,2]. It is suitable for manufacturing a relatively large number of low-volume, high-value-added products. In many chemical industries, such as pharmaceuticals, paints, and petrochemicals, batch mode has been used in many operations. The effective scheduling for batch plants can not only increase customer services, reduce the need for excess capacities of corporations but also provide the potential of identifying key data and mechanistic understandings of processes [3]. So, it is significant to do further research on scheduling for batch plants.

The problem of short-term scheduling of batch plants has as an objective to determine the optimal production plan for satisfying the production demands for different products at due dates and/or at the end of a given time horizon. It involves the allocation of equipment and resources to orders, the sequencing of these orders, and the determination of the material flows through the plant.

The problem of short-term scheduling of batch plants has developed a lot in the past ten years, mainly focus on the modeling of scheduling, such as various descriptions for batch process, the representation of 0-1 variable and time etc. On the basis of the notation of time slot, Karimi and McDonald [4] proposed two models for parallel semicontinuous processes considering the sequence-dependent setup times, orders, and their corresponding due dates in order to minimize the inventory. Ierapetritou and Floudas [5] built models for the batch, continuous, and semicontinuous processes represented in STN (State Task Network) framework. They also took the due dates of products and intermediates into consideration. Cerda et al. [6] developed a short-term scheduling model for the single-stage multiproduct batch plants with parallel lines to minimize makespan, total tardiness, or number of tardy orders. They also considered the positive release times of orders and ready times of units, as well as sequence-dependent setup times and forbidden processing subsequences. Méndez et al [7] reformulated this model by separately handling assignment and sequencing decisions through different sets of binary variables to get a reduction in the number of 0-1 variables. They also addressed a model to optimally select both batch sizes and number of batches of each size for every product to be processed along the production horizon according to the sizes of orders and plant capacity.

However, the scheduling of batch plants is one of the most difficult problems, as it is classified as an NP-complete one [8,9]. In many cases, the combination of goals and resources exponentially increases the search space, and thus the generation of consistently good scheduling is particularly difficult because we have a very large combinatorial search space and precedence constraints between operations. Exact methods such as the branch and bound method and dynamic programming take considerable

computing time if an optimum solution exists. In order to overcome this difficulty, it is more sensible to obtain a good solution near the optimal one. Heuristic optimization algorithms such as particle swarm optimization algorithms can be used to find a good solution.

Particle swarm optimization (PSO) is one of heuristic optimization algorithms. It was developed by Dr. Eberhart and Dr. Kennedy [10,11] in 1995, inspired by social behavior of bird flocking or fish schooling. Compared to GA, PSO is easy to implement and has few parameters to adjust. In recent years, a lot of reported works focused on PSO have been applied widely in the function optimization, artificial neural network (ANN) training, pattern recognition, fuzzy control and some other fields [12-15]. However, there is rare research on the application of PSO in short-term scheduling of batch plants, due to its encoding problem.

This paper proposes a novel particle swarm optimization (NPSO) algorithm and applies it to short-term scheduling of batch plants with Parallel Lines. Section 2 presents the single-stage mathematical model, including model assumptions and some mathematical symbols for the scheduling model. The model is formulated as a MILP problem using the continuous-time domain representation. Sections 3 propose some heuristic rules, crossover operators and mutation operators in NPSO algorithm, and then the iterative model are presented. In sections 4, the 4 algorithms, GA and NPSO algorithm with and without heuristic rules (HR) are compared to illustrate the effectiveness of the NPSO and heuristic rule. Results are compared with that of GAs. The contribution of this paper and conclusions are summarized in Section 5.

2. Mathematical model

Time representation is very important for a mathematical scheduling model. This is because the overall profile of resource utilization is discontinuous. The model has to track such discontinuities within the scheduling horizon; i.e., the profile is compared with the resource availabilities to ensure feasibilities. Discrete and continuous-time representations are two existing approaches to deal with such complexity. The main drawback of the model based on uniform discrete-time representation is that the time representation is approximate, and the model comprises a large number of binary variables and linear constraints when the problem of industrial size is considered. Hence, the continuous-time representation is widely used in research of batch process scheduling now.

The problem considered is the short-term scheduling for a set of orders with due dates in batch plants with one stage and units in parallel. It will be formulated as an MILP model in which the objective is the minimization of earliness. MILP model use the continuous-time domain representation.

