Particle Swarm Optimization Algorithm for a Batching Problem in the Process Industry

Lixin Tang* and Ping Yan

Liaoning Key Laboratory of Manufacturing System and Logistics, The Logistics Institute, Northeastern University, Shenyang 110004, PR China

An improved particle swarm optimization (PSO) algorithm is proposed to solve a typical batching problem in a batch processing plant of the process industry. The batching problem (BP) is to transform the primary requirements for products into sets of batches for each task with the objective of minimizing the total workload. On the basis of some preliminary properties, a novel particle solution representation is designed for the BP. Unlike the ordinary idea of taking an objective function as the fitness function for PSO, the original objective function incorporated with a constraint function is to act as the fitness function of the PSO where the constraint and the objective functions are evaluated successively. Such a fitness function, together with a forward repair mechanism, makes it possible for a faster convergence. Further, for each iterative generation, a local search heuristic is used to improve the global best particle found so far. To verify the performance of the proposed PSO algorithm, the well-known benchmark batching instances are tested. The relatively large-scale instances are also added to evaluate the algorithm. The computational results show that the improved PSO may find optimal or suboptimal solutions within a much shorter run time for all the instances.

1. Introduction

The batch mode of operation is prevalent across a wide spectrum of the process industry. Such operations are encountered in many applications including specialty chemical processes and the pharmaceutical and food-processing industries like manufacturing of sorbitol, modified starches, and specialty sugars. Batching decisions play an important role in batch plants of the process industry since any schedule performs on the basis of batches in a batch plant. In chemical batch processing, a batch is the quantity of material that undergoes processing by a single chemical process using an individual equipment unit[1] and the chemical process are called tasks, such as filtration and heating. Generally, batching decisions are composed of two aspects: (1) the number of batches and (2) the batch size. The batch size is often related to the reactor capacity and involves two categories including fixed and variable sizes.[2] With fixed batch sizes, all batches are of the same size for a given task, as opposed to a variable batch size environment, in which each batch has its own size. For instance, pharmaceutical plants usually handle fixed sizes for which integrity must be maintained, while solvent or polymer plants often handle variable sizes. In this paper, we consider a multipurpose batch processing problem with a fixed batch size environment which models a typical situation in the process industry. This problem is first introduced by Westenberger and Kallrath[3] who present it based on a batching problem (BP) in an existing plant, and we call it the WK problem. In the BP, given primary requirements of the final products, the requirements of all the products including both intermediate products and final products can be converted into batches for tasks during production. We address the problem of determining the batching decisions and the proportion of output (input) products for tasks to minimize workload over a horizon.

Batching problems arising in chemical batch production have been studied extensively in the past decade, owing to its practical importance and theoretical challenging nature. Numerous models and solution approaches have been developed for batching problems. The relative papers on batching problems can be roughly classified into two categories according to whether the batching decisions involve joint decisions of scheduling. Some researchers concentrate on integrating batching decisions with scheduling. Their monolithic models require both batching and scheduling decisions to be taken simultaneously. Considering the complexity of an integrated problem, most researchers adopt the method of solving the models directly with the help of mixed integer linear programming (MILP) or mixed integer nonlinear programming (MINLP) solvers. Such work can be found in the works of Maravelias and Grossmann,[4] Sundaramoorthy and Karimi,[5] and Shaik and Floudas.[6] Other researchers solve the batching problems separately without considering the scheduling decisions. For instance, Bruker and Hurink[7] dealt with a multiproduct batch processing problem which models a special situation in the chemical processing industry. They solved the BP in principle by a heuristic approach. Recently, Neumann et al.[3] have been working on the WK batching problem. They show that the feasibility problem for the BP is NP-hard and present a nonlinear mixed-integer program formulation. By transforming that nonlinear program into a linear formulation, they solve the BP using a MILP solver CPLEX. As for some comprehensive overviews of planning problems in the process industry, they can be found in the works of Kallrath[8] and Méndez et al.[2] In the literature, there have been significant research efforts in the development of modeling techniques[9−11] where most models are solved by using software packages. However, such methods may not be applied widely to practical projects because of a much longer time for obtaining an optimal solution to the large-scale problem. In this paper, we focus on a new solution approach based on a particle swarm optimization (PSO) algorithm which is a competitive optimization approach especially for those large engineering optimization problems with complex constraints. In the proposed PSO algorithm, some characteristics of the BP are primarily analyzed to reduce the problem size slightly. Then, particles are redesigned to represent the batching schemes based on the characteristics of the BP. A novel constraint-handling mechanism is also introduced into the

* To whom correspondence should be addressed. Tel.: +86-24-83680169. Fax: +86-24-83680169. E-mail: lixintang@mail.neu.edu.cn (L.T.).

10.1021/ie801742m CCC: $40.75
Published on Web 09/15/2009

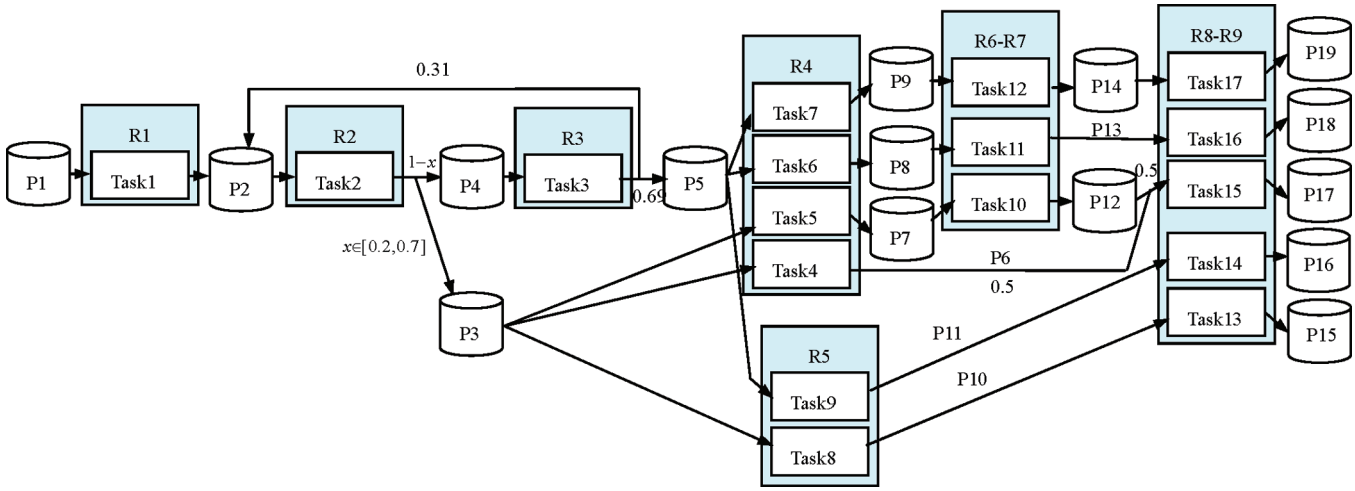

Figure 1. Production flow of the batching problem.

