Available online at www.sciencedirect.com

ScienceDirect

Expert Systems with Applications 34 (2008) 1133–1143

Expert Systems with Applications

www.elsevier.com/locate/eswa

Proportionate flexible flow shop scheduling via a hybrid constructive genetic algorithm

Der-Fang Shiau [a], Shu-Chen Cheng [b], Yueh-Min Huang [a,*]

[a] *Department of Engineering Science, National Cheng-Kung University, Tainan 701, Taiwan, ROC*
[b] *Department of Computer Science and Information Engineering, Southern Taiwan University of Technology, Tainan County 71005, Taiwan, ROC*

Abstract

The proportionate flow shop (PFS) is considered as a unique case of the flow shop problem in which the processing times of the operations belonging to the same job are equal. A proportionate flexible flow shop (PFFS) is a machine environment with parallel identical machines at each stage. This study presents an effective hybrid approach based on constructive genetic algorithm (CGA) for PFFS scheduling with the criterion to minimize the total weighted completion time (WCT). Minimizing the WCT in a PFFS problem significantly differs from the parallel-identical-machine scheduling problem, an optimal schedule in which the jobs on each machine are in weighted shortest processing time (WSPT) order. The proposed approach incorporates two fitness functions, and a population trained by a local improvement search based on tabu search with a candidate list strategy into CGA. Simulation results are compared with those of the column generation (CG) approach to demonstrate the effectiveness of the proposed hybrid approach. In particular, the CG approach has been applied successfully to solve various parallel machine scheduling problems, and yields high-quality solutions.
© 2006 Elsevier Ltd. All rights reserved.

Keywords: Proportionate flow shop; Total weighted completion time; Constructive genetic algorithm; Tabu search

1. Introduction

Pinedo (2002) described that a proportionate flow shop problem (PFS), the processing times of each job j on machines are the same and equal to p_j. In such a shop, any number of stages has a single machine at each stage. A proportionate flexible flow shop (PFFS) is a machine environment with a number of stages in series, with each stage having several identical machines in parallel. To the best of our knowledge, there is no published paper for dealing with the PFFS problem for minimizing total weighted completion time (WCT). Pinedo (2002) also observed the PFS problems in a number of cases similar to their single machine counterparts. Such cases include minimizing total completion time, maximum lateness, total number of tardy jobs and total tardiness. Unfortunately, minimizing the WCT in a PFS problem with equal machine speeds significantly differs from the single-machine scheduling problem with the same objective, i.e., an optimal permutation schedule in a PFS problem is obtained by executing the jobs not only by adopting the rule imposed by the weighted shortest processing time (WSPT) order (Smith, 1956), but also by scheduling the jobs based on the shortest processing time (SPT) rule. This study extends the PFS problem developed by Shakhlevich, Hoogeveen, and Pinedo (1998) to a PFFS problem with the criterion to minimize the WCT.

Genetic algorithms (GAs) have been intensively studied and applied in the production scheduling problems. Traditional GAs start with an initial set of random solutions called population. Each individual in the population is called a structure (chromosome), representing a solution to the problem. During each generation, the structures are evaluated applying some measures of fitness. A selection operator works together with the crossover operator to generate new structures and mutation operator to modify a structure, which provides the opportunity to escape

* Corresponding author. Tel.: +886 6 2757575x63336; fax: +886 6 2766549.
E-mail address: huang@mail.ncku.edu.tw (Y.-M. Huang).

0957-4174/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2006.12.002

from local optimal. Holland (1992) presented the "building block" hypothesis (schema formation and conservation) as a theoretical basis for the GA mechanism. From his perspective, preventing disruption of a good schema is the basis for the good behavior of a GA. However, a major problem with building blocks is that the schemata are measured indirectly by evaluating of their instances (structures). Goldberg, Korb, and Deb (1989), Goldberg, Deb, Kargupta, and Harik (1993) have addressed the problem of schemata evaluation and introduced the messy-GA that allows variable length strings and looks for the construction and preservation of good building blocks. An alternative to the traditional GA, named constructive genetic algorithm (CGA), has been proposed by de Oliveira and Lorena (2002), Lorena and Furtado (2001), Ribeiro and Lorena (2001) for evaluating schemata directly.

To the best of our knowledge, no published work deals with the parallel machine scheduling problem using the CGA approach, and applying the approach in different areas of problems is a challenge. This study proposes a hybrid CGA (HCGA) for PFFS scheduling with the WCT criterion. The CGA has some unique features over those of a traditional genetic algorithm, and in particular, incorporates two novel fitness functions on a structure into the evolution process. The first fitness function reflects the total cost (total weighted completion time) of a given schedule, while the second function drives the evolutionary process to a population trained by a local improvement search based on tabu search (TS) heuristic. Thus, the entire population of convergence moves faster and in a more appropriate direction than in traditional genetic algorithms based on a single fitness function. The proposed HCGA applies the CGA-based evolutionary process to effectively perform exploration for promising solutions. Moreover, The HCGA utilizes TS simultaneously to perform exploitation for solution improvement, where a candidate list strategy is designed to save time and improve the performance on a given iteration. Some effective hybrid heuristics have been shown to achieve better quality, time performance, and robustness than pure heuristic methods in flow shop problems (Liu, Wang, & Jin, in press; Noorul Hag, Ravindran, Aruna, & Nithiya, 2004; Wang & Zheng, 2003). Futhermore, Funda and Ulusoy (1999) and Chang et al. (2005) concluded that if the local search was included in GA, the solution quality can be improved. Kim and Han (2003) also proposed a hybrid genetic algorithm and neural network approach in activity-based costing. The CGA also evolves a population initially formed only by schemata, and controlled by recombination, to a dynamic population of well-adapted structures and schemata, which replaces the old structures or schemata with new ones to increase the diversity of solution pool. A comparative study is performed to test the performance of HCGA with that of the column generation (CG) approach to demonstrate the effectiveness of HCGA, since CG has been applied successfully to various parallel machine scheduling problems, and yields solutions of superior quality (Chen & Powell, 1999; Lee & Chen, 2000; Van den Akker, Hoogeveen, & Van de Velde, 1999).

The rest of this paper is organized as follows. In Sections 2 and 3, a survey of literature and the PFFS problem are discussed, respectively. In Section 4, the HCGA is introduced for dealing with the PFFS problem to minimize the WCT. In Section 5, results and analysis are made. Finally, some conclusions follow in Section 6.

2. Literature review

The proportionate flow shop (PFS) scheduling problem was first addressed by Ow (1985) and Pinedo (1985). Unlike the flow shop scheduling problem under certain constraints that has attracted considerable attention, the PFS problem has received less attention, with such studies including those of Adenso-Díaz (1992), Allahverdi (1996), Edwin Cheng and Shakhlevich (1999) and Hou and Hoogeveen (2003). The problem of minimizing the total completion time in a two-machine flow shop is NP-hard in the strong sense (Garey, Johnson, & Sethi, 1976). Shakhlevich et al. (1998) proved that the PFS problem of minimizing the total weighted completion time (WCT) in an n-job, m-machine can be solved optimally in $O(n^2)$. Choi, Yoon, and Chung (2006) also provided a tight lower bound to an optimal objective value in a two-machine PFS problem with different machine speeds. The objective was to minimize the WCT.

