Hybrid flow shop scheduling with finite buffers

Zhonghua Han

Faculty of Information and Control Engineering,
Shenyang Jianzhu University,
Shenyang, China
and
Department of Digital Factory,
Shenyang Institute of Automation,
Chinese Academy of Sciences,
Shenyang, China
Email: xiaozhonghua1977@163.com

Yue Sun*

Faculty of Information and Control Engineering,
Shenyang Jianzhu University,
Shenyang, China
Email: SUN_YUE_1212@163.com
*Corresponding author

Xiaofu Ma

Department of Electrical and Computer Engineering,
Virginia Tech,
Blacksburg, USA
Email: xfma@vt.edu

Zhe Lv

Faculty of Information and Control Engineering,
Shenyang Jianzhu University,
Shenyang, China
Email: m18509486051@163.com

Abstract: In this paper, the scheduling problem for hybrid flow shop is investigated with the consideration of the finite buffers. Different from the existing works which focus on the makespan minimisation under the constraints of given intermediate buffer sizes, this paper investigates how to decide the reasonable size of the intermediate buffer. Firstly, the problem is modelled with the design of the accurate and reasonable buffer space without affecting the flow shop production efficiency. Then, a hybrid heuristic method is proposed to reduce the algorithm complexity. Specifically, the buffer size is estimated based on the theory of probability distribution, and the searching for the global optimal solution is processed by using a novel and effective self-adaptive differential evolution algorithm. The proposed algorithm can adjust parameters intelligently for stopping unnecessary iterations as well as avoiding the stagnant situation of the local optimum. Lastly, a wide range of practical scenarios are considered for algorithm evaluation, and the numerical results show that the proposed approach is effective on: 1) reducing the buffer size; 2) guaranteeing the hybrid flow shop efficiency.

Keywords: hybrid flow shop; buffer size; probability distribution; self-adaptive differential evolution algorithm.

Reference to this paper should be made as follows: Han, Z., Sun, Y., Ma, X. and Lv, Z. (2018) 'Hybrid flow shop scheduling with finite buffers', *Int. J. Simulation and Process Modelling*, Vol. 13, No. 2, pp.156–166.

Biographical notes: Zhonghua Han received his PhD at the Shenyang Institute of Automation, China in 2014. He is currently a Professor with the Faculty of Information and Control Engineering, Shenyang Jianzhu University, Shenyang, China. His main research includes production and operation management, integrated technology of automation system in enterprise, and the engineering application research of production scheduling method.

Yue Sun received her Bachelors from the Faculty of Information and Control Engineering, Shenyang Jianzhu University, Shenyang, China, and in the same place, she currently does her graduate study under the supervising of Zhonghua Han. Her main research area is production scheduling.

Xiaofu Ma received his BS in Electronics from Northwest University, Xi'an, China, in 2008 and his MS in Computer Science from Tongji University, Shanghai, China, in 2011. Currently, he is working toward his PhD in Electrical Engineering at the Virginia Polytechnic Institute and State University, Blacksburg, VA, USA. His research interests include network optimisation, image and signal processing, machine learning, cognitive radio, and wireless healthcare.

Zhe Lv received his Bachelors degree from the Faculty of Information and Control Engineering, Shenyang Jianzhu University, Shenyang, China, and in the same place, he currently does his graduate study under the supervising of Prof. Zhonghua Han. His main research areas are integrated technology of automation system in enterprise and intelligent control.

1 Introduction

The classical flow shop problem has been investigated from many perspectives and can be extended to many derived problems. One of them is the hybrid flow shop scheduling problem (HFSP) which is very important and challenging in the field of industry automation control. HFSP allows parallel machines in certain stages to process multiple jobs. The goal of HFSP is to arrange the machine assignment and the processing order of the jobs on the same machine smartly such that the finishing time of the jobs can be optimised. Some of the theoretical works focus on such scheduling models (Han et al., 2012, 2016; Eskandari and Hosseinzadeh, 2014; Komaki et al., 2015; Arfi et al., 2016). In those works, the buffers in between stages are not considered. In order to apply flow shop schedule algorithms for applications, analysing buffer size is significant for the real-world system deployment. In such a real-world flow shop system (Agnetis et al., 1998), one job, after being fully processed on the current machine, is either delivered immediately to one machine in the next processing stage, or stored in the intermediate buffer between those two machines if the assigned machine in the next stage is not available currently. If the buffer is completely occupied, the job has to stay on its current machine, and thus other jobs would be blocked until the buffer is fully or partially available (Smutnicki, 1998; Qian et al., 2006; Irohara, 2010).