PSO to solve the relevant constraints. In addition, the proposed PSO adopts a local search heuristic to improve the global best particle found so far.

The rest of this paper is organized as follows: details of the BP are described in section 2. In section 3, we discuss some properties of the BP which give a guide for the solution approach. In section 4, a novel PSO combined with a constraint-handling mechanism and a local search heuristic for the global best particle is proposed to solve the BP. Computational results and comparisons with the results obtained from CPLEX solver are presented in section 5. Finally, we end the paper with some conclusions and future work in section 6.

2. Problem Description

The BP under study may be stated as follows. Some final products must be produced within a given planning horizon T. For each final product, a sequence of chemical processes will be undergone where the chemical processes are called tasks in what follows. Each task has its own input and output products. Aside from the final products, some raw materials and intermediate products are also involved in the production process. In batch production mode, some products can be buffered in storage facilities (e.g., tanks) of given capacity. Other intermediate products are perishable and cannot be stored. Such perishable intermediate products must be consumed without any delay so that no perishable product is in stock at any time. The production flow can be represented by a network connecting the processing tasks and storage facilities according to the material flows. The predecessors of a task i provide inputs which are transformed by i into outputs serving as inputs for successors of i. Part of the output of i may also be used again as the input of predecessors of task i. Thus, the processing tasks and storage facilities are linked by divergent, convergent, and cyclic material flows. Proportions of input and output products of tasks are fixed for some tasks but flexible for others. In the case of flexible proportion situation, input or output proportions of the task may vary within the given range.

Tasks are assigned to the processing units. For each task, it is associated with a set of processing units which are suitable for performing the corresponding chemical process. The tasks are performed in batch production mode in which a batch is the basic production unit of a task. Depending on the chemical reactions involved and the capacity of the corresponding processing units, minimum and maximum batch sizes are limited for each task. All the batches of a task have the same batch size. It is necessary to split a task into several batches which are processed one by one on the assigned processing unit. Note that the time needed for processing a batch is a constant for a given task and is independent of the production quantity.

Table 1. Task Information of the Batching Problem

task	alternative processing units	alternative processing times	materials consumed	materials produced	batch size [min, max]
1	R1	2	*P*1	*P*2	[3, 10]
2	R2	4	*P*2	*P*3, *P*4	[5, 20]
3	R3	2	*P*4	*P*2, *P*5	[4, 10]
4	R4	4	*P*3	*P*6	[4, 10]
5	R4	4	*P*3	*P*7	[4, 10]
6	R4	4	*P*5	*P*8	[4, 10]
7	R4	4	*P*5	*P*9	[4, 10]
8	R5	6	*P*3	*P*10	[4, 10]
9	R5	6	*P*5	*P*11	[4, 10]
10	R6/R7	4/5	*P*7	*P*12	[3, 7]
11	R6/R7	5/6	*P*8	*P*13	[3, 7]
12	R6/R7	6/6	*P*9	*P*14	[3, 7]
13	R8/R9	4/6	*P*10	*P*15	[4, 12]
14	R8/R9	4/6	*P*11	*P*16	[4, 12]
15	R8	4	*P*6, *P*12	*P*17	[4, 12]
16	R8/R9	6/6	*P*13	*P*18	[4, 12]
17	R8/R9	6/6	*P*14	*P*19	[4, 12]

Analyzing the BP in more detail, it is easy to see that there are three kinds of decisions need to be made, namely, proportions of input and output products for tasks and the number and the size of batches for each task. The constraints mainly result from three aspects: the material balance; the limited capacities of storage facilities; and the batch size restrictions of tasks. The BP is to convert the primary requirements of products into sets of batches for each task such that the workload is minimized. The detailed production flow of the BP is illustrated by Figure 1. The production flow involves 15 storage facilities for the storable products $P1$, ..., $P5$, $P7$, $P8$, $P9$, $P12$, $P14$, ..., and $P19$ and 9 multipurpose processing units R1−R9. Note that the intermediate products $P6$, $P10$, $P11$, and $P13$ are perishable. Table 2 summarizes inventory bounds for the storable products. The primary requirements for the final products $P15-P19$ have to be produced within a given planning horizon with 24 h of working time each day, where the time unit is 1 h. Table 1 describes the bounds of batch size, processing time of the corresponding processing unit, and the materials consumed and produced by each task. For each task, input and output proportions are shown in Figure 1. Among all the tasks, only the output proportions of task 2 are flexible. Recall that product

Table 2. Inventory Bounds of the Batching Problem

	*P*1	*P*2	*P*3	*P*4	*P*5	*P*7	*P*8	*P*9	*P*12	*P*14	*P*15	*P*16	*P*17	*P*18	*P*19
initial stock	∞	10	10	0	10	0	0	0	0	0	0	0	0	0	0
max stock	∞	30	30	15	30	10	10	10	10	10	∞	∞	∞	∞	∞

*P*2, which belongs to a cycle, cannot be consumed completely for production in the case of cyclic material flow.

The model formulation for the BP can be formulated as follows, which is presented by Neumann et al.:[3]

$$\min \sum_{i\in I} \bar{t}_i n_i \tag{1}$$

subject to

$$\underline{B}_i \le B_i \le \bar{B}_i \quad i \in I \tag{2}$$

$$0 \le n_i \le \bar{n}_i \quad i \in I \tag{3}$$

$$\sum_{p\in P_i^+} \lambda_{ip} = -\sum_{p\in P_i^-} \lambda_{ip} = 1 \quad i \in I \tag{4}$$

$$\underline{\lambda}_{ip} \le \lambda_{ip} \le \bar{\lambda}_{ip} \quad i \in I, p \in P_i \tag{5}$$

$$\sum_{i\in I_p} \lambda_{ip} B_i n_i \ge D_p \quad p \in P \tag{6}$$

$$\sum_{i\in I_p} \lambda_{ip} B_i n_i \le D_p + C_p \quad p \in P \tag{7}$$

$$\lambda_{ip} B_i = -\lambda_{i'p} B_{i'} \quad p \in P^p, (i,i') \in I_p^+ I_p^- \tag{8}$$

The objective function (1) aims at the minimization of workload. Constraints (2) and (3) are the restrictions of size and number of batches, respectively. Constraints (4) are the material proportion balance constraints. Constraints (5) ensure that the proportion of output (input) product for each task is within the given range. Constraints (6) guarantee that sufficient quantities of products are available as inputs of consuming tasks, and inventory constraints (7) ensure that the final inventory must not exceed the given capacity of the storage facility. The last constraints (8) guarantee that, for a perishable product, the amount produced by a batch of a task must equal the amount consumed by a batch of another task.

3. Preliminary Properties

Analyzing the production flow of the BP in detail, we can observe some relationships among the material amounts consumed by tasks.

Property 1. Suppose that p is a perishable product. If p is the only input of task i and is also the only output of task j, then $B_i = B_j$, $n_i = n_j$ in any feasible batching solution.

Proof. Since product p cannot be stored, it must be consumed completely without any residuals at any time once product p is produced. Therefore, this property holds.