Flexible flow shop (FFS) or hybrid flow shop (HFS) scheduling problems are widely used in process industries. Moursli and Pochet (2000) presented a branch-and-bound algorithm to minimize the objective of the makespan (MS). Oğuz, Ercan, Edwin Cheng, and Fung (2003) developed nine heuristic algorithms for solving the two-stage HFS problem of minimizing MS. Gupta, Hariri, and Potts (1997) also proposed heuristics to minimize MS in a two-stage HFS with parallel identical machines at the first stage.

Recently, a genetic algorithm (GA) has been developed for HFS problem, with such studies including those of Şerifoğlu and Ulusoy (2004) and Oğuz and Ercan (2005). The objective was also to minimize MS. Chang, Chen, and Lin (2005) introduced a two-phase sub population GA to solve the parallel machine-scheduling problem with multiple objectives (MS and total tardiness). Moreover, Chiang, Chang, and Huang (2006) proposed a novel evolutionary approach, named particle swarm optimization (PSO), for solving multi-processor scheduling problem.

However, efforts to minimize the WCT objective are scarce in previous literature. Tozkapan, Kirca, and Chung (2003) considered a two-stage assembly flowshop problem to minimize the WCT and developed a branch-and-bound algorithm capable of handling problems with the number of jobs $\leqslant 20$. Recently, Tang, Xuan, and Liu (2006) proposed a new Lagrangian relaxation algorithm based on stage decomposition for HFS problems to minimize the WCT.

3. Formulation of the PFFS problem

The PFFS scheduling problem considered in this study is described as follows. There are n jobs, $N = \{1, 2, \ldots, n\}$, to be processed through s stages in series, $S = \{1, 2, \ldots, s\}$. Each stage $i \in S$ has m parallel identical machines with the property that all jobs have to be processed through all the stages on the same order, $M = \{1, 2, \ldots, m\}$. Each job $j \in N$ consists of s sequential operations O_{ij} ($i \in S; j \in N$) and each operation requires a processing time $p_{ij} = p_j$ on any one of the machines at stage i, and has a positive weight w_j (or priority).

3.1. Objective function

The objective is to find a feasible schedule that minimizes the total weighted completion time (WCT). Minimizing the WCT of the PFFS problems is referred to as $FFc|p_{ij} = p_j|\sum_{j=1}^{n} WjCj$, where $C_j = C_{sj}$ is the completion time of job j at stage s and can be calculated as follows:

$$C_j = \sum_{i=1}^{j} p_i + (s-1)\max\{p_1, \ldots, p_j\} \tag{1}$$

Then, a partial schedule consists of a subset of jobs of $N, \{1, 2, \ldots, h\}$, on a machine at each stage, and the total weighted completion time is computed:

$$\sum_{j=1}^{h} w_j C_j = \sum_{j=1}^{h} \sum_{i=1}^{j} w_j p_i + (s-1) \sum_{j=1}^{h} w_j \max\{p_1, \ldots, p_j\} \tag{2}$$

Eq. (2) illustrates that the first term is minimized by scheduling the jobs on a single machine according to the weighted shortest processing time (WSPT) order. The second term is minimized by scheduling the jobs according to the shortest processing time (SPT) rule.

3.2. Assumptions

(1) All N jobs to schedule are independent and available for processing at time zero.
(2) Each machine can handle at most one job at a time while a job can be processed on at most one machine at any time.
(3) Preemption is not allowed: a job, once started on a machine, it is processed to completion without interruption.
(4) Jobs are allowed to wait between two stages.

4. HCGA for PFFS problem

4.1. Representation of structure and schema

The most important issue in applying CGA to the problem $FFc|p_{ij} = p_j|\sum_{j=1}^{n} WjCj$ is to define an encoding scheme that allows one-to-one mapping between solutions and structures (chromosomes). Additionally, a representation decoder (assignment heuristic) is designed to convert a structure to a solution, which can thus be evaluated. A structure can be represented by $\tau = (\tau_1, \tau_2, \ldots, \tau_n)$ where $\tau_i \in \{0, 1, \#\}$. Each τ is partitioned into three sets, the seed set $J_1(\tau)$, the non-seed set $J_0(\tau)$ and the set $J_\#(\tau)$. Consider an example of 10 jobs to be processed through three stages, and each stage has three identical machines (see Fig. 1), $J_1(\tau) = \{1, 4, 8\}$ is the seed set; $J_0(\tau) = \{2, 3, 5, 7, 9\}$ is the non-seed set; and $J_\#(\tau) = \{6, 10\}$. Jobs 1, 4 and 8 form the seed, and every other job is assigned to one seed. The CGA works directly with schemata. The three-machine schema for the example can be $\tau = (1, 0, 0, 1, 0, \#, 0, 1, 0, \#)$, where each position τ_j in τ, receiving labels 1, 0 or #, represents that job $j \in N$ belongs to $J_1(\tau)$, $J_0(\tau)$ or $J_\#(\tau)$, respectively. The label # is indicated to express indetermination on schemata (# – *do not care*); i.e., a non-seed job not yet assigned to a machine. Moreover, τ is a schema since it contains #s. By contrast, the structure $\tau = (1, 0, 0, 1, 0, 0, 0, 1, 0, 0)$ without # is a feasible solution.

After m seed jobs are identified, a proposed assignment heuristic (AH) is employed to assign non-seed jobs to machines. Each seed job is assigned to one machine at the beginning of the assignment procedure. Subsequently, the unscheduled jobs in non-seed are attracted to join on each machine.

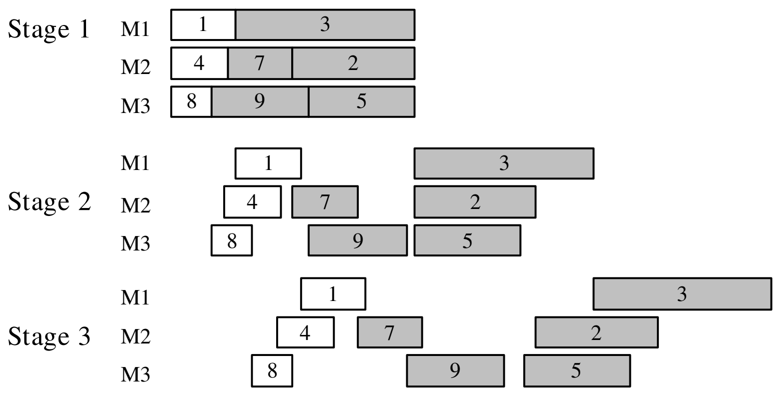

Fig. 1. An example of three-machine schema.