Obviously, unreasonable intermediate buffer size would result in inefficient production process and even system stagnation if the job task scale is large and/or the jobs on consecutive stages are unbalanced. Therefore, the configuration of reasonable intermediate buffer size becomes one of the key techniques to avoid system process blocking. On the other side, the increasing job diversity increases the uncertainty of production lines. Thus, the intelligent configuration of the buffer size is thus becoming very important for the modern production enterprises.

Application fields of HFSP with finite intermediate buffer include many industrial areas (Ruiz and Vázquez-Rodríguez, 2010), such as the production line of automobile (Rane et al., 2017), steel, tobacco, semiconductor, and oil supply. Specifically, it is not only used to store the works in process (WIP), but also used for re-ordering WIP to get a better processing sequence for the subsequent operation. If the buffer size is too small, machine blocking may happen; whereas too large buffer is also a waste of storage resources. Both the above two cases should be avoided, and selecting a reasonable intermediate buffer could increase the production efficiency and reduce the system cost.

2 Related works

As a widely studied scheduling problem, HFSP has been gaining considerable attention, and numerous research works exist in literature. A comprehensive review of related research on HFSP, including modelling, definition, processing complexity, scheduling criterion and methods, can be found in the work of Naderi et al. (2014). Specifically, as one vital research branch, finite intermediate buffer consideration becomes one of the most important topics for HFSP studies.

It is well known that even a relatively simple HFSP, such as a two-stage HFSP with any stage that uses more than one machine, is NP-hard (Gupta et al., 1988). Adding finite buffer constraints to HFSP mades this derived problem more complex. Researchers have been working heuristic approaches. Akrami et al. (2006) investigated a common cycle economic lot sizing and scheduling problem in flexible flow shops with limited intermediate buffers by developing two heuristic algorithms. Wang and Tang (2009) studied the hybrid flow shop scheduling with finite intermediate buffer, and a tabu search heuristic method was proposed to minimise the sum of weighted completion time of all jobs. Yaurima et al. (2009) took additional limited buffer constraints into account in hybrid flow shop environment, and used a modified genetic algorithm to finish the scheduling. Abyaneh and Zandieh (2012) studied bi-objective HFSP with sequence-dependent setup times and limited buffers and showed that the non-dominated sorting genetic algorithm and sub-population genetic algorithm were effective for solving this problem. A study by Wang (2012) illustrated how to optimise the production sequences in order to minimise the makespan in hybrid flow

shops with limited intermediate buffers, and an improved memetic algorithm (IMA) was proposed. A simulated annealing method combined with some embedded heuristic was verified to be excellent to solving cyclic HFSP with limited buffers and machine eligibility constraints (Soltani and Karimi, 2015). Tran and Ng (2013) studied the multi-objective flexible flow shop scheduling problems with limited intermediate buffers, and a hybrid water flow algorithm was proposed. Li and Pan (2015) presented a novel hybrid algorithm that combined the artificial bee colony (ABC) with tabu search (TS) to solve large-scale HFSP with limited buffers.

Different from the existing works which focus on the makespan minimisation under the constraints of given intermediate buffer sizes, this paper investigates how to decide the reasonable size of the intermediate buffer. Specifically, probability and statistical methods are jointly considered to design the heuristic algorithm for HFSP with intermediate buffer configuration. The outline of the paper is as follows. The problem is described in Section 3. Section 4 presents our proposed algorithms. The simulation results are shown in Section 5. Finally, Section 6 includes the paper.