In fact, if there are some other input (output) products except for the perishable product in a task, similar properties can also be concluded. For instance, *P*6 is perishable in the BP. Task 4 produces *P*6 while task 15 consumes it. Besides, *P*12 that is produced by task 10 is also an input of task 15. But, the number and size of batches for task 15 are independent of task 10 and completely depend on the batch information of task 4 since *P*6 is perishable and the input proportions of *P*6 and *P*12 in task 15 are fixed. Property 1 implies that it is not necessary to make batch decisions for all the tasks when the production flow involves some perishable products. As a result, property 1 reduces the problem scale to some extent.

For each task, batching decisions are hard to make, because of the divergent, convergent, cyclic material flows and the flexible proportions of input and output products of tasks. However, among all the tasks, task 1 as the source task which has no predecessor tasks in the production flow, is influenced only by its successor tasks. Let TP_i ($i \in I$) be the production amount of task i. We denote the initial and residual stocks of product P as initstock$_P$ and residulstock$_P$, respectively. Then, a precondition for the optimal batching can be obtained from task 1.

Property 2. The optimal batching solution to the BP satisfies the following condition:

$$n_1 = \lceil (TP_2 - \lambda_{32} TP_3 - \text{initstock}_{P2} + \text{residulstock}_{P2})/\bar{B}_1 \rceil$$

Proof. Let n_1 denote the number of batches for task 1 which is calculated by the equation in property 2. To prove this property for the BP, suppose by contradiction, there exists an optimal solution S in which the number of batches for task 1 is not equal to n_1 and we represent the number of batches for task 1 in S as n_1'. On the one hand, if $n_1' < n_1$, then the following inequality holds:

$$n_1' \le n_1 - 1 < (TP_2 - \lambda_{32} TP_3 - \text{initstock}_{P2} + \text{residulstock}_{P2})/\bar{B}_1$$

This means that $n_1'\bar{B}_1 < TP_2 - \lambda_{32} TP_3 - \text{initstock}_{P2} + \text{residulstock}_{P2}$. In that case, the total amount of *P*2, i.e., $n_1'\bar{B}_1 + \lambda_{32} TP_3 + \text{initstock}_{P2}$, cannot meet the production requirement of *P*2 which is equal to $TP_2 + \text{residulstock}_{P2}$. On the other hand, if $n_1' > n_1$, then we have $\bar{t}_1 n_1 + \sum_{i=2}^{|I|} \bar{t}_i n_i < \bar{t}_1 n_1' + \sum_{i=2}^{|I|} \bar{t}_i n_i$. Clearly, it violates the optimality of S. This completes the proof.

The result of property 2 indicates that the batching decisions of task 1 totally depend on the production amount of tasks 2 and 3. This property frees us from making batching decisions for task 1.

Property 3. There exists an optimal solution to the BP which satisfies the following properties:

(a) If $D_{15} = 0$, then $TP_8 = TP_{13} = 0$
(b) If $D_{16} = 0$, then $TP_9 = TP_{14} = 0$
(c) If $D_{17} = 0$, then $TP_4 = TP_5 = TP_{10} = TP_{15} = 0$
(d) If $D_{18} = 0$, then $TP_6 = TP_{11} = TP_{16} = 0$
(e) If $D_{19} = 0$, then $TP_7 = TP_{12} = TP_{17} = 0$

Proof. The proof can be done by contradiction. Suppose that there does not exist any optimal solution which satisfies property 3 when $D_p = 0$ for a final product p. Let S be an optimal solution to the BP in which $D_p = 0$. As there is no primary requirement for the final product p, the requirement amounts of some relative intermediate products are also equal to zero. Then, production could be decreased for the relative tasks in S until the inventory levels of both the relative intermediate products and p equal to zero at the completion of production. The workload will be unchanged in S as the workload is calculated based on the number of batches instead of the production amount. Thus, S as an optimal solution to the BP satisfies property 3, and this contradicts the supposition. This completes the proof of a−e in property 3.

d	1	...	$\lvert P_1\rvert$	...	$\sum_{i\in I}\lvert P_i\rvert$	$\sum_{i\in I}\lvert P_i\rvert+1$	...	$\sum_{i\in I}\lvert P_i\rvert+\lvert I\rvert$
x_{jd}	λ_{11}	...	$\lambda_{1,\lvert P_1\rvert}$	...	$\lambda_{\lvert I\rvert,\lvert P_{\lvert I\rvert}\rvert}$	TP_1	...	$TP_{\lvert I\rvert}$

Figure 2. Encoding strategy of particle j.

d	1	2	3	4	5	6	7	8	9	10	11	12	13
x_{jd}	λ_{23}	TP_2	TP_3	TP_4	TP_5	TP_6	TP_7	TP_8	TP_9	TP_{10}	TP_{11}	TP_{12}	TP_{17}

Figure 3. Solution representation of particle j in the batching problem.

4. PSO Algorithm for the Batching Problem

4.1. Standard PSO Algorithm. PSO is a relatively recent evolutionary computation technique developed by Kennedy and Eberhart.[12] It is motivated by the observation of the social behavior of composed organisms, such as bird flocking and fish schooling. PSO executes a population-based search procedure in which the exploring agents are called particles. In PSO, the system is initialized with a population of random solutions and searches for optima by updating generations. The position of a particle is represented as an m-dimensional vector in the problem space. Each particle flies in the dimensional problem space with a velocity which is adjusted during time according not only to their own experience but also to the experience of other particles: in particular, the best position so far achieved by the particle j itself $P_j = (p_{j1}, p_{j2}, ..., p_{jm})$ and the best position so far achieved by the whole population $P_g = (p_{g1}, p_{g2}, ..., p_{gm})$. Denoting with $X_j^t = (x_{j1}^t, x_{j2}^t, ..., x_{jm}^t)$ and $V_j^t = (v_{j1}^t, v_{j2}^t, ..., v_{jm}^t)$ respectively the position and velocity of particle j at iteration t of the PSO algorithm, the following equations are used to iteratively modify the particles' velocities and positions:[13]

$$v_{jd}^{t+1} = wv_{jd}^t + c_1r_1(p_{jd} - x_{jd}^t) + c_2r_2(p_{gd} - x_{jd}^t) \quad (9)$$

$$x_{jd}^{t+1} = x_{jd}^t + v_{jd}^{t+1} \quad (10)$$

where $j = 1, 2, ..., N$ and N is the size of population. The inertia weight and the iteration counter are denoted by w and t, respectively. Acceleration coefficients c_1 and c_2 are both positive constants. r_1 and r_2 are the random numbers in [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.