Relevant previous work on PFS problem of the minimizing WCT began with Shakhlevich et al. (1998), who showed that the problem can be solved optimally in O(n^2) time, and that the jobs can be scheduled in *WSPT–SPT* order. The *WSPT–SPT* order implies that the jobs are sequenced by Shakhlevich's WSPT-MCI algorithm, which is given as follows:

(1) Reindex the jobs in WSPT order, settling ties according to non-decreasing processing times. Let σ_1 be the sequence consisting of job 1; set $j := 2$.
(2) Derive σ_j from σ_{j-1} such that the cost increase is minimum; if there are several possibilities, then choose the one in which job j is inserted lasted. Set $j := j + 1$.
(3) If $j \leqslant n$, then go to step 2.
(4) Determine an optimal permutation schedule by scheduling the jobs in order of occurrence in σ_n.

That is, the optimal schedule for an assignment of the jobs to machines preserves the *WSPT-SPT* order of jobs on each machine. Therefore, reindex all jobs in *WSPT-SPT* order before applying AH. Let H be the set of all structures and schemata, for $i \in M$, $j \in N$, $s \in S$ and $\tau \in H$, then

δ_i,τ	ith machine schedule in τ,
$C_{i,\tau}$	completion time of ith machine schedule in τ,
tt_j	total execution time of job j; $tt_j = p_{1j} + p_{2j} + \cdots + p_{sj} = \mathrm{sp}_j$.

Thus, a total schedule for τ can be expressed as a set $\Pi_\tau = \{\delta_{1,\tau}, \delta_{2,\tau}, \ldots, \delta_{m,\tau}\}$ after performing AH, where $\delta_{m,\tau}$ is the job schedule on machine i, $\delta_{i,\tau}$ consists of a subset of jobs of N, $\{1,2,\ldots,n_i\}$, that preserves the *WSPT–SPT* order on a machine at each stage, where n_i denotes the number of jobs on machine i. The following AH is applied to assign non-seed jobs to machine schedules for each τ.

Step 1: Read τ from the current population.
Step 2: Initialize the completion time $C_{i,\tau}$ of each machine schedule δ_i,τ, $i = 1,2,\ldots,m$, to zero.
Step 3: Assign the seed jobs with label 1 to the seed set and the remaining ones with label 0 to the non-seed set for τ. Discard any job with label #. The order of the jobs in non-seed must preserve the *WSPT–SPT* order.
Step 4: For each seed job k ($k \in N$) in the seed set, assign to one machine schedule $\delta_{i,\tau}$ and add the total execution time tt_k of job k to the $C_{i,\tau}$.
Step 5: Assign the first job j in the non-seed set to the chosen machine schedule $\delta_{i,\tau}$ that has the minimum completion time $C_{i,\tau}$ among all m machine schedules; if two machine schedules have the same minimum $C_{i,\tau}$ value, choose one arbitrarily.
Step 6: Add the total execution time tt_j of job j to the completion time $C_{i,\tau}$ of the chosen machine schedule $\delta_{i,\tau}$, that is, set $C_{i,\tau} := C_{i,\tau} + tt_j$.
Step 7: Remove job j from the non-seed set list.
Step 8: Repeat Steps 5 to 7 until the non-seed set job list is empty.

Exactly m machine schedules are created after applying AH for each τ. For example in Fig. 1, $\Pi_\tau = \{\delta_{1,\tau}, \delta_{2,\tau}, \delta_{3,\tau}\}, \delta_{1,\tau} = \{1,3\}$, $\delta_{2,\tau} = \{4,7,2\}$ and $\delta_{3,\tau} = \{8,9,5\}$, corresponding to the seed set $J_1(\tau) = \{1,4,8\}$. Notably, the seed job k in $\delta_{i,\tau}$ may not be scheduled if the position that it occupies does not satisfy the *WSPT-SPT* order after performing AH. Hence, an order in which the insertion should be applied has to be specified.

4.2. CGA modeling

The problem $FFc|p_{ij} = p_j|\sum_{j=1}^{n} WjCj$ is modeled as the following bi-objective optimization problem (BOP):

$$\begin{array}{ll} \text{Min} & \{G(\tau) - F(\tau)\} \\ \text{Max} & G(\tau) \\ \text{Subject to} & G(\tau) \geqslant F(\tau), \quad \forall \tau \in H \end{array} \tag{3}$$

Functions G and F are defined as the double fitness evaluation of a structure or schema τ, which can be obtained using a specific representation. Both G and F are the key support functions guiding the evolution process to find the best structures, and must be properly identified in the problem $FFc|p_{ij} = p_j|\sum_{j=1}^{n} WjCj$. Function G reflects the total cost (total weighted completion time) of a total schedule Π_τ after the application of AH. Let $L(\Pi_\tau)$ be the objective value of a schedule Π_τ, and function G is defined by $G(\tau) = L(\Pi_\tau)$.

$$L(\Pi_\tau) = \sum_{i=1}^{m} T_{i,\tau} \tag{4}$$

where $T_{i,\tau} = \sum_{j\in\delta i,\tau}\sum_{k=1}^{j} w_j p_k + (s-1)\sum_{j\in\delta i,\tau} w_j \max\{p_1,\ldots,p_j\}$ is the total weighted completion time of the machine schedule $\delta_{i,\tau} \in \Pi_\tau$.

The other fitness function F is defined to play a role in a population trained by using a heuristic during the evolutionary process. The chosen heuristic is the tabu search, which is incorporated in CGA to propose the HCGA, and can be described in the next subsection.

4.2.1. Tabu search

As discussed in (Barnes & Laguna, 1993), tabu search (TS) is a kind of neighborhood search technique. The basic TS based on insert and swap moves for generating a neighborhood from the current schedule $\Pi_{C\tau}$. An insert move removes a job from one machine and inserts it into another. A swap move chooses a pair of jobs and switches their machine assignment. During the local search for the best move, only those moves that preserve the *WSPT–SPT* order in every machine are considered. After that the best neighborhood schedule $\Pi_{B\tau}$ is selected, the neighborhood generation scheme is then applied again, starting from $\Pi_{B\tau}$. This procedure is iterated until a $\Pi_{B\tau}$ that satisfies a user-defined termination condition is obtained. Moreover, schedule Π_τ is an initial schedule for TS, and the initial solution often influenes the quality of the final solution (Kim & Shin, 2003). Thus, function F can be obtained by

$$F(\tau) = L(\Pi_{B\tau}) \tag{5}$$

Clearly $G(\tau) \geqslant F(\tau)$ for all $\tau \in H$. Furthermore, the execution of swap moves does not result in a change in the number of jobs on machines. A swap move generally has a smaller *move distance* (the change in current solution caused by the execution of a given move) than insert moves. To avoid that the computational effort increases, we present an efficiency of the scheme for generating neighborhood schedules from the current schedule. Thus, a candidate list strategy is designed and based on (Bilge, Kiraç, Kurtulan, & Pekgün, 2004) to restrict the number of solutions examined on a given iteration for insert moves. This strategy is described as follows:

- Calculate the total weighted completion time of each machine schedule $\delta_{i,\tau} \in \Pi_{C\tau}$ described in Eq. (4).
- Choose the machine schedule $\delta_{r,\tau}$ with highest total weighted completion time.
- Consider every job in machine schedule $\delta_{r,\tau}$ for an insertion on any other machine schedule $\delta_{q,\tau}$, $r \neq q$. Note that an insert move can be determined in constant time if it preserves the *WSPT–SPT* order on the receiving machine schedule.