3 HFSP problem description and formulation

3.1 Problem description

The description of HFSP is illustrated in Figure 1. There are n jobs to be processed in m stages. Among the m stages, at least in one stage, there are at least two parallel machines. Each job needs to be assigned one machine in each stage, and the job processing time on different machines in the same stage may be different. There are intermediate buffers between stages. After a job is processed completely on the current stage, it will join the waiting list in the next stage. If this intermediate buffer is full and none of the machines in the next stage is available, the job stays on the current machine, which would possibly block other jobs' processing until there are available spaces for other jobs. The processing time of a job on a stage includes the preparation time (for example, setting machine parameters, changing tools, etc.), and the transfer time for jobs between contiguous states. In this paper, the processing time is considered to be deterministic and known to the scheduler. The model parameters are listed in Table 1. In our analysis, the other rules for HFSP are:

1 one machine can only process one job at a time, and once a job has started to be processed on one machine, the procedure cannot be interrupted

2 all the machines are capable of processing each job, and there is no job or machine priority

3 all machines can work continuously, and breakdowns are not considered.

Table 1 Parameters in the model

Parameters	*Specification*
n	Quantity of jobs to be scheduled
m	Quantity of stages
J_i	Job i
M_j	Quantity of machines in each stage
k	Machine number
$WS_{j,k}$	Machine k of stage j, $j \in \{1, 2,\ldots, m\}$, $k \in \{1, 2, \ldots, M_j\}$
Bu_j	The buffer of stage j, $j \in \{1, 2,\ldots, m-1\}$
bc_j	The buffer size of buffer Bu_j, $j \in \{1, 2,\ldots, m-1\}$
$bu_{j,l}$	Storage position l of buffer Bu_j, $j \in \{1, 2,\ldots, m-1\}$, $l \in \{1, 2,\ldots, bc_j\}$
$S_{i,j,k}$	The start time to process job i at stage j on machine k
$S_{i,j,k}$	The completion time to process job i at stage j on machine k
$Tw_{i,j,k}$	The processing time of job i at stage j on machine k
$C_{\max}$	The maximum completion time of all jobs at the last stage
$Td_{i,j,k}$	The departure time of job i on machine k in stage j
$Ta_{j,k}$	The release time of machine k in stage j

3.2 Problem formulation

The basic HFSP can be described as

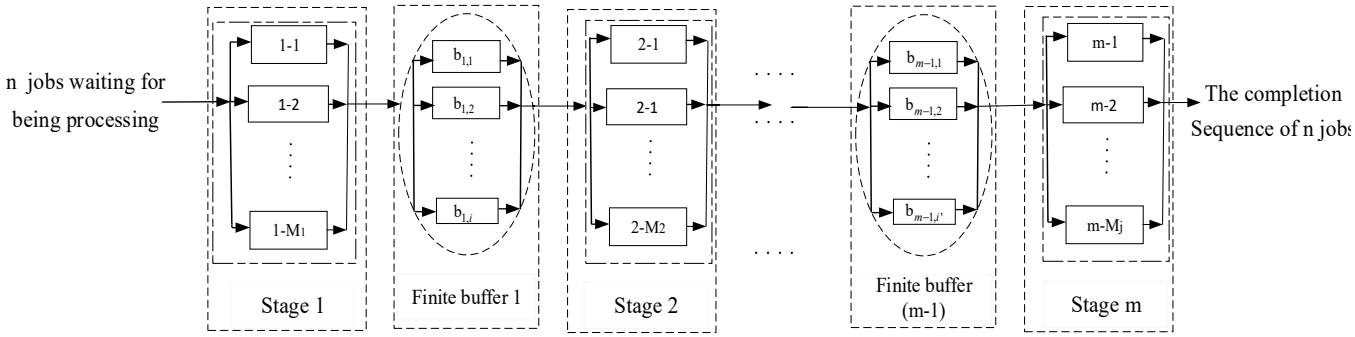
$$\min C_{\max} = \max\left(C_{1,m,k}, C_{2,m,k}, \ldots, C_{n,m,k}\right) \tag{1}$$

$$\text{s.t. } C_{i,j,k} = S_{i,j,k} + Tw_{i,j,k},\ j \in \{1,2,\cdots,m\},\ i \in \{1,2,\cdots,n\} \tag{2}$$

$$C_{i,j,k} \le S_{i,j+1,k'},\ i \in \{1,2,\cdots,n\},\ j \in \{1,2,\cdots,m-1\},\ k \in \{1,\ldots,M_j\},\ k' \in \{1,\ldots,M_{j+1}\} \tag{3}$$

where equation (1) is the optimisation goal, the maximum completion time in hybrid flow shop. Equation (2) constrains the relationship among start time, processing time and completion time for each job in each stage. Equation (3) constrains each job's start time, processing time and completion time in continuous processing stage. The consideration of intermediate buffers and buffer size will be investigated in the next section with our proposed algorithms.