Owing to a very few parameters that need to be adjusted, PSO has been used across a wide range of applications including power and voltage control,[14] mass-spring systems,[15] task assignment,[16] and traveling salesman problems.[17] In recent years, several studies applying the PSO approach to the optimization problems arising from the chemical industry have appeared in the literature. PSO has been applied to the parameter estimation in the chemical engineering field for the development of mathematical models[18] and the dynamical analysis in chemical processes.[19] However, to the best of the authors' knowledge, none of them faces the BP. In the following sections, we propose an improved PSO algorithm to solve the BP. First, the particle is to be redesigned to represent a batching scheme for the BP. Otherwise, infeasible particles may be generated at the search stage because of the complex constraints from the BP. Thus, constraints are dealt with in the PSO while evaluating solutions. At last, a local search heuristic for the global best particle found so far is merged into the PSO to enhance the performance of the PSO. Details are given in the following.

4.2. Particle Solution Representation and Initial Population Generation. Solution representation is one of the most important issues when designing the PSO algorithm. Generally, each particle in PSO is encoded as an array with m dimensions or elements, representing a possible solution in a multidimensional problem space. In order to construct a relationship between the problem domain and the PSO particles for the BP, we present dimensions of a particle for proportions λ_{ip} ($i \in I$, $p \in P_i$) of input (output) products and the production amount TP_i ($i \in I$) of tasks. In other words, for a particle, some dimension members represent the proportions of input (output) products and the others signify the production amount of tasks. TP_i and λ_{ip} ($i \in I$, $p \in P_i$) of the predecessor tasks always locate before those of the successor tasks in the encoding strategy of a particle. All the position members of a particle are continuous values. Once the production amount of each task is determined, the number and size of batches can be calculated simultaneously. The particle is herein used to represent a possible batching scheme indirectly. The following Figure 2 describes the encoding strategy of particles, where x_{jd} is the position value of particle j in the dth dimension.

Note that batch information of some tasks can be obtained by the other tasks when a perishable product occurs according to properties 1 and 2. Moreover, if the primary requirements of some final products are zero, property 3 will give some batch information for those tasks which are related to the final products. Therefore, we do not need to make batching decisions for all the tasks and the dimension size of a particle will be decreased effectively. In the proposed PSO algorithm, the dimension size of a particle, i.e., m, is set to 13 for the BP. The first dimension denotes the output proportion λ_{23} of $P3$ in task 2, and the following dimensions represent the production amount of task 2, ..., task 12 and task 17, respectively. Figure 3 illustrates the solution representation of particle j for the PSO algorithm. Let $\bar{B}_i$ be the maximal batch-size of task i. Then, we will create $\lceil TP_i/\bar{B}_i \rceil$ batches of size $TP_i/\lceil TP_i/\bar{B}_i \rceil$ for task i. Note that the number of batches created by the above method is minimal for a given TP_i. The position of each particle is updated at each iteration step in the PSO algorithm, thus resulting in different batching schemes.

Recall that the proportion ranges of input (output) products for tasks are given a priori. We only need to compute the bounds on TP_i ($i \in I$) for the generation of initial population in the proposed PSO. Let FP be the set of final products. Given the primary requirements of final products, we can easily deduce the minimal production requirements R_p of all the products by a backtracked approach. Then, we convert the minimal production requirements for products into a lower bound on TP_i ($i \in I$). This method helps us to get the lower bounds $x_{ji}^{\text{initial\_low}}$ ($i \in$

I) on position values TP_i for particle j ($j \in \{1, 2, ..., N\}$) at the initial population stage. The eqs 11 and 12 define $x_{ji}^{\text{initial\_low}}$ and R_p, respectively. Let $x_{ji}^{\text{initial\_upp}}$ ($i \in I$) be the upper bound on TP_i for particle j. The upper bounds $x_{ji}^{\text{initial\_upp}}$ can be reckoned based on a forward approach by eq 13. For instance, the upper bounds on TP_3 and TP_4 are computed by $x_{j3}^{\text{initial\_upp}} = \bar{\lambda}_{24} x_{j2}^{\text{initial\_upp}} + \text{initstock}_{P4}$ and $x_{j4}^{\text{initial\_upp}} = \bar{\lambda}_{23} x_{j2}^{\text{initial\_upp}} + \text{initstock}_{P3} - x_{j5}^{\text{initial\_low}} - x_{j8}^{\text{initial\_low}}$, respectively.

$$x_{ji}^{\text{initial\_low}} = \max_{p \in P_i^+} \{R_p / \bar{\lambda}_{ip}\} \tag{11}$$

$$R_p = \begin{cases} \max\{\sum_{i \in I_p^-} (-\underline{\lambda}_{ip}) x_{ji}^{\text{initial\_low}} - \text{initstock}_p, 0\}, & p \in P/FP \\ D_p, & p \in FP \end{cases} \tag{12}$$

$$x_{ji}^{\text{initial\_upp}} = \begin{cases} \max\{R_{p3}/\underline{\lambda}_{23}, R_{p4}/\underline{\lambda}_{24}\}, & i = 2 \\ \max_{p \in P_i^-} \{[\sum_{i' \in I_p^+} \bar{\lambda}_{i'p} x_{ji'}^{\text{initial\_upp}} + \text{initstock}_p - \sum_{i' \in I_p^-, i' \neq i} (-\underline{\lambda}_{i'p}) x_{ji'}^{\text{initial\_low}}]/(-\underline{\lambda}_{ip})\}, & i = 3, ..., |I| \end{cases} \tag{13}$$

An initial population of particles is herein constructed randomly in terms of the range of position values discussed above. As for the range of particle velocity, we clip the range of particle velocities v_{jd} ($d = 1, ..., m$) in $[-(x_{jd}^{\text{initial\_upp}} - x_{jd}^{\text{initial\_low}}), x_{jd}^{\text{initial\_upp}} - x_{jd}^{\text{initial\_low}}]$.

4.3. Method of Constraints Handling. As we all known, among the most constraint-handling techniques, the penalty function method has been the most popular technique which has been applied widely in engineering optimization due to its simple principle and easy implementation.[20] However, it tends to have difficulties in dealing with highly constrained search spaces.[21−23] Motivated by this fact, we propose a new constraint-handling mechanism for the PSO to solve the BP. The proposed constraint-handling mechanism consists of two parts: a forward repair procedure (dealing with the constraints resulted from the material balance and the limited storage capacities) and a constraint fitness-based method (solving the restrictions about the number and size of batches for tasks).