The first objective on BOP is conducted by the interval minimization *G–F*, which occurs on structures near to a local tabu search minimum. Significantly, a direct objective of the problem $FFc|p_{ij} = p_j|\sum_{j=1}^{n} WjCj$ is indirectly reproduced at the (*G–F*) minimization. The second objective guides the evolution process to constructive schemata in structures.

4.3. The evolution process

The evolution process proceeds with an adaptive rejection threshold, which considers the objectives (interval minimization and G maximization) of BOP. The evolution process is guided by a variable $\gamma \geqslant 0$, the following expression presents a condition for elimination of a structure or schema τ from the current population.

$$G(\tau) - F(\tau) \geqslant d \cdot G_{\text{upper}} - \gamma \cdot d[G_{\text{upper}} - G(\tau)] \tag{6}$$

Two expected values are given at the beginning of the process. The first value is a non-negative real number $G_{\text{upper}} > \max_{\tau \in H} G(\tau)$, which is an upper bound on the objective value obtained by generating a structure using a heuristic, ensuring that G_{upper} receives the G evaluation for that structure. The other expected value is the interval length $d \cdot G_{\text{upper}}$ obtained from G_{upper} where d is a real number $0 < d \leqslant 1$.

As initially good schemata must be preserved for recombination, parameter γ starts at zero and increases slowly from generation to generation. The population at evolution time γ, denoted by Γ_γ, is dynamic in size according to the value of the parameter γ. The population can shrink to zero during the evolution process. Rearranging Eq. (6), a structure or schema τ should be discarded from current population if it satisfies

$$\varphi(\tau) = \frac{d \cdot G_{\text{upper}} - [G(\tau) - F(\tau)]}{d[G_{\text{upper}} - G(\tau)]} \leqslant \gamma \tag{7}$$

structures or schemata are given their corresponding rank $\varphi(\tau)$ when they are created. If an offspring resulting from recombination or mutation with a higher rank value is found, then that structure or schema has a higher probability of surviving for recombination in the next generation.

4.3.1. Initial population

The initial population is composed exclusively of schemata, such that for each schema, some random positions are labeled 0 and m (number of machines at each stage) random positions are labeled 1, and the remaining positions are labeled #. For a sequence of generations, the population can increase by adding new offspring generated through recombination or mutation.

4.3.2. Selection and recombination operators

According to Eq. (6), structures or schemata with small gap $[G(\tau) - F(\tau)]$ and/or higher $G(\tau)$ values have more chance of surviving for recombination or mutation. The structures and schemata in population Γ_γ are non-decreasing ordered using the following key:

$$\Delta(\tau) = \frac{1 + d(\tau)}{\eta} \tag{8}$$

where $d(\tau) = [G(\tau) - F(\tau)]/G(\tau)$, η is the number of labels different from # in τ. Thus, structures or schemata with small number of labels # and/or presenting small $d(\tau)$ are considered better and appear in first order positions.

Two structures and/or schemata are selected for recombination. The first is called the base (τ_{base}) and is randomly selected out of the first positions in Γ_γ. The second one is called the guide (τ_{guide}) and is randomly selected out of the total population. Let τ_{new} be the new structure or schema (offspring) that preserves the number of seeds (number of machines at each stage) after performing recombination. The operations of recombination are described as follows (to be executed in this order).

```
Recombination
For each $j \in \{1,\ldots,n\}$ if $\tau_{\text{base}\ j} = \tau_{\text{guide}j}$ then
    set $\tau_{\text{new}\ j} = \tau_{\text{base}\ j}$
For each $j \in \{1,\ldots,n\}$ if $\tau_{\text{guide}\ j} = \#$ then
    set $\tau_{\text{new}\ j} = \tau_{\text{base}\ j}$
For each $j \in \{1,\ldots,n\}$ if $\tau_{\text{base}\ j} = \#$ and $\tau_{\text{guide}\ j} = 0$ then
    set $\tau_{\text{new}\ j} = 0$
For each $j \in \{1,\ldots,n\}$ if $\tau_{\text{base}\ j} = \#$ or 0 and $\tau_{\text{guide}\ j} = 1$
then
    set $\tau_{\text{new}\ j} = 1$ and set $\tau_{\text{new}\ i} = 0$ for some $\tau_{\text{new}\ i} = 1$
```

```
For each $j \in \{1,\ldots,n\}$ if $\tau_{\text{base } j} = 1$ and $\tau_{\text{guide } j} = 0$ then
    set $\tau_{\text{new } j} = 0$ and set $\tau_{\text{new } i} = 1$ for some $\tau_{\text{new } i} = 0$
```

4.3.3. Mutation operator

Mutation process can enhance the diversity and provide the opportunity to escape from local optima. If the selected base is a schema, then it is combined with a guide individual (schema or structure) to generate a new offspring; otherwise, a mutation operator is applied and is compared to the best structure found so far. At each generation, n new individuals (structures and/or schemata) are created, and their ranks are computed and compared with γ, which determines whether they are included in the new population.

```
Mutation
For each position $j$ with label 1 do
    For each position $k$ with label 0 do
        Interchange the labels on position $j$ and $k$ generating an offspring $\tau_{\text{new}}$; (offspring generation)
        Interchange the labels on position $j$ and $k$; (return to the original $\tau$)
    End_for
End_for
```

4.4. The algorithm

Based on the proposed assignment heuristic, tabu search heuristic, population initialization and CGA operators, the HCGA approach begins with recombination procedure (over schemata only) and the constructive process builds structure (full individuals) progressively at each generation. Additionally, the structure can be generated to fill the τ_{base} substituting the # labels for 0 labels before the recombination.

The HCGA framework is proposed as illustrated in Fig. 2. It can be seen that HCGA not only applies the CGA-based evolutionary process to effectively perform exploration for promising solutions, but it also adopts TS simultaneously to perform exploitation for solution