Figure 1 Model of HFSP with finite intermediate buffer



4 Algorithm for HFSP with finite buffers

This section firstly describes the use of the ROV method to encode the individual which is used to represent the solution in Section 4.1. The detailed procedure for constructing the complete solution is shown in Section 4.2. Then, we describe the hybrid heuristic algorithm, SADE-BSEA, which mainly includes two parts, self-adaptive differential evolution (SADE) algorithm and buffer size estimation algorithm (BSEA). Our contribution of the SADE algorithm is described in Section 4.3, and the process of estimating buffer size by the BSEA algorithm is presented in Section 4.4.

4.1 Solution representation

In order to apply the optimisation algorithm to the discrete problem, this paper adopts ranked order value (ROV) method to encode the individual which is the HFSP solution vector representation. The ROV-based encoding rule allows convert the vector component values into job sequences. Specifically, the smallest component in the vector is set to be 1, and the second smallest is set to be 2, and so on, until all of the component assignments are complete.

4.2 Procedures for constructing a complete solution

This section uses n-dimensional integer sequences $A = (a_1, a_2, \ldots, a_n)$ to describe the sequence of the waiting processing queue of all the jobs to be processed on each stage. The first stage sequence is the processing sequence assigned by the individual in population which is the collection of individuals. On the following stages, the waiting processing sequences of jobs will be based on the FIFO principle. That is to say, the job that is completed first in one stage will be processed in priority before the next stage. Machine selection is based on first available machine (FAM) rule. In addition, since there is the intermediate buffer restriction, decoding process which is the inverse process of encoding need take the machine blocking issue into consideration. This decoding method integrates with FAM rule and the shortest processing time (SPT) rule. This decoding method in this paper is called hybrid heuristic decoding methods (HHDM). The following steps describe the process of obtaining the complete solution.

Step 1 Assume that there is an intermediate buffer with bc_j storage positions which can also be considered as machines with zero processing time. The buffer can be considered as a stage with bc_j identical parallel machines. Therefore, the HFS scheduling problem with finite intermediate buffer can be converted into one HFS scheduling problem with $2m - 1$ stages but without intermediate buffer. On this basis, parallel machines set of $2m - 1$ stages is $\{M_1, b_1, M_2, b_2, \ldots, M_{j-1}, b_{j-1}, M_j\}$, where M'_{jj} indicates parallel machine quantity in stage $jj (jj \in \{1, 2, \ldots, 2m - 1\})$ of above new models, kk, $(kk \in \{1, 2, \ldots, M'_{jj}\})$ indicates machine number in new model and A_{jj} indicates waiting processing sequence of stage jj.

In this production process, when job J_i is completed in the current processing stage, if there is no free processing machine at the next stage, this job will be blocked at the current machine.

Step 2 jj is set to 1 and the available time in initial time of all machines in each stage is zero.

Step 3 A_{jj} is the ordered waiting processing sequence online which is specified by population when $jj = 1$. However, when $jj \neq 1$, A_{jj} is a waiting processing sequence of the completion time of previous stage $jj - 1$. Previous M'_{jj} jobs in A_{jj} will be assigned to the stage jj and the SPT machine as the selecting principles. Starting processing time $S_{i,jj,kk}$, completing processing time $C_{i,jj,kk} = S_{i,jj,kk} + Tw_{i,jj,kk}$, departure machine time $Td_{i,jj,kk} = C_{i,jj,kk}$ of job J_i for machine $WS_{jj,kk}$ and release time $Ta_{jj,kk} = Td_{i,jj,kk}$ of machine $WS_{jj,kk}$ in stage jj of job J_i will be recorded.

Step 4 According to FAM rule and the SPT rule, the job J_i with the earliest completion time from the stage jj will be selected to be assigned to the appropriate machine on stages $jj + 1$ and $jj + 2$.