4.3.1. Forward Repair Procedure. Considering the material flow in the BP, we can observe that a lesser production amount of some task will possibly lead to surplus inventory for output products of predecessor tasks and scarce material provision as inputs for the successor tasks. The opposite circumstances will occur when the production amount of some task is too much. This means that there must be some quantitative relationships among x_{jd} ($d = 1, 2, ..., m$) of particle j ($j \in \{1, 2, ..., N\}$). This inspires us to design a forward repair procedure to amend the position values x_{jd}. For each particle j, the forward repair procedure consists of a sequence of repair operators from x_{j1} to x_{jm} along the production flow. The repair operator, whose function is to amend a certain position value x_{jd} ($d \in \{1, ..., m\}$) for particle j, is composed of two steps. First, identify lower bound x_{ji}^{low} and upper bound x_{ji}^{upp} on position value TP_i for particle j. Suppose that the position value TP_i ($i \in I$) is in the dth dimension of particle j. Values of x_{ji}^{low} and x_{ji}^{upp} are calculated based on the frontal position values x_{jr} ($r < d$), lower bounds $x_{ji}^{\text{initial\_low}}$ and upper bounds $x_{ji}^{\text{initial\_upp}}$ by eqs 14 and 15. Then, adjust the current dth position value x_{jd} into the identified range $[x_{jd}^{\text{low}}, x_{jd}^{\text{upp}}]$. Starting from the first dimension, this repair procedure adjusts position values one by one until the last dimension is corrected.

$$x_{ji}^{\text{upp}} = \begin{cases} x_{ji}^{\text{initial\_upp}}, & i = 2 \\ \min\{\min_{p \in P_i^-} \{[\sum_{i' \in I_p^+} \lambda_{i'p} x_{ji'} + \text{initstock}_p - \sum_{i' \in I_p^-, i' \neq i} (-\lambda_{i'p}) x_{ji'}^{\text{initial\_low}}]/(-\lambda_{ip})\}, x_{ji}^{\text{initial\_upp}}\}, & i = 3, ..., |I| \end{cases} \tag{14}$$

$$x_{ji}^{\text{low}} = \begin{cases} \max\{x_{ji}^{\text{initial\_low}}, R_{p3}/\lambda_{23}, R_{p4}/\lambda_{24}\}, & i = 2 \\ \max\{\max_{p \in P_i^-} \{[\sum_{i' \in I_p^+} \lambda_{i'p} x_{ji'} + \text{initstock}_p - \sum_{i' \in I_p^-, i' \neq i} (-\lambda_{i'p}) x_{ji'}^{\text{initial\_low}} - \text{maxstock}_p]/(-\lambda_{ip})\}, x_{ji}^{\text{initial\_low}}\}, & i = 3, ..., |I| \end{cases} \tag{15}$$

The following example illustrates the repair operator for x_{j3} of particle j. First, we calculate x_{j3}^{upp} and x_{j3}^{low}, namely, $x_{j3}^{\text{upp}} = \min\{\lambda_{24} x_{j2} + \text{initstock}_{P4}, x_{j3}^{\text{initial\_upp}}\}$ and $x_{j3}^{\text{low}} = \max\{\lambda_{24} x_{j2} + \text{initstock}_{P4} - \text{maxstock}_{P4}, x_{j3}^{\text{initial\_low}}\}$. And then, x_{j3} is rectified: if $x_{j3} < x_{j3}^{\text{low}}$, then $x_{j3} = x_{j3}^{\text{low}}$. If $x_{j3} > x_{j3}^{\text{upp}}$, then $x_{j3} = x_{j3}^{\text{upp}}$. Otherwise, we accept x_{j3} without any modification. Other position values of particle j can be amended in the same way as x_{j3}. Recall that this forward approach implies that the position values of frontal dimensions affect those of back dimensions greatly.

At each iteration step, after the new positions of the particles are determined according to eqs 9 and 10, this forward repair procedure will be applied to every particle in the swarm. Such a repair procedure guarantees that materials satisfy production requirements for all the products and flow feasibly within the production flow as a whole under the given inventory bounds simultaneously.

4.3.2. Constraint Fitness-Based Method. A repaired particle is possibly still unfeasible in that the bounds on the number and size of batches are not considered so far. Inspired by the basic idea for solving engineering optimization problems in genetic algorithm,[24] a constraint fitness-based method is constructed to deal with the residual constraints. In the proposed method, the fitness function consists of two kinds: one is the original objective function $f_{\text{obj}}(X)$ of the BP, and the other is the constraint function $f_{\text{con}}(X)$. It can be seen that the residual constraints are all inequalities, and we describe these constraints as $h_s(X) \leq 0$ ($s = 1, ..., n$), where n is the number of constraints to be solved. The following definition 1 gives the definition of member function $f_s(X)$ of the constraint function $f_{\text{con}}(X)$.

Definition 1. The member function $f_s(X)$ is defined as the fitness value of particle X to the constrained condition (s),

$$f_s(X) = \begin{cases} 0, & h_s(X) \leq 0 \\ \dfrac{h_s(X)}{h_{\max}(X)}, & h_s(X) > 0 \end{cases} \qquad s = 1, ..., n$$

where $h_{\max}(X) = \max\{h_s(X), s = 1, ..., n\}$.

Definition 2. The constraint function $f_{\text{con}}(X)$ is defined as the total member function $f_s(X)$ for particle X,

$$f_{\text{con}}(X) = \sum_{s=1}^{n} \beta_s f_s(X), \quad \sum_{s=1}^{n} \beta_s = 1, \quad 0 \le \beta_s \le 1, \quad \forall s \in \{1, ..., n\}$$

In order to preserve diversity in the swarm, β_s as the weight coefficient for constraint s is generated randomly. The function $f_{\text{con}}(X)$ gives a metric criterion to measure the feasibility of a particle X. Obviously, if $f_{\text{con}}(X) = 0$, it means that particle X is within the feasible domain. Otherwise, if $0 < f_{\text{con}}(X) < 1$, the bigger $f_{\text{con}}(X)$, the further away particle X is from the feasible domain. We assume that a feasible solution is better than any infeasible solution. Under this assumption, the constraint function $f_{\text{con}}(X)$ is considered first when we compare two particles, and the particle that has the smallest constraint function value will win at any time. If two particles have the same constraint function values, then their objective function values are compared. The one with the better objective function value will win. Compared with the penalty function method, this constraint fitness-based method frees us from adjusting weighted factors between constraints and objective functions and thus is easier to apply.

4.4. Local Search Heuristic for the Global Best Particle. Let P_g denote the position vector of the global best particle found so far in the proposed PSO. After the forward repair procedure, the lower and upper bounds on P_g are represented by X_g^{low} and X_g^{upp}, respectively. At first, the local search heuristic calculates the distance values y_d between x_{gd}^{low} and x_{gd}^{upp} ($d = 1, 2, ..., m$) to identify the neighborhood of P_g. And then, the heuristic uses uniform distribution to randomly generate five particles within the neighborhood of P_g. The new particles are created according to the following equation:

$$X_j^{\text{new}} = P_g + A, \quad j = 1, 2, ..., 5 \tag{16}$$

where $A^{\text{T}} = [\alpha_1, \alpha_2, ..., \alpha_m]$, $\alpha_d \sim U(-y_d/2, y_d/2)$, and $d = 1, 2, ..., m$. Finally, the neighborhood best, whose fitness is better than that of P_g, is culled from the five new particles to replace P_g. The local search heuristic is applied to the PSO algorithm after the global best particle is identified at each iteration step. In the particle swarm, individuals strive to improve themselves by imitating traits found in their successful peers, such as the global best and the personal best particles. So, a local search strategy for the global best particle will accelerate the search speed to some extend.

4.5. Framework of PSO for the Batching Problem. The proposed PSO algorithm accommodating the above constraint-handling mechanism and a local search heuristic for the global best particle can be summarized in the following steps:

Step 1. Determine the initial range of position and velocity for particles in the population.