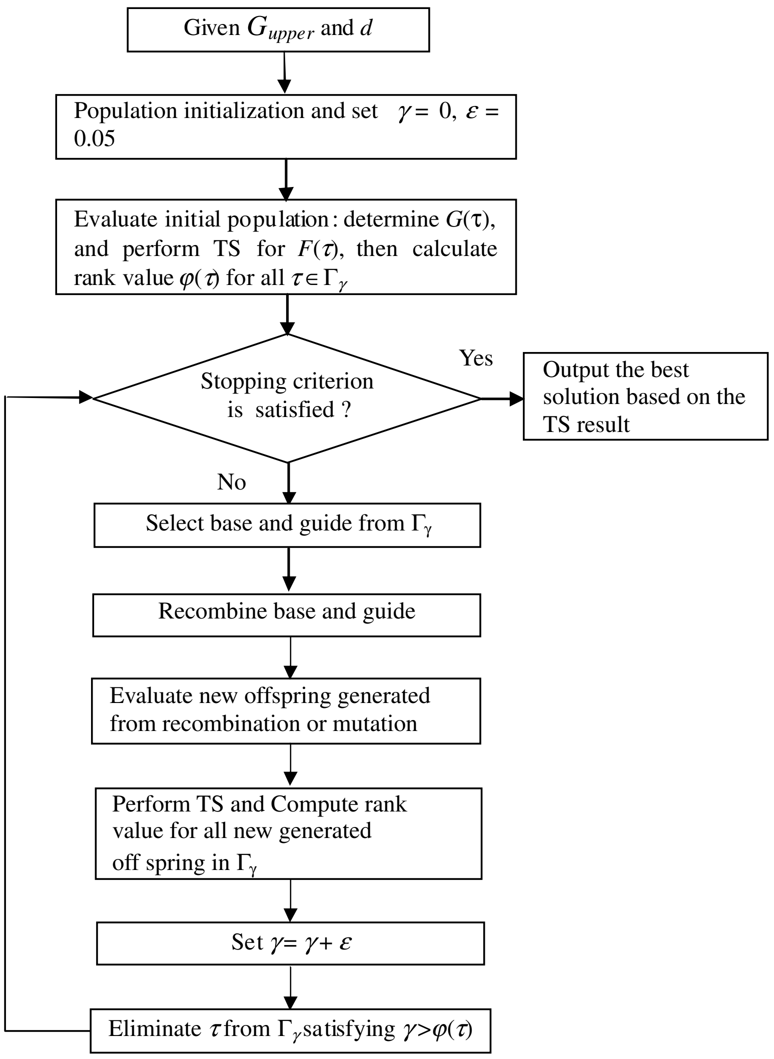
Given G_{upper} and d
Population initialization and set $\gamma = 0$, $\varepsilon = 0.05$
Evaluate initial population: determine $G(\tau)$, and perform TS for $F(\tau)$, then calculate rank value $\varphi(\tau)$ for all $\tau \in \Gamma_\gamma$
Stopping criterion is satisfied ?
Yes
Output the best solution based on the TS result
No
Select base and guide from Γ_γ
Recombine base and guide
Evaluate new offspring generated from recombination or mutation
Perform TS and Compute rank value for all new generated off spring in Γ_γ
Set $\gamma = \gamma + \varepsilon$
Eliminate τ from Γ_γ satisfying $\gamma > \varphi(\tau)$

Fig. 2. Framework of HCGA.

improvement. Two stopping conditions are considered: stop when the population is empty, or at a predefined number of generations.

5. Computational results

5.1. Experimental setup

The HCGA approach described in Section 4 was implemented in C and tested on a Pentium 4 (1.80 GHz). Since no sample problems that could be adopted as benchmarks for measuring the HCGA approach were found in the literature, the test problems were randomly generated to compare with the column generation (CG) approach proposed by Van den Akker et al. (1999) in solving parallel machine scheduling problem. However, the problem $FFc|p_{ij} = p_j|\sum_{j=1}^{n} WjCj$ significantly differs from the parallel-identical-machine scheduling problem with the same objective. Thus, some algorithms in (Van den Akker et al., 1999) need to be redesigned and modified according to the characteristic of a PFS problem in (Shakhlevich et al., 1998) for solving the problem $FFc|p_{ij} = p_j|\sum_{j=1}^{n} WjCj$, and called CG-PFFS. For completeness, CG-PFFS is explained in detail in the Appendix. LINGO 8.0 solver was used to solve the linear programming involved in CG-PFFS. Some HCGA parameters were adjusted for all results presented in Tables 2 and 3. The γ increase interval was set to 0.05 for $0 \leqslant \gamma \leqslant 1$, and 0.025 for $\gamma > 1$, where $d = 0.1$. These parameters avoid the premature termination with an empty population. For each schema in the initial population, 20% of the n ($n \in N$) random positions received Label 0, and 10–20% of better individuals in the population were considered for the selection of the individuals base. The number of individuals initially generated was proportional to the problem length. In addition, the TS is only applied to a population training and the solution improvement. Thus, the search iteration number of TS was fixed at 20, and not too great a computational effort will be consumed. Ideally the situation would be to use greater iteration number; however, this would turn the algorithm very slow. Four other parameters are chosen as follows:

(1) Number of machines $m \in \{3, 4, 5, 8, 12, 16, 20\}$.
(2) Number of jobs $n \in \{20, 30, 40, 50, 60, 80, 100\}$.
(3) The processing time p_j drawn from the uniform distribution [1, 20].
(4) The weights are randomly generated from a uniform distribution [1, 100].

5.2. Results from CG-PFFS

Twenty test problems are randomly generated for each combination of n and m. Furthermore, for a fixed m and n, the number of stages is increased without significantly increasing the time complexity. Therefore, the experiments do not differentiate between the number of stages for the fixed n and m. For simplicity, all other entries shown in Table 1 represent the average performance value based on 20 problems with the number of stages fixed to three for each combination of n and m.

Table 1
Computational results from CG-PFFS

Problem structures $n \times s \times m$	Solved at root node	Number of nodes searched	Max LP–IP gap (%)	Avg CPU time (s)
$20 \times 3 \times 3$	20	0	0.000	2.21
$20 \times 3 \times 4$	19	2	0.008	1.81
$20 \times 3 \times 5$	20	0	0.000	1.18
$30 \times 3 \times 3$	13	1.2	0.001	6.34
$30 \times 3 \times 4$	10	1.8	0.001	5.35
$30 \times 3 \times 5$	15	2.5	0.000	2.81
$40 \times 3 \times 3$	5	8.5	0.001	53.34
$40 \times 3 \times 4$	2	11.5	0.008	15.21
$40 \times 3 \times 5$	8	8.2	0.003	9.09
$50 \times 3 \times 3$	1	13.1	0.001	289.21
$50 \times 3 \times 4$	1	12.3	0.002	81.32
$50 \times 3 \times 5$	2	5.3	0.002	28.11
$60 \times 3 \times 8$	5	18.5	0.003	61.45
$60 \times 3 \times 12$	6	10.5	0.001	22.18
$60 \times 3 \times 16$	8	5.3	0.000	9.31
$60 \times 3 \times 20$	15	2.5	0.000	3.74
$80 \times 3 \times 8$	0	31.8	0.005	496.33
$80 \times 3 \times 12$	0	25.4	0.005	132.27
$80 \times 3 \times 16$	2	18.3	0.001	61.34
$80 \times 3 \times 20$	4	11.5	0.000	26.28
$100 \times 3 \times 8$	0	50.3	0.017	1596.35
$100 \times 3 \times 12$	0	31.5	0.010	503.61
$100 \times 3 \times 16$	0	24.1	0.004	305.02
$100 \times 3 \times 20$	0	19.8	0.000	112.15

Max LP–IP gap = the maximum gap in percentage between the solution value of linear relaxation solved at root node (LP) and the integral solution value (IP) based on 20 problems for each combination of n and m, computed as [(IP–LP)/IP]100%.