The main assumptions of the model are:

- A1. The model parameters are all deterministic.
- A2. Transition times are only unit-dependent.
- A3. Batch sizes are fixed parameters.
- A4. Each order is to be processed only once by exactly one unit of one stage.
- A5. Each order represents one batch.
- A6. No resource constraints except unit are considered.

Assumption 1 seems quite reasonable. Relaxing assumption 2 would considerably complicate the model. The next three assumptions can be relaxed at the time of specifying the order requirements by breaking the demand for products, as described next. Assumption 5 is possible that there might exist several orders of the same product. Assumption 6 cannot be easily relaxed and will be considered explicitly in a future publication.

In the proposed formulation, the subscripts are listed below:

Indices

$i, i' =$ orders; $j =$ units; $n =$ event points;

$n_i, n_i' =$ event points assigned to orders i, i' , respectively.

Sets

$I =$ orders; $J =$ units;

$I_j =$ tasks which can be performed in unit j ;

$J_i =$ units which are suitable for performing task i ;

$N =$ event points within the time horizon;

$N_i =$ event points assigned to order i .

Parameters

$H =$ time horizon; $due(i) =$ due date of order i ;

$\alpha_{ij} =$ constant term of processing time of task i at unit j ;

$\beta_j =$ constant term of setup time in unit j ;

$yv(j, n) =$ binary variables that assign the utilization of unit j at event point n ;

$T^s(i, j, n) =$ time that task i starts in unit j at event point n ;

$T^f(i, j, n) =$ time that task i finishes in unit j while it starts at event point n .

On the basis of these notations, the mathematical model for the short-term scheduling of a single-stage batch plant without considering resource constraints involves the following constraints.

(1) Allocation Constraints

$$\sum_{j \in J_i} yv(j, n) = 1 \quad \forall i \in I, n \in N_i \quad (1)$$

These constraints express that an order only has to be processed once, since only one unit can be assigned to it.

(2) Duration Constraints

$$T^f(i, j, n) = T^s(i, j, n) + yv(j, n)(\alpha_{ij} + \beta_j) \quad \forall i \in I, j \in J_i, n \in N_i \quad (2)$$

These constraints ensure that the processing time of an order i in a particular machine j is the sum of the setup time

characteristic in the unit, β_j , and the processing time of this order in this particular unit, α_{ij} .

(3) Sequence Constraints: Different Tasks in the Same Unit

$$\begin{aligned} T^s(i', j, n_{i'}) &\geq T^f(i, j, n_i) \\ \forall i \in I, i' \in I, j \in J, n_{i'} > n_i, i \neq i' \end{aligned} \quad (3)$$

These constraints express the requirement that only one order at a time can be processed in a unit; the beginning of a later event must follow after the end of all earlier events in the same unit.

(4) Due Dates Constraints

$$T^f(i, j, n_i) \leq due(i) \quad \forall i \in I, \forall j \in J_i \quad (4)$$

Where n_i corresponds to the assignment of node n to order i . These constraints ensure that the order would be ready at its due date, $due(i)$.

(5) Objective: Minimization of Earliness

$$\text{Min} \sum_{i \in I, j \in J} \{due(i) - [T^s(i, j, n) + yv(j, n) \times (\alpha_{ij} + \beta_{ij})]\} \quad (5)$$

The objective function represents the minimization the time interval between the end of processing of the orders and their due dates.

3. The novel particle swarm optimization (NPSO) algorithm

PSO is an evolutionary algorithm that the system is initialized with a population (named swarm in PSO) of random solutions and searches for optima by updating generations. Each individual or potential solution, named particle, flies in the dimensional problem space with a velocity which is dynamically adjusted according to the flying experiences of its own and its colleagues. The PSO algorithm mimics the behavior of flying birds and their means of information exchange to solve optimization problems. PSO has been introduced as an optimization technique in real-number spaces. But many optimization problems are set in a discrete space. Typical examples include problems that require ordering, such as in scheduling of batch plants.

In this section, 5 heuristic rules are defined to generate a better initialization population to improve the quality of solution. And, GA operators are introduced to the proposed NPSO algorithm for generating particle's flying velocity and position in the application of PSO to short-term scheduling of batch plants.