Step 2. Generate randomly an initial population within the initial range of position values obtained in step 1.

Step 3. Repair all the particles in the population by the forward repair procedure.

Step 4. Evaluate all particle individuals in the population based on the proposed fitness function (including $f_{\text{obj}}(X)$ and $f_{\text{con}}(X)$) and update the personal best particle as well as the global best particle.

Step 5. Apply the local search heuristic to the global best particle found so far in the population.

Step 6. Update the velocities and positions of particles according to eqs 9 and 10.

Step 7. Go back to step 3, unless the termination condition is met.

Table 3. Computational Results of Small-Scale Instance Set with a Planning Horizon of 6 Days

instance	primary requirements	OPT	time[a] (s)	time[b] (s)	success rate (%)
1	(20, 20, 20, 0, 0)	97	2.250	0.125	100
2	(20, 20, 0, 20, 0)	116.5	0.937	0.121	100
3	(20, 20, 0, 0, 20)	112	1.204	0.125	100
4	(20, 0, 20, 20, 0)	117.5	1.234	0.120	100
5	(20, 0, 20, 0, 20)	113	0.828	0.122	100
6	(20, 0, 0, 20, 20)	132.5	1.063	0.125	100
7	(0, 20, 20, 20, 0)	123.5	1.360	0.125	100
8	(0, 20, 20, 0, 20)	119	0.875	0.126	100
9	(0, 20, 0, 20, 20)	152.5	0.719	0.109	100
10	(0, 0, 20, 20, 20)	139.5	1.016	0.105	100
11	(10, 10, 20, 20, 30)	203.5	1.485	0.141	100
12	(30, 20, 20, 10, 10)	177	1.969	0.122	100
13	(10, 20, 30, 20, 10)	189	3.093	0.125	100
14	(18, 18, 18, 18, 18)	201.5	2.312	0.141	100
15	(15, 15, 30, 30, 45)	325	3.515	0.125	100
16	(45, 30, 30, 15, 15)	282	1.985	0.116	100
17	(15, 30, 45, 30, 15)	300.5	1.828	0.140	100
18	(27, 27, 27, 27, 27)	297	4.250	0.125	100
19	(20, 20, 40, 40, 60)	398.5	1.359	0.121	100
20	(60, 40, 40, 20, 20)	338	2.109	0.141	100
21	(20, 40, 60, 40, 20)	367.5	1.422	0.125	100
22	(36, 36, 36, 36, 36)	394.5	3.547	0.141	100

[a] CPLEX. [b] PSO.

5. Experimental Results

In this section, we report on the results of computational tests to assess the effectiveness of our PSO algorithm for solving the BP. There are two data sets for testing in our experiments. The small-scale instance set with a planning horizon of 6 days (see Table 3) is presented by Neumann et al.[3] who solved all the instances to optimality by using CPLEX 6.0. As for the large-scale instance set, the test instances are generated randomly within the planning horizon T between 150 and 190 days as follows. For each planning horizon T, there are five instances and the primary requirements on the final products are random integers generated from a uniform distribution $[\bar{n}_1 \underline{B}_1/|FP|, \bar{n}_1 \bar{B}_1/|FP|]$ where $\bar{n}_1 = \sum_{k \in U_1} T/t_{1k}$. Table 4 describes all the 25 large-scale instances in details.

The parameters of the proposed PSO algorithm are configured as follows. A swarm population size of 20 is used for all the test instances. Default values for the parameters c_1 and c_2 have been used: $c_1 = c_2 = 1.5$. The inertia weight is set to decrease linearly from 0.9 to 0.4 during a run of the PSO. The time decreasing inertia weight allows the PSO to explore a large area at the start of the run (when the inertia weight is large) and to refine the search later by using a smaller inertia weight. For each test instance, the proposed PSO has been run 30 times, and the average results are presented. A run is terminated if the maximum of 1000 iteration steps is reached. The proposed PSO algorithm was coded in Visual C++. For all the test instances, we compare our algorithm with the results of CPLEX 11.0 including both the lower bounds (LB) and the optimal solutions (OPT). Computation is abandoned for CPLEX if a time limit of 3600 s is exceeded. All our experiments are implemented on a 2.33 GHz PC with 1.95 GB memory.

Table 3 gives the computational results for the small-scale instances. Column OPT reports the optimal solution value found by CPLEX, for the same instance as in the paper of Neumann et al.,[3] while the time columns contain the average computation time of CPLEX and PSO in seconds, respectively from left to right. The success rate denotes the percentage of successful runs. It is measured by the percentage of runs (out of 30) locating the global optima within the 1000 iteration steps. It can be seen from Table 3 that the proposed PSO has converged to optimality

Table 4. Computational Results of Large-Scale Instance Set

T (days)	primary requirements	LB[a]	OPT[b]	PSO[c]	time[d] (s)	Time[e] (s)	success rate (%)
150	(1165, 1641, 2409, 2376, 2738)	22334	22342.5	22342.5	2464.26	0.142	100
150	(1716, 2454, 1418, 3274, 1224)	21803.23	21809	21809	2702	0.140	100
150	(2167, 3573, 1571, 3506, 2122)	27752.18	27763	27763	2939.23	0.141	100
150	(1296, 2296, 3372, 2172, 1163)	20715.16	20719	20719	2510.38	0.125	100
150	(3422, 3276, 1844, 2087, 3110)	28028.25	28033.5	28033.5	2354.38	0.125	100
160	(2803, 1315, 3288, 3501, 2413)	26658.09	*f*	27934	3600	0.140	*f*
160	(1443, 1743, 1648, 1506, 3057)	20058.27	*f*	20097	3600	0.125	*f*
160	(1851, 3499, 2086, 2359, 2581)	25887.24	25892	25892	2929.69	0.143	100
160	(2003, 1922, 3752, 1171, 3173)	23960.99	23964	23964	3572.45	0.141	100
160	(3648, 1924, 1360, 2881, 2473)	25681.99	25685	25685	3581.17	0.125	100
170	(3656, 2317, 1979, 2688, 1932)	25473.99	25477	25477	3581.5	0.124	100
170	(2968, 2927, 4026, 2339, 2510)	29051	*f*	29338.5	3600	0.140	*f*
170	(4065, 2955, 2508, 2848, 3110)	31541.70	*f*	31585.5	3600	0.126	*f*
170	(2224, 3260, 2543, 1489, 2931)	24814.85	*f*	25157.5	3600	0.125	*f*
170	(3820, 3420, 1785, 3154, 3046)	31576.15	*f*	31638.5	3600	0.125	*f*
180	(4003, 2077, 2037, 4015, 3021)	31597.99	*f*	32163	3600	0.141	*f*
180	(2327, 2769, 3223, 4254, 1935)	30747.08	30751.5	30751.5	3123.34	0.141	100
180	(1513, 1971, 2403, 2936, 1467)	21246.88	*f*	21836	3600	0.140	*f*
180	(2025, 3175, 1335, 3776, 2558)	27613.62	*f*	28390	3600	0.143	*f*
180	(1795, 2393, 2864, 3516, 4239)	32237.68	*f*	32449	3600	0.144	*f*
190	(4293, 3875, 2559, 4070, 2390)	34228.33	*f*	35366.5	3600	0.127	*f*
190	(3330, 2691, 4151, 2979, 3209)	32888.94	*f*	33214.5	3600	0.122	*f*
190	(1719, 2730, 3697, 4214, 2607)	31738.04	*f*	32104.5	3600	0.141	*f*
190	(2558, 2400, 3431, 2082, 2150)	25107.33	*f*	25146	3600	0.125	*f*
190	(3683, 3240, 1967, 3633, 3398)	33604.14	*f*	33630	3600	0.123	*f*

[a] LB: lower bound obtained by CPLEX within 3600 s. [b] OPT: optimal solution obtained by CPLEX within 3600 s. [c] PSO: solution obtained by the proposed PSO algorithm. [d] CPLEX. [e] PSO. [f] Optimal solution can not be obtained by CPLEX within 3600 s.