Table 1 demonstrates the resulting performance for the PFFS problems of smaller size and larger size. For the problems of smaller size, the numbers of jobs are 20, 30, 40 and 50. It can be seen that the problems are solved at the root node of the linear relaxation problem without branching for 116 problems out of 240. Particularly, the integral solutions of linear relaxation programming are often obtained at the root node with $n \leqslant 30$. Additionally, the maximum gap between the LP and IP is lower than 0.01%, which is extremely small. Furthermore, 85% of the 240 test problems occur in which the integral solution value and lower bound concur in our experiments. For the problems of larger size, the numbers of jobs are extended to 60, 80 and 100, and the numbers of machines are increased to 8, 12, 16 and 20. Clearly, branching is involved more than 80% of these large problems. Moreover, the lower bound value remains tight for most of the test problems; the maximum gap between the LP and IP is also lower than 0.05%.

It has been observed from Table 1 that CG-PFFS yields solutions of superior quality and performs very well when the number of jobs to number of machines is fairly small. However, for a fixed n, given a smaller number of machines m, an ideal machine schedule on a single machine should contain more jobs. Additionally, more columns may have to be produced by the column generation procedure, slowing the algorithm. This finding is consistent with the results obtained by Van den Akker et al. (1999), Chen and Powell (1999) and Lee and Chen (2000) for solving other parallel machine scheduling problems using the CG approach.

5.3. Comparison of CG-PFFS and HCGA

This study has already shown that CG-PFFS produces excellent quality solutions. This subsection demonstrates that the HCGA approach can also obtain solutions as good as those of CG-PFFS, but with less computation time. To test the performance of the proposed HCGA, each combination of n and m was run ten times with the same processing time and weight. Each run was stopped when the best solution of CG-PFFS was reached, or when the population is empty. The average CPU times shown in Tables 2 and 3 represent the average of performance measures for ten runs of the corresponding problem.

Table 2 demonstrates that the deviation between the CG-PFFS and the HCGA best objective values found are equal to 0%. The result of the large-size problem presented in Table 3 shows that the deviation between the CG-PFFS and the HCGA best objective values found are also equal to 0% when $m \leqslant 8$. These results indicate that incorporating the proposed TS into HCGA not only demonstrates the effective utilization in a population training, driving

Table 2
Comparisons of HCGA, CG-PFFS and PTS for problems with $n = 20$, 30, 40, and 50

$n \times s \times m$	[a]CG-PFFS	%Dev		Avg CPU time (s)		
		HCGA	PTS	CG-PFFS	HCGA	PTS
$20 \times 3 \times 3$	41,129	0.000	0.000	2.4	2.7	2.2
$20 \times 3 \times 4$	34,534	0.000	0.000	1.7	5.2	3.1
$20 \times 3 \times 5$	29,678	0.000	0.000	1.5	7.5	3.9
$30 \times 3 \times 3$	69,179	0.000	0.000	6.7	6.4	3.8
$30 \times 3 \times 4$	55,008	0.000	0.000	5.2	9.3	5.2
$30 \times 3 \times 5$	46,543	0.000	0.000	2.7	13.2	6.9
$40 \times 3 \times 3$	196,381	0.000	0.000	51.2	8.4	5.8
$40 \times 3 \times 4$	152,822	0.000	0.137	14.3	13.6	8.5
$40 \times 3 \times 5$	126,676	0.000	0.159	10.1	18.5	12.3
$50 \times 3 \times 3$	191,723	0.000	0.000	293.5	13.7	8.2
$50 \times 3 \times 4$	148,945	0.000	0.184	78.3	19.6	13.3
$50 \times 3 \times 5$	123,295	0.000	0.493	26.7	26.2	20.6
Average		0.000	0.081	41.2	12.0	7.8

%Dev = the deviation in percentage between the CG-PFFS and the HCGA/PTS best objective value found, computed as [(HCGA/PTS best objective value – [a]CG-PFFS) × 100]/(HCGA/PTS best objective value).
[a] CG-PFFS = the best objective value found by CG-PFFS.

Table 3
Comparisons of HCGA, CG-PFFS and PTS for problems with $n = 60$, 80, 100

$n \times s \times m$	[a]CG-PFFS	% Dev		Avg CPU time(sec)		
		HCGA	PTS	CG-PFFS	HCGA	PTS
$60 \times 3 \times 4$	227,014	0.000	0.088	585.1	27.5	21.4
$60 \times 3 \times 8$	126,958	0.000	0.588	63.7	40.1	29.1
$60 \times 3 \times 12$	94,058	0.325	0.649	23.2	65.5	41.5
$60 \times 3 \times 16$	77,708	0.543	0.767	11.4	95.4	53.3
$80 \times 3 \times 4$	319,033	0.000	0.236	1761.5	36.4	32.2
$80 \times 3 \times 8$	174,642	0.000	0.419	506.3	65.3	48.8
$80 \times 3 \times 12$	126,958	0.378	0.928	138.5	92.8	67.4
$80 \times 3 \times 16$	103,373	0.451	1.121	60.9	122.6	90.4
$80 \times 3 \times 20$	89,474	0.745	1.583	28.8	238.8	121.3
$100 \times 3 \times 8$	337,979	0.233	0.413	1611.3	79.3	73.9
$100 \times 3 \times 12$	240,223	0.365	1.075	515.5	155.8	112.3
$100 \times 3 \times 16$	191,549	0.627	1.642	312.9	247.6	174.2
$100 \times 3 \times 20$	162,707	1.045	2.226	125.8	498.8	244.8
Average		0.362	0.903	441.9	135.8	85.4

%Dev = the deviation in percentage between the CG-PFFS and the HCGA/PTS best objective value found, computed as [(HCGA/PTS best objective value – [a]CG-PFFS) × 100]/(HCGA/PTS best objective value).
[a] CG-PFFS = the best objective value found by CG-PFFS.

the evolutionary process in an appropriate direction and the fast convergence, but it also performs exploitation for solution improvement. This can be explained that the new offspring generated by recombination or mutation were considered as good structures or schemata and also good quality of the initial schedules for TS during the evolution process. In general, the performance of TS is largely affected not only by the efficiency of the scheme for generating neighborhood schedules from the current schedule, but also by the quality of the initial schedule (Kim & Shin, 2003).

Significantly, the average CPU time consumed by HCGA is much lower than that of CG-PFFS and does not increase greatly as the problem size increases, especially when the ratio of number of jobs to number of machines is fairly high. Notably, the gap result of HCGA is slightly worse than that of CG-PFFS when $m \geqslant 12$. However, the maximum %Dev in Table 3 is 1.045%, and is very close to the solution of CG-PFFS.

5.4. Comparison of HCGA and pure tabu search

To further demonstrate the effectiveness of HCGA, an additional simulation is carried out to compare the HCGA with a pure tabu search (PTS) algorithm proposed by Barnes and Laguna (1993). We terminate the algorithm if there is no improvement to the best solution obtained after 100 iterations or if the maximum number of iterations reaches at 1000. To generate an initial schedule for PTS, a simple heuristic is applied by assigning the jobs that preserve the *WSPT–SPT* order randomly to the machines, where an earlier available machine has a higher probability of getting the next job on the list, until all jobs are scheduled.