3.1. Standard PSO algorithm

Suppose that the searching space is D-dimensional and m particles form the colony. The i th particle represents a D-dimensional vector $X_i (i=1, 2, \dots, m)$. It means that the i th particle positions at $X_i = (x_{i1}, x_{i2}, \dots, x_{iD}) (i=1, 2, \dots, m)$ in the searching space. The position of each particle is a potential

result. We could calculate the particle's fitness by putting its position into a designated objective function. When the fitness is higher, the corresponding X_i is "better". The i th particle's "flying" velocity is also a D-dimensional vector, denoted as $V_i = (v_{i1}, v_{i2}, \dots, v_{iD})$. Denote the best position of the i th particle as $P_i = (p_{i1}, p_{i2}, \dots, p_{iD})$, and the best position of the colony as $P_g = (p_{g1}, p_{g2}, \dots, p_{gD})$ respectively. The PSO algorithm could be performed by the following equations.

$$\begin{aligned} v_{id}(k+1) &= v_{id}(k) + c_1 r_1 (p_{id}(k) - x_{id}(k)) \\ &+ c_2 r_2 (p_{gd}(k) - x_{id}(k)) \end{aligned} \quad (6)$$

$$x_{id}(k+1) = x_{id}(k) + v_{id}(k+1) \quad (7)$$

Where k represents the iterative number, c_1, c_2 are learning factors, usually $c_1 = c_2 = 2$. r_1, r_2 are random numbers between (0, 1). The termination criterion for the iterations is determined according to whether the max generation or a designated value of the fitness of P_g is reached.

3.2. The NPSO algorithm for scheduling of batch plants

3.2.1 Heuristic rules. A feasible resolution of the scheduling problem is a process sequence of orders, which is regarded as a particle of initialization population. The countered particles based on the given particle are generated by 5 heuristic rules. Then, the initialization population is got from all feasible resolution of the scheduling problem. The generation process of the countered particles by heuristic rules is illustrated as follow.

Take the short-term scheduling of batch plants with 13 orders as example. Given a feasible resolution is the order sequence 11 8 12 2 13 9 3 6 1 10 7 5 4, and the due dates are 15 25 13 31 11 25 19 23 25 25 17 27 25 respectively. Firstly, rank orders according to the due dates, the order with earliest due date will be arranged first, so the order sequence becomes 13 12 11 7 3 6 8 9 1 10 4 5 2; secondly, the countered particles based on the given particle are generated by 5 heuristic rules.

Rule 1. A moving segment (MS) and one inserting point (IP) are selected randomly in the orders which the due dates are same or close. The orders inside (MS) are moved and inserted in the (IP). Thus, a new particle (NP) is generated. The definition of "close" is one-fifth of the amount of all the orders.

Rule 2. A moving segment (MS) and one inserting point (IP) are randomly selected along the length of the predecessor chromosome. The orders inside (MS) are moved and inserted in the (IP).

Rule 3. It is same to rule 2, while there are two moving segment and inserting point.

Rule 4. A reverse segment (RMS) is selected randomly in the orders which the due dates are same or close. The orders inside segment (RMS) are reversed.

Rule 5. Base on rule 4, orders inside segment (RMS) are reversed and inserted in the inserting point (IP) to generate a new particle. These rules are illustrated in Figure 1.

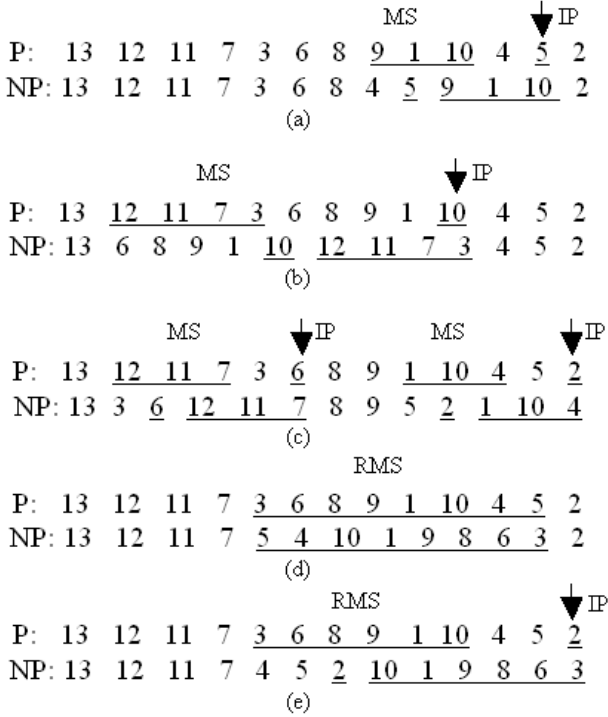


Figure 1 : Illustration of the 5 heuristic rules

It is noted that the heuristic rules are not to cut down the size of the model but to generate a better initialization population to improve the quality of solution. So, these heuristic rules have no effect on the optimality of the model.