Table 5. Gap Value Statistics for Large-Scale Instance Set[a]

T (days)	MAXOL (%)	MINOL (%)	AVGOL (%)	MAXPO (%)	MINPO (%)	AVGPO (%)	MAXPL (%)	MINPL (%)	AVGPL (%)
150	0.0390	0.0185	0.0282	0	0	0	0.0390	0.0185	0.0282
160	0.0184	0.0117	0.0142	0	0	0	4.7862	0.0117	1.0044
170	0.0118	0.0118	0.0118	0	0	0	1.3808	0.0118	0.5437
180	0.0144	0.0144	0.0144	0	0	0	2.8116	0.0144	1.6085
190							3.3252	0.0770	1.1401

[a] MAXOL/MINOL/AVGOL: maximum/minimum/average deviation from the lower bound (OPT − LB)/LB. MAXPO/MINPO/AVGPO: maximum/minimum/average deviation from the optimal solution (PSO − OPT)/OPT. MAXPL/MINPL/AVGPL: maximum/minimum/average deviation from the lower bound (PSO − LB)/LB.

with 100% success rate for all the small-scale test instances. In comparison with the results of Neumann et al., the proposed PSO algorithm finds the optimal solutions within a shorter amount of time since the computation times of CPLEX vary from 0.719 s (instance 9) to 4.25 s (instance 18). The superiority of PSO is more apparent in terms of the computational results of large-scale test set which is given in Tables 4 and 5. Three kinds of gaps are defined for comparison: (OPT − LB)/LB, (PSO − OPT)/OPT, and (PSO − LB)/LB. Statistics of the available gap values for the large-scale instances are shown in Table 5. From Table 4, we can see that the proposed PSO converges to optimality with a 100% success rate for those instances whose optimal solutions can be obtained by CPLEX. As for the instances whose optimal solutions are not available, the proposed PSO is compared with the lower bounds computed by CPLEX. Table 5 reports that the maximum gap (PSO − LB)/LB, i.e., MAXPL, is no more than 5% for all the large-scale instances. The experimental results demonstrate that the proposed PSO algorithm is capable of solving large-scale instances using lesser computational times (we never exceed 0.15 s) while it is difficult for CPLEX to solve those large-scale instances within 3600 s. From the computational results, it is possible to see that the proposed PSO algorithm works quite well since it has converged to optimality with 100% success rate for the instances whose optimal solutions are available and the average computation times are very short. The reason that the optimal solution is approached rapidly during the PSO implementation mainly lies in three aspects. First of all, some preliminary properties are applied to the PSO and thus the problem scale is decreased slightly. And then, the PSO incorporates the original objective function as well as the constraint function into the fitness function where the constraint and the objective functions are evaluated successively. By designing such a fitness function and a forward repair procedure, the proposed PSO can find the feasible particles rather easily in the iteration process since the fitness values of the feasible particles are always better than those of the infeasible ones. Finally, a local search heuristic for the global best particle extends more latitude of search space to anchor the global optimum.

6. Conclusions

Batching problems arising in the process industry have been studied extensively. Instead of the method of solving the models with the help of software packages that is commonly used, we present an improved PSO algorithm to solve the batching problem in the process industry. A novel constraint handling mechanism is incorporated into the proposed PSO algorithm to speed up the convergence of particles toward the feasible area, while a local search heuristic for the global best particle found so far improves the performance of PSO further. The performance of the proposed PSO algorithm is evaluated by comparing the experimental results with the optimal solutions and the lower bounds obtained by CPLEX solver. The proposed PSO obtained the optimal or suboptimal solutions for all the test instances

within a much shorter computation time and the maximal deviation from the lower bound is no more than 5%. In summary, the proposed PSO algorithm provides a novel approach to solve the batching problem in the process industry. Computational experience gained on a wide variety of test instances confirms the promising potential of the proposed PSO in solving the batching problem. Further research may be conducted to investigate the applications of other metaheuristics to the batching problems. It is also worthwhile to design PSO in solving some other batch scheduling problems with batching decisions in the process industry.

Acknowledgment

The authors wish to thank Prof. Christoph Schwindt, Prof. Norbert Trautmann, Prof. Rafael Fink, and Prof. Klaus Neumann for providing us all the test data and results with respect to the batching problem in their paper. The authors would also like to thank the anonymous referees for their helpful comments and suggestions. This work was supported by National Natural Science Foundation for Distinguished Young Scholars of China (Grant No. 70425003), National 863 High-Tech Research and Development Program of China (Grant No. 2006AA04Z174), and National Natural Science Foundation of China (Grant No. 60674084).

Nomenclature

Abbreviations

BP = batching problem
PSO = particle swarm optimization

Indices

i, i' = tasks
p = products
k = units
g, j = particles of PSO algorithm
d, m, r = dimensions of particles in PSO algorithm
s = constraints
t = iterations of PSO algorithm

Sets

I = all the tasks
I_p^+ = tasks producing product p
I_p^- = tasks consuming product p
P = all the products
P_i^+ = output products of task i
P_i^- = input products of task i
P_i = products which are either input or output of task i, $P_i = P_i^+ \cup P_i^-$
P^p = perishable products
FP = final products
U_i = alternative processing units of task i

Model Parameters and Variables

(1) Model Parameters
T = planning horizon
D_p = primary requirement minus the initial stock for product p
C_p = capacity of the storage facility for product p
t_{ik} = processing time of task i on the processing unit k
$\bar{t}_i$ = mean processing time of task i on any of the alternative processing units, $\bar{t}_i = \sum_{k \in U_i} t_{ik}/|U_i|$
$\underline{B}_i$ = minimum batch-size of task i
$\bar{B}_i$ = maximum batch-size of task i
$\bar{n}_i$ = maximum number of batches for task i, $\bar{n}_i = \sum_{k \in U_i} T/t_{ik}$
$\underline{\lambda}_{ip}$ = lower bound on the proportion of output (input) product p of task i
$\bar{\lambda}_{ip}$ = upper bound on the proportion of output (input) product p of task i
TP_i = production amount of task i
(2a) Integer Model Variables
n_i = number of batches for task i
(2b) Continuous Model Variables
B_i = batch size of task i
λ_{ip} = proportion of output (input) product p for task i ($\lambda_{ip} > 0$, if p is output product of task i; $\lambda_{ip} < 0$, if p is input product of task i; $\lambda_{ip} = 0$, if p is neither input nor output product of task i)