Tables 2 and 3 demonstrate that the both algorithms obtained the same objective values when $n \leqslant 30$. However, the %Dev values from HCGA are much better than those obtained by PTS when $n \geqslant 40$, which show the effectiveness of incorporating the TS into CGA.

6. Conclusions

This study has investigated the HCGA approach for minimizing total weighted completion time in a proportionate flexible flow shop problem. To the best of our knowledge, this is the first attempt to deal with the parallel machine scheduling problems using the CGA approach. The proposed HCGA applied the evolutionary searching mechanism of CGA characterized by population training to effectively perform exploration. Moreover, HCGA adopted the TS to perform exploitation for solution improvement. Additionally, the proposed candidate list strategy for TS not only increases the speed of the search, but also improves the solution quality. The experimental results demonstrate the robustness of the proposed HCGA in terms of solution quality and the average computation time. In particular, HCGA seems to be superior when the ratio of number of jobs to number of machines is fairly high. Accordingly, HCGA can be considered as another effective approach for other parallel machine scheduling problems to minimize makespan, tardiness, maximum completion time or maximum lateness. Moreover, the CGA concepts and properties are independent on the representation and decoders used, i.e., different representations of the CGA application can be designed for various machine scheduling problems. Further research encourages the extension of CGA incorporating other local search methods for job shop and multi-objective scheduling problems.

Appendix. Modifications in (Van den Akker et al., 1999) for solving the problem $FFc|p_{ij} = p_j|\sum_{j=1}^{n} WjCj$

This appendix describes some modifications to be made in (Van den Akker et al., 1999) for solving the problem $FFc|p_{ij} = p_j|\sum_{j=1}^{n} WjCj$, based on ideas provided by Shakhlevich et al. (1998). The following lemma describes the property associated with an optimal schedule of the PFS problem to minimize WCT in (Shakhlevich et al., 1998). The lemma is adopted later to restrict the search space for finding the optimum schedule of the problem $FFc|p_{ij} = p_j|\sum_{j=1}^{n} WjCj$ by using the column generation approach.

Lemma. *There exists an optimal schedule, if for two jobs j and k both $w_j/p_j \geqslant w_k/p_k$ and $p_j \leqslant p_k$, then job jprecedes job k on the same machine.*

According to the proof of the Lemma, an interchange argument (IA) is applied to obtain a schedule that satisfies the rule described in Lemma. The IA is stated as follows:

Step 1: First reindex all jobs $(1, 2, \ldots, n)$ in the WSPT order.
Step 2: Let job j be the job with the smallest processing time, but it is not scheduled according to its position in the shortest processing time (SPT) order; assume job k occupies that position in the current sequence.
Step 3: Move job j immediately in front of job k, but after all jobs that precede job j in the SPT order.
Step 4: Proceed steps 2 and 3, and eventually produce a sequence that satisfies the Lemma.

After the execution of IA, let $P = \sum_{j=1}^{n} p_j + (s-1)\max\{p_1, \ldots, p_n\}$ denote the total processing time of the jobs. Thus the dynamic programming (DP) is designed for solving a single machine subproblem, and the DP described by (Van den Akker et al., 1999) can be modified as follows:

Let σ represent any machine schedule on a single machine and $F_j(t)$ denote the minimum reduced cost of a machine schedule consisting of a subset of jobs $\{1, 2, \ldots, j\}$ that satisfies the Lemma, where job j is the current last job and is completed at time$t$ at stage s. For machine schedule that realizes $F_j(t)$, there are two possibilities: either job j is not part of it, or the machine schedule contains job j. The first possibility is that the machine schedule with the first $j-1$ jobs completed at time t is selected; the value of this solution can be determined by $F_{j-1}(t)$. The second possibility is then to include job j in the machine schedule for the first $j-1$ jobs that completes at time $t-\mathrm{sp}_j+(s-1)p_{j-1}$; the value of this solution then can be determined by $F_{j-1}(t-\mathrm{sp}_j+(s-1)p_{j-1})+w_jt-\lambda_j$. Initialization

$$F_j(t)=\begin{cases}-\lambda 0, & \text{if } j=0 \text{ and } t=0\\ \infty, & \text{otherwise}\end{cases}$$

Recurrence relation

For $j=1,\dots,n, t=0,\dots,P$,

$F_j(t)=\min\{F_{j-1}(t), F_{j-1}(t-\mathrm{sp}_j+(s-1)p_{j-1})+w_jt-\lambda_j\}$, $s\in S$

The value is determined by solving:

$\min_{0\leqslant t\leqslant P} F_n(t)$

The next modification describes the branch-and-bound algorithm applied when a fractional solution is obtained. The following theorem, described by (Van den Akker et al., 1999), converts a fractional linear relaxation programming solution into an integral solution with the same objective solution. Let $C_j(\sigma)$ denote the completion time of job j in σ.

Theorem. *If $C_j(\sigma)=C_j$ for each job $j(j=1,\dots,n)$ and for each σ with $x^*_\sigma>0$, then the schedule obtained by processing job j in the time interval $[C_j-p_j, C_j]$ is feasible and has minimum cost.*

As shown above, the jobs have been indexed to satisfy the Lemma in each machine schedule σ generated by performing the dynamic programming algorithm for solving a subproblem. Thus, the completion time of job j in σ equals the starting time on that machine at the first stage plus sp_j. Therefore, the time interval $[C_j-p_j, C_j]$ described in Theorem can be modified and be rewritten as $[C_j-\mathrm{sp}_j, C_j]$.

If the optimal solution to the linear relaxation programming neither is integral nor satisfies the conditions of the modified Theorem, then a branch and bound algorithm is applied to find an integral solution. Moreover, according to the description in (Van den Akker et al., 1999), two descendant nodes are created such a job j a fractional job: one for the condition that $C_j\leqslant\min\{C_j(\sigma)|x^*_\sigma>0\}$ and other one for the condition that $C_j\geqslant_{\min}\{C_j(\sigma)|x^*_\sigma>0\}+1$. The first constraint is that the job j must be completed at deadline d_j at stage $s\in S$. The second constraint specifies a release date $r_j=\min\{C_j(\sigma)|x^*_\sigma>0\}+1-\mathrm{sp}_j$. Both constraints can be easily incorporated into the dynamic programming algorithm. Therefore, the recurrence relation described in above can be rewritten as follows:

$$F_j(t)=\begin{cases}\min\{Fj-1(t), Fj-1(t-spj+(s-1)pj-1)+wjt-\lambda j\}, & \text{if } rj+spj\leqslant t\leqslant dj,\\ Fj-1(t), & \text{otherwise}\end{cases}$$

Finally, after an integral solution has been determined, let Ω is the set containing $m$machine schedules σ, and reconstruct the jobs in each $\sigma\in\Omega$ to form the *WSPT–SPT* order using Shakhlevich's WSPT-MCI algorithm described in Section 4.1.