3.2.2 Evolution operators

1) Crossover operators [16]

C1: A crossing segment (CS) is randomly selected along the length of the first predecessor chromosome (P1). The orders inside CS are copied into the offspring. The remaining places of the offspring are filled up by order in the second predecessor (P2), which is not occurred in CS.

C2, C3: Their processes are same to C1, while there are two and three moving segments are copied respectively. Figure 2(a)-(c) illustrates the operation of these crossover operators.

2) Mutation operators

Five mutation operators are designed. The idea of operator M1, M2 are exactly same to rule 2 and 3 respectively. M4 and M5 are similar to rule 4 and 5. The difference lies in that the scope of reverse order segment has no limitation. M3 represents that three moving segments and three insert points are randomly selected in predecessor chromosome. The orders inside three segments are moved and inserted in the inserting points respectively. M3 operator is illustrated in Figure 2(d).

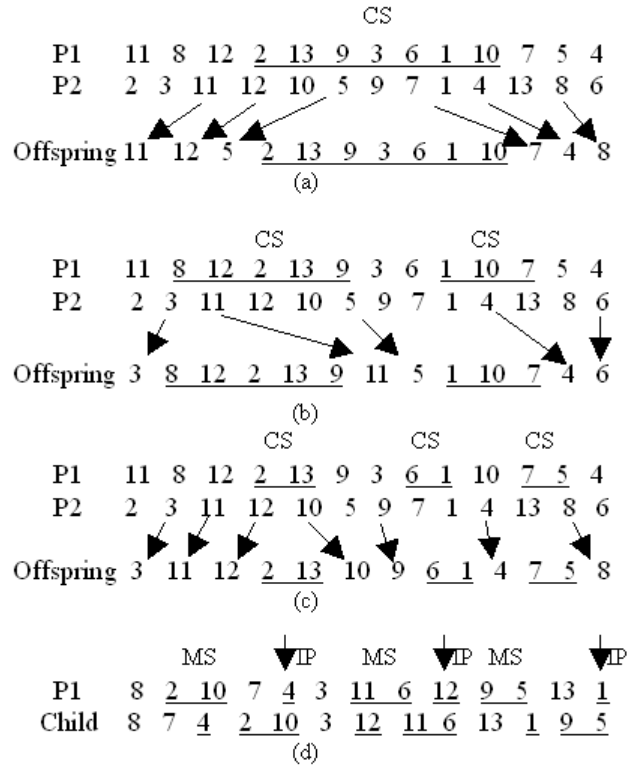


Figure 2 : Illustration of the various types of crossover, mutation operators

3.2.3 Iterative algorithm of NPSO. If the short-term scheduling of batch plants is n -orders and m -parallel units, suppose that the searching space is n -dimensional and S particles form the colony. The i th particle represents an n -dimensional vector $X_i (i=1,2,\dots,s)$. It means that the i th particle position at X_i , which is one sequence in the searching space. The position of each particle is a potential result. We could calculate the particle's fitness by putting its position into a designated objective function. When the fitness is lower, the corresponding X_i is "better". The i th particle's "flying" velocity is an n -dimensional vector, denoted as V_i . Denote the best position of the i th particle as P_i , and the best position of the colony as P_g . NPSO algorithm could be performed as:

$$v_i(k+1) = p_i(k) \oplus p_g(k) \tag{8}$$

$$(v_{r_1}, v_{r_2}, \dots, v_{r_N})(k+1) = M(v_{r_1}, v_{r_2}, \dots, v_{r_N}) \tag{9}$$

$$x_i(k+1) = x_i(k) \oplus v_i(k+1) \tag{10}$$

$$(x_{r_1}, x_{r_2}, \dots, x_{r_N})(k+1) = M(x_{r_1}, x_{r_2}, \dots, x_{r_N}) \tag{11}$$

Where k is the number of iterative generation. Symbol \oplus is the crossover denotation, which means two particles make crossover operation. $r_N (1 \leq r_N \leq psize)$ is a random integer which denote mutating particle. $psize$ is the number of initialization population. N is the sum of every generation

mutating particles. $M(v_{r_N}), M(x_{r_N})$ represents mutation operation of particle v_{r_N} and x_{r_N} . The termination criterion for the iterations is determined according to whether the max generation or a designated value of the fitness of P_g is reached.