Algorithm Parameters and Variables

(1) Parameters
N = population size
w = inertia weight
c_1 = cognition learning factor
c_2 = social learning factor
r_1 = random number uniformly distributed in [0, 1]
r_2 = random number uniformly distributed in [0, 1]
$x_{jd}^{\text{initial\_low}}$ = lower bound on x_{jd} ($j \in \{1, 2, ..., N\}$, $d \in \{2, ..., m\}$) at the initial population stage
$x_{jd}^{\text{initial\_upp}}$ = upper bound on x_{jd} ($j \in \{1, 2, ..., N\}$, $d \in \{2, ..., m\}$) at the initial population stage
x_{jd}^{low} = lower bound on x_{jd} ($j \in \{1, 2, ..., N\}$, $d \in \{2, ..., m\}$) in the forward repair procedure
x_{jd}^{upp} = upper bound on x_{jd} ($j \in \{1, 2, ..., N\}$, $d \in \{2, ..., m\}$) in the forward repair procedure
(2) Variables
X_j^t = position of particle j at iteration t, $X_j^t = (x_{j1}^t, x_{j2}^t, ..., x_{jm}^t)$
V_j^t = velocity of particle j at iteration t, $V_j^t = (v_{j1}^t, v_{j2}^t, ..., v_{jm}^t)$
P_j = best position so far achieved by particle j, $P_j = (p_{j1}, p_{j2}, ..., p_{jm})$
P_g = best position so far achieved by the whole population $P_g = (p_{g1}, p_{g2}, ..., p_{gm})$

Literature Cited

(1) Grunow, M.; Günther, H.; Lehmann, M. Campaign Planning for Multi-stage Batch Processes in the Chemical Industry. *OR Spectrum.* **2002**, *24*, 281–314.

(2) Méndez, C. A.; Cerdá, J.; Grossmann, I. E.; Harjunkoski, I.; Fahl, M. State-of-the-art Review of Optimization Methods for Short-term Scheduling of Batch Processes. *Comput. Chem. Eng.* **2006**, *30*, 913–946.

(3) Neumann, K.; Schwindt, C.; Trautmann, N. Advanced Production Scheduling for Batch Plants in Process Industries. *OR Spectrum.* **2002**, *24*, 251–279.

(4) Maravelias, C. T.; Grossmann, I. E. New General Continuous-time State-task Network Formulation for Short-term Scheduling of Multipurpose Batch Plants. *Ind. Eng. Chem. Res.* **2003**, *42* (13), 3056–3074.

(5) Sundaramoorthy, A.; Karimi, I. A. A Simpler Better Slot-based Continuous-time Formulation for Short-term Scheduling in Multipurpose Batch Plants. *Chem. Eng. Sci.* **2005**, *60*, 2679–2702.

(6) Shaik, M. A.; Floudas, C. A. Unit-specific Event-based Continuous-time Approach for Short-term Scheduling of Batch Plants Using RTN Framework. *Comput. Chem. Eng.* **2008**, *32*, 260–274.

(7) Brucker, P.; Hurink, J. Solving a Chemical Batch Scheduling Problem by Local Search. *Ann. Oper. Res.* **2000**, *96*, 17–38.

(8) Kallrath, J. Planning and Scheduling in the Process Industry. *OR Spectrum.* **2002**, *24*, 219–250.

(9) Floudas, C. A.; Lin, X. Continuous-time versus Discrete-time Approaches for Scheduling of Chemical Processes: a Review. *Comput. Chem. Eng.* **2004**, *28* (11), 2109–2129.

(10) Castro, P. M.; Grossmann, I. E. New Continuous-time MILP Model for the Short-term Scheduling of Multistage Batch Plants. *Ind. Eng. Chem. Res.* **2005**, *44* (24), 9175–9190.

(11) Liu, Y.; Karimi, I. A. Scheduling Multistage, Multiproduct Batch Plants with Nonidentical Parallel Units and Unlimited Intermediate Storage. *Chem. Eng. Sci.* **2007**, *62*, 1549–1556.

(12) Kennedy, J.; Eberhart, R. Particle Swarm Optimization. *Proc. IEEE-ICNN* **1995**, 1942–1948.

(13) Eberhart, R. C.; Shi, Y. H. Evolving Artificial Neural Networks. *Proc. ICNNB* **1998**, 5–13.

(14) Abido, M. A. Optimal Power Flow Using Particle Swarm Optimization. *Electr. Power Energy Syst.* **2002**, *24*, 563–571.

(15) Brandstatter, B.; Baumgartner, U. Particle Swarm Optimization: Mass-Spring System Analogon. *IEEE Trans. Magn.* **2002**, *38*, 997−1000.

(16) Salman, A.; Ahmad, I.; Al-Madani, S. Particle Swarm Optimization for Task Assignment Problem. *Microprocess. Microsyst.* **2003**, *26*, 363–371.

(17) Clerc, M. Discrete Particle Swarm Optimization, Illustrated by the Travelling Salesman Problem. *New Optimization Techniques in Engineering*; Springer: Heidelberg, Germany, 2004; pp 219−239.

(18) Schwaab, M.; Biscaia, E. C., Jr.; Monteiro, J. L.; Pinto, J. C. Nonlinear Parameter Estimation through Particle Swarm Optimization. *Comput. Chem. Eng.* **2008**, *63*, 1542–1552.

(19) Ourique, C. O.; Biscaia, E. C., Jr.; Pinto, J. C. The Use of Particle Swarm Optimization for Dynamical Analysis in Chemical Processes. *Comput. Chem. Eng.* **2002**, *26*, 1783–1793.

(20) Coello, C. A. Use of a Self-adaptive Penalty Approach for Engineering Optimization Problems. *Comput. Ind.* **2000**, *41*, 113–127.

(21) Koziel, S.; Michalewicz, Z. Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization. *Evol. Comput.* **1999**, *7* (1), 19–44.

(22) Runarsson, T. P.; Yao, X. Stochastic Ranking for Constrained Evolutionary Optimization. *IEEE Trans. Evol. Comput.* **2000**, *4* (3), 284–294.

(23) Coello, C. A. Theoretical and Numerical Constraint Handling Techniques Used with Evolutionary Algorithms: a Survey of the State of the Art. *Comput. Methods Appl. Mech. Eng.* **2002**, *191* (11−12), 1245–1287.

(24) Deb, K. An Efficient Constraint Handling Method for Genetic Algorithms. *Comput. Methods Appl. Mech. Eng.* **2000**, *186*, 311–338.

Received for review November 15, 2008
Revised manuscript received July 7, 2009
Accepted August 28, 2009

IE801742M