References

Adenso-Díaz, B. (1992). Restricted neighborhood in the tabu search for the flowshop problem. *European Journal of Operational Research, 62*(1), 27–37.

Allahverdi, A. (1996). Two-machine proportionate flowshop scheduling with breakdowns to minimize maximum lateness. *Computers and Operations Research, 23*(10), 909–916.

Barnes, J. W., & Laguna, M. (1993). Solving the multiple-machine weighted flow time problem using tabu search. *IIE Transactions, 25*, 121–128.

Bilge, Ü., Kiraç, F., Kurtulan, M., & Pekgün, P. (2004). A tabu search algorithm for parallel machine total tardiness problem. *Computers and Operations Research, 31*, 397–414.

Chang, P. C., Chen, S. H., & Lin, K. L. (2005). Two-phase population genetic algorithm for parallel machine-scheduling problem. *Expert Systems with Applications, 29*, 705–712.

Chen, Z. L., & Powell, W. B. (1999). Solving parallel machine scheduling problems by column generation. *INFORMS Journal on Computing, 11*(1), 78–94.

Chiang, T. C., Chang, P. Y., & Huang, Y. M. (2006). Scheduling multiprocessor tasks with resource and timing constraints using particle swarm optimization. *International Journal of Computer Science and Network Security, 6*(4), 71–77.

Choi, B. C., Yoon, H., & Chung, S. J. (2006). Minimizing the total weighted completion time in a two-machine proportionate flow shop with different machine speeds. *International Journal of Production Research, 44*(4), 715–728.

de Oliveira, A. C. M., & Lorena, L. A. N. (2002). A constructive genetic algorithm for gate matrix layout problems. *IEEE Transactions on Computer-Aided design of inegrated circuits and systems, 21*, 969–974.

Edwin Cheng, T. C., & Shakhlevich, N. (1999). Proportionate flow shop with controllable processing times. *Journal of Scheduling, 2*, 253–265.

Funda, S. S., & Ulusoy, G. (1999). Parallel machine scheduling with earliness and tardiness penalties. *Computers and Operations Research, 26*(8), 773–787.

Garey, M., Johnson, D., & Sethi, R. (1976). The complexity of flow shop and job shop scheduling. *Mathematics of Operations Research, 1*, 117–129.

Goldberg, D. E., Deb, K., Kargupta, H., & Harik, G. (1993). Rapid, accurate optimization of difficult problems using fast messy genetic algorithm. IlliGAL Rpt No. 93,004, Illinois Genetic Algorithm Lab., Dept of General Engineering, University of Illinois, Urbana.

Goldberg, D. E., Korb, B., & Deb, K. (1989). Messy genetic algorithm: motivation, analysis, and first results. *Complex Systems, 3*, 493–530.

Gupta, J. N. D., Hariri, A. M. A., & Potts, C. N. (1997). Scheduling a two-stage hybrid flow shop with parallel machines at the first stage. *Annals of Operations Research, 69*, 171–191.

Holland, J. H. (1992). *Adaptation in natural and artificial systems* (1st ed.). Massachusetts: MIT press.

Hou, S., & Hoogeveen, H. (2003). The three-machine proportionate flow shop problem with unequal machine speeds. *Operations Research Letters, 31*(3), 225–231.

Kim, K. J., & Han, I. (2003). Application of a hybrid genetic algorithm and neural network approach in activity-based costing. *Expert Systems with Applications, 24*, 73–77.

Kim, C. O., & Shin, H. J. (2003). Scheduling jobs on parallel machines: a restricted tabu search approach. *International Journal of Advanced Manufacturing Technology, 22*, 278–287.

Lee, C. Y., & Chen, Z. L. (2000). Scheduling jobs and maintenance activities on parallel machines. *Naval Research Logistics, 47*(2), 145–165.

Liu, B., Wang, L., & Jin, Y. H. (in press). An effective hybrid particle swarm optimization for no-wait flow shop scheduling. *International Journal of Advanced Manufacturing Technology*.

Lorena, L. A. N., & Furtado, J. C. (2001). Constructive genetic algorithm for clustering problems. *Evolutionary Computation, 9*(3), 309–327.

Moursli, O., & Pochet, Y. (2000). A branch-and-bound algorithm for the hybrid flow shop. *International Journal of Production Economics, 64*, 113–125.

Noorul Hag, A., Ravindran, D., Aruna, V., & Nithiya, S. (2004). A hybridization of metaheuristics for flow shop scheduling. *International Journal of Advanced Manufacturing Technology, 24*, 376–380.

Oğuz, C., & Ercan, M. (2005). A genetic algorithm for hybrid flow-shop scheduling with multiprocessor tasks. *Journal of Scheduling, 8*(4), 323–351.

Oğuz, C., Ercan, M., Edwin Cheng, T. C., & Fung, Y. F. (2003). Heuristic algorithms for multiprocessor task scheduling in a two-stage hybrid flow-shop. *European Journal of Operational Research, 149*, 390–403.

Ow, P. S. (1985). Focused scheduling in proportionate flow shops. *Management Science, 31*, 852–869.

Pinedo, M. L. (1985). A note on stochastic shop models in which jobs have the same processing requirements on each machine. *Management Science, 31*, 840–845.

Pinedo, M. L. (2002). *Scheduling: theory, algorithms, and systems* (2nd ed.). Englewood Cliffs, NJ: Prentice-Hall.

Ribeiro, G., & Lorena, L. A. N. (2001). A constructive evolutionary approach to school timetabling. *Lecture Notes in Computer Science, 2037*, 130–139.

Şerifoğlu, F. S., & Ulusoy, G. (2004). Multiprocessor task scheduling in multistage hybrid flow-shops: a genetic algorithm approach. *Journal of the Operational Research Society, 55*(5), 504–512.

Shakhlevich, N. V., Hoogeveen, H., & Pinedo, M. L. (1998). Minimizing total weighted completion time in a proportionate flow shop. *Journal of Scheduling, 1*, 157–168.

Smith, W. E. (1956). Various optimizers for single-stage production. *Naval Research Logistics Quarterly, 3*, 59–66.

Tang, L. X., Xuan, H., & Liu, J. (2006). A new Lagrangian relaxation algorithm for hybrid flowshop scheduling to minimize total weighted completion time. *Computers and Operations Research, 33*, 3344–3359.

Tozkapan, A., Kirca, Ö., & Chung, C. S. (2003). A branch and bound algorithm to minimize the total weighted flowtime for the two-stage assembly scheduling problem. *Computers and Operations Research, 30*(2), 309–320.

Van den Akker, J. M., Hoogeveen, J. A., & Van de Velde, S. L. (1999). Parallel machine scheduling by column generation. *Operations Research, 47*, 862–872.

Wang, L., & Zheng, D. Z. (2003). An effective hybrid heuristic for flow shop scheduling. *International Journal of Advanced Manufacturing Technology, 21*(1), 38–44.