3.2.4 Pseudo code of NPSO algorithm

Step1. Initialize parameters, including swarm size PS , maximum of generation G ;

Step2. Assignment and scheduling;

*Generate initialization population and velocity stochastically using heuristic rules;

*Evaluate each particle's fitness;

*Initialize $gbest$ position with the particle with the lowest fitness in the swarm;

*Initialize $pbest$ position with a copy of particle itself;

*current_gen:=0;

While (the maximum of generation is not met)

current_gen:= current_gen+1;

Generate nest swarm by equation (8)-(11);

Evaluate swarm {

Compute each particle's fitness in the swarm;

Find new $gbest$ and $pbest$ by comparison;

Update $gbest$ of the swarm and $pbest$ of particle;}

Step3. Output optimization results.

NPSO algorithm subprogram (for each particle s of swarm) as follow:

For (from the last order in a process sequence of orders move to the first order)

*the selection principle of the unit for order i is: to compare the absolute value of the difference of the due date of i and the beginning time of former order executed in the units that i can select. The unit with minimum difference is selected as the unit for i ;

*the finishing time of order i is: the minimum value between the due date of order i and the beginning time of the former order executed in the selected unit;

*the beginning time of order i is: the finishing time minus the executing time in the selected unit;

End

In NPSO, each particle of the swarm shares mutual information globally and benefits from the discoveries and previous experiences of all other colleagues during the search process. NPSO requires only primitive and simple mathematical operators, and is computationally inexpensive in terms of both memory requirements and time.

3.3. Comparisons between GAs and NPSO

We can learn that NPSO algorithm shares many common points with GA from their procedure. Both algorithms start with a group of a randomly generated population; both have fitness values to evaluate the population. The GA updates the population and search for the optimum with random techniques but the NPSO does not. Particles update themselves with the internal velocity and they also have

memory, which is important to the algorithm. Compared with genetic algorithms, the information sharing mechanism in NPSO is significantly different and operators are simple. In GAs, chromosomes share information with each other. So the whole population moves like a one group towards an optimal area. In NPSO, P_i , P_g and V_i gives out the information to others. It is a one-way information sharing mechanism and all the particles tend to converge to the best in the NPSO. Compared with GA, the advantages of NPSO are that it is easy to implement and there are few parameters to adjust.

4. Computational studies

In this section, NPSO was used to implement the model and the solution method. An example from Pinto and Grossmann [17] is considered to illustrate the effectiveness and performance of NPSO algorithm. Batch plant has one critical processing stage with four parallel units with unequal capacities and processing rates. There are a very large number of orders that need to be scheduled on very few units. Unit 1 and unit 4 can handle many orders, while unit 2 and unit 3 can only process a few. Data are shown in Table 1.

Table 1 : Data, Single State

Order	Due date(day)	Processing time (day)			
		Uint 1	Uint 2	Uint 3	Uint 4
1	15	1.538			1.194
2	30	1.500			0.789
3	22	1.607			0.818
4	25			1.564	2.143
5	20			0.736	1.017
6	30	5.263			3.200
7	21	4.865		3.025	3.214
8	26			1.500	1.440
9	30			1.869	2.459
10	29		1.282		
11	30		3.750		3.000
12	21		6.796	7.000	5.600
13	30	11.250			6.716
14	25	2.632			1.527
15	24	5.000			2.985
16	30	1.250			0.783
17	30	4.474			3.036
18	30		1.429		
19	13		3.130		2.687
20	19	2.424		1.074	1.600
21	30	7.317		3.614	
22	20			0.864	
23	12			3.624	
24	30			2.667	4.000
25	17	5.952		3.448	4.902
26	20	3.824			1.757
27	11	6.410			3.937
28	30	5.500			3.235
29	25				4.286
transition		0.180	0.175	0.000	0.237

Preordering was performed simultaneously for all units so that orders could be assigned to nodes, and the

combination of the node and order could be assigned an upper time limit. This is done because the assignment of nodes had to take place such that, in each unit, an order is coupled to the same node number. To avoid conflicts, the processing times of each order should have the same relative order in each machine. If conflicts do arise, the unit with the least possible orders should govern the node assignment decisions.

First, preordering is performed based only on the due dates of the orders. Then, the order in which each unit should handle orders with the same due date is determined. The processing times of orders in units are compared, but only if two orders are processed in the same unit. To keep the node assignment consistent, the sequencing of orders has to be the same in each unit; that is, the processing times must be of the same relative size in all units.

The orders are numbered 1-29, in ascending order, according to aforementioned sequencing. This number is then assigned to each order as its node number. The results are shown in Table 2.

Table 2 : Number Assignment

order	1	2	3	4	5	6	7	8	9	10
number	4	28	12	15	9	23	11	17	27	18
order	11	12	13	14	15	16	17	18	19	20
number	25	10	19	16	13	29	24	26	3	6
order	21	22	23	24	25	26	27	28	29	
number	20	8	2	21	5	7	1	22	14	

On the basis of preordering, the MILP model is applied for the cases where 10 orders and all 29 orders are considered. The effectiveness of the NPSO algorithm is examined by comparing with stander GA. They have the same 3 crossover operators and the 5 mutation operators designed in part III. The selection mechanism of GA is proportional model. To get the stable performance, GA and NPSO algorithm with and without heuristic rules (HR) are tested ten times. The numerical tests were performed on an HP 1.8GHz, 256MB AMD PC.

The results of scheduling problem involving 29 orders are listed in Table 3. As show,

(1) The minimum and average value of earliness of NPSO and NPSO with heuristic rules are both better than GA and GA with heuristic rules in any operator. It means NPSO are more appropriate than GA for short-term scheduling of single stage batch plants.

(2) NPSO and GA become more effective when heuristic rules are added.

(3) Among all crossover and mutation operators, NPSO algorithm with heuristic rules performs best when C3 and M2 are chose.

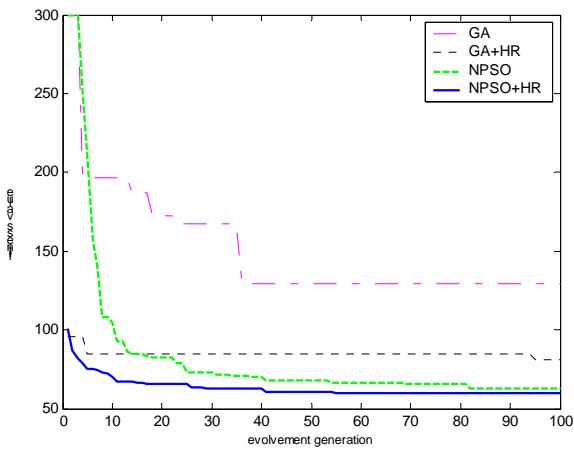
The optimal solutions are illustrated with bold letters. Where, *PS* and *GN* indicate the size of population and the generation number of termination respectively. The convergence rate figure of 4 algorithms with C3 and M2 operators for the case having all 29 orders and 10 orders are

shown in Figure 3(a), (b). Gantt charts of NPSO algorithm with heuristic rules used C3 and M2 operators for the case having all 29 orders and 10 orders are shown in Figure 4(a), (b). Number in frame denotes order and number out of frame denotes the beginning and finishing time of the order respectively, which is shown in integer here. The numbers below the horizontal lines indicate the order number being processed at that time.

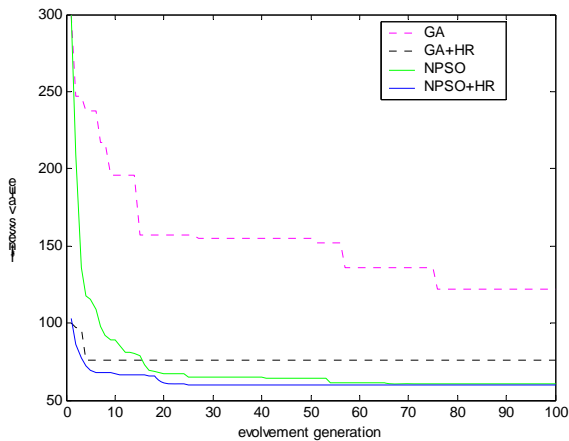
Figure 3(b) demonstrates the effectiveness of NPSO algorithm and heuristic rules to the scheduling problems involving 10 orders. And minimization of earliness is zero since each order is finished in due date as shown in Figure 4(b).

Table 3 : The comparisons of 4 algorithms

Algorithm		Min / Average earliness		
		C1	C2	C3
GA	M1	121.8450/136.9290	114.5980/141.3619	117.5090/148.1940
NPSO		60.6970/62.2991	64.6680/68.7159	61.3320/67.1802
GA+ heuristic rules		75.6800/88.2136	81.3160/90.4419	82.3630/92.3404
NPSO+ heuristic rules		60.0220/61.4641	60.4340/62.0779	60.0220/61.6425
GA	M2	129.8820/143.9154	129.2560/ 147.6314	116.9010/152.0024
NPSO		60.3990/63.8701	61.5510/ 69.5250	61.4910/66.3328
GA+ heuristic rules		80.4650/89.4721	84.6760/ 90.4824	82.4340/87.2815
NPSO+ heuristic rules		60.0570/62.3959	60.0220/ 61.1994	60.0220/61.0251
GA	M3	117.9050/141.6352	109.8680/ 150.4345	131.4330/150.0472
NPSO		60.8600/63.3337	61.4490/ 65.1619	62.4680/65.1448
GA+ heuristic rules		80.1300/86.5243	85.1720/ 91.1883	77.4850/90.7956
NPSO+ heuristic rules		60.0220/61.4897	60.4340/ 63.0027	60.8600/61.6759
GA	M4	133.1520/151.9793	114.1100/141.0924	130.6190/148.4472
NPSO		61.1520/64.7309	61.9010/66.8772	61.2180/66.4415
GA+ heuristic rules		83.3420/88.7571	80.2410/86.7714	80.9980/91.8598
NPSO+ heuristic rules		60.0570/61.7522	60.8600/62.3254	60.7580/62.0366
GA	M5	109.8190/135.2402	133.4080/150.1857	126.4960/142.4303
NPSO		60.0220/62.1280	66.2000/70.7749	62.2530/65.9475
GA+ heuristic rules		82.0730/91.5971	78.0230/90.6729	84.8050/91.0062
NPSO+ heuristic rules		60.0220/61.6858	60.3990/61.5579	60.0570/62.0037
PS=200, GN=100				

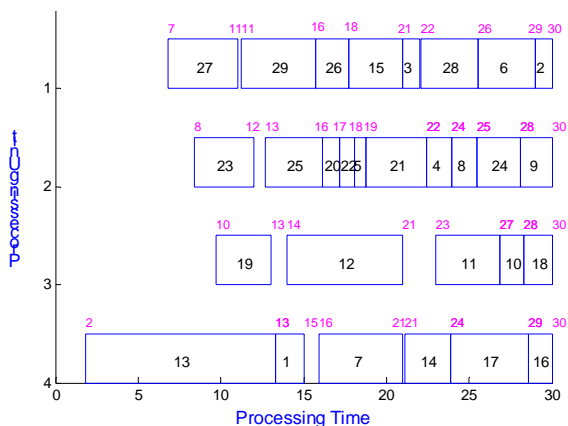


(a) Order 29

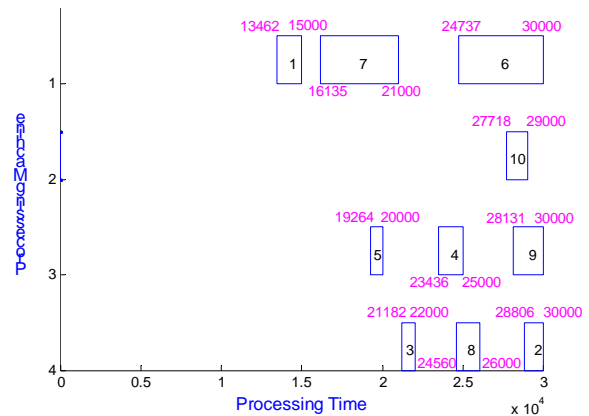


(b) Order 10

Figure 3 : Convergence rate contrast figure of 4 algorithms with C3, M2 operators



(a) Order 29



(b) Order 10

Figure 4 : Gantt chart of NPSO with heuristic rules used C3, M2 operators

5. Conclusions

PSO has been introduced as an optimization technique in real-number spaces. However, scheduling of batch plants is set in a discrete space. In this paper, the proposed NPSO algorithm generated particle's flying velocity and position by introducing GAs operators for the short-term scheduling of batch plants with parallel units to minimization of earliness. And, 5 heuristic rules were defined to generate a better initialization population and no effect on the optimality of the scheduling problem. The problem of scheduling considered is the short-term scheduling for a set of orders with due dates in batch plants with one stage and units in parallel. It was formulated as an MILP model in which the objective is the minimization of earliness. MILP model used the continuous-time domain representation. The 4 algorithms, GA and NPSO algorithm with and without heuristic rules were compared. Results show that NPSO and GA become more effective when heuristic rules are added. Although NPSO does not guarantee the optimality, it provides solutions with good quality in a reasonable time limited. Meanwhile, all the particles tend to converge to the best.

6. References

- [1] J.M. Pinto, I.E. Grossmann, "Assignment and Sequencing Model for the Scheduling of Process Systems", *Annals of Operations Research*, 1998, 81, pp. 433-441.
- [2] D.W.T. Rippin, "Batch Process Systems Engineering: A Retrospective and Prospective Review", *Computers & Chemical Engineering*, 1993, 17, Suppl, pp. 1-13.
- [3] J.F. Pekney, G.V. Reklaitis, "Towards The Convergences of Theory and Practice: A

- Technology Guide for Scheduling/Planning Methodology”, Pekney J,Blau G. Foundations of Computer Aided Process Operations--American Institute of Chemical Engineers Symposium Series 320, 1998, 94, pp. 91-111.
- [4] I.A. Karimi, C.M. McDonald, “Planning and Scheduling of Parallel Semicontinuous Processes. 2. Short-Term Scheduling.” *Industrial & Engineering Chemistry Research*, 1997, 36(7), pp. 2701-2714.
- [5] M.G. Ierapetritou, C.A. Floudas, “Effective Continuous-Time for Short-Term Scheduling. 3. Multiple Intermediate Due Dates.” *Industrial & Engineering Chemistry Research*, 1999, 37, pp. 3446-3461.
- [6] J. Cerda, G.P. Henning, and I.E. Grossmann, “A Mixed-Integer Linear Programming Model for Short-Term Scheduling of Single-Stage Multiproduct Batch Plants with Parallel Lines.” *Industrial & Engineering Chemistry Research*, 1997, 36, pp. 1695-1707.
- [7] C.A. Méndez, G.P. Henning, and J. Cerda, “Optimal Scheduling of Batch Plants Satisfying Multiple Product Orders with Different Due-Dates.” *Computers & Chemical Engineering*, 2000, 24(9), pp. 2223-2245.
- [8] X. Xu, G. Zheng, and S. Chug, “Design and optimize of batch plants,” *Petrochemical Technology*, 1991, 9, pp. 639-644.
- [9] W. Chen, W. Jiang, “Production scheduling of FMS in chemical industry,” *Chemical Automation and Equipment*, 1991, 6, pp. 23-30.
- [10] J. Kennedy, R.C. Eberhart, “Particle Swarm Optimization”, IEEE Int'l Conf on Neural Networks, Perth, Australia, 1995, pp. 1942-1948.
- [11] R.C. Eberhart, Y. Shi, “Tracking and optimizing dynamic systems with particle swarms”, Proceedings of the IEEE Congress on Evolutionary Computation, Seoul, Korea, 2001, pp. 94-97.
- [12] Y. Shi, R.C. Eberhart, “A modified particle swarm optimizer”, IEEE World Congress on Computational Intelligence, 1998, pp. 69-73.
- [13] Y. Shi, R.C. Eberhart, “Parameter selection in particle swarm optimization”, Evolutionary Programming VII: Proceedings of the Seventh Annual Conference on Evolutionary Programming, New York, 1998, pp. 591-600.
- [14] A. Salman, I. Ahmad, and S. Al-Madani, “Particle swarm optimization for task assignment problem,” *Microprocessors and Microsystems*, 2002, 26(8), pp. 363-371.
- [15] I.C. Trelea, “The particle swarm optimization algorithm: convergence analysis and parameter selection”, *Information Processing Letters*, 2003, 85(6), pp. 317-325.
- [16] Z. Lian, X. Gu, and B. Jiao, “A similar particle swarm optimization algorithm for permutation flowshop scheduling to minimize makespan,” *Applied Mathematics and Computation*, 2006, 175(1), pp. 773-785.
- [17] J. Pinto, I. Grossmann, “A Continuous-Time Mixed Integer Linear Programming Model for Short-Term Scheduling of Multistage Batch Plants,” *Industrial & Engineering Chemistry Research*, 1995, 34, pp. 3037-3051.