(Each three stages jj, $jj + 1$ and $jj + 2$ will be processed together and stage $jj + 1$ is used to simulate buffer which is between stages jj and $jj + 2$). The start time $S_{i,jj+1,kk}$ and $S_{i,jj+2,kk}$, the completion time $C_{i,jj+1,kk}$ and $C_{i,jj+2,kk}$,

and the departure time $Td_{i,jj+2,kk} = C_{i,jj+2,kk}$ will be recorded together.

Step 5 i is set to $i + 1$. If $i \le n$, according to FAM rule, the job J_i is assigned to the appropriate machine $WS_{jj,kk}$ of stage jj for processing. Its start time is $S_{i,jj,kk} = \max(Td_{i,jj-1,kk}, Ta_{jj,kk})$. The start time $S_{i,jj,kk}$, ending time $C_{i,j,k}$ and departure time $Td_{i,jj,kk}$ of the machine $WS_{jj,kk}$ are recorded. If there are one or more available machines in $jj + 1$ or $jj + 2$ stage, it is $Td_{i,jj,kk} = C_{i,jj,kk}$, otherwise, another condition is that $Td_{i,jj,kk} = \min(Ta_{i,jj+1,kk}, Ta_{i,jj+2,kk})$, the releasing time of machine $WS_{jj,kk}$ is $Ta_{jj,kk} = Td_{i,jj,kk}$. Repeat the Step 4 and Step 5 until $i > n$, otherwise, go to Step 6.

Step 6 There are still $M'_{jj} - 1$ jobs which have not been assigned to the stage $jj + 1$ and $jj + 2$ on stage jj. The job J_i which has been finished earliest on stage jj will be selected from $M'_{jj} - 1$ jobs, based on FAM and SPT. The job J_i is assigned to the appropriate machine of stages $jj + 1$ and $jj + 2$ for processing. Meanwhile, the starting time $S_{i,jj+1,kk}$, $S_{i,jj+2,kk}$, ending time $C_{i,jj+1,kk}$, $C_{i,jj+2,kk}$ and departure time $Td_{i,jj+1,kk} = C_{i,jj+1,kk}$, $Td_{i,jj+2,kk}$, $C_{i,jj+2,kk}$ will all be recorded.

Step 7 Repeat Step 6 until n jobs are finished on stages jj, $jj + 1$, and $jj + 2$.

Step 8 jj is set to $jj + 2$ if $jj < (2m - 1)$, go to Step 3. Otherwise, exit.

4.3 SADE algorithm

In this section, the improved differential evolution (DE) algorithm (Storn and Price, 1997) is described to search the optimal solution. In the first phase, to get maximum population diversity, the initial population will be generated randomly. The diversity of solution is reduced and search ability of solution space is also reduced in the process of evolution. We apply rand/1/bin (Wu et al., 2007) evolution mode. Under this mode, a self-adaptive parameter adjusting strategy is proposed. It is designed that the cross operator CR and mutation operator F will automatically change with the evolution stop iterations. Equations (4) and (5) is the relationship between evolution parameter and stop iterations.

$$CR' = CR \cdot rand(0,1) \cdot 2^{\sin\left(\frac{StopGen}{stopGen_{\max}} \cdot \frac{\pi}{2}\right)}, \quad StopGen \in \{1, StopGen_{\max}\} \tag{4}$$

$$F' = F \cdot rand(0,1) \cdot 2^{\sin\left(\frac{StopGen}{stopGen_{\max}} \cdot \frac{\pi}{2}\right)}, \quad StopGen \in \{1, StopGen_{\max}\} \tag{5}$$

Cross operator CR and mutation operator F change along with the evolution stop iteration, and the fluctuation amplitude has an exponential growth along with the increase of stop iteration, which will enhance the DE' capacity in escaping from the local optimal when the stagnant situation is met.

4.4 Buffer size estimation algorithm

Since different task sets have different number of jobs and production rhythms, different intermediate buffer sizes are needed. Hence, it is necessary to estimate whether there are enough buffer size and how to allocate the buffer space for each stage in advance to ensure the production runs smoothly. If not, the production may fall into blocking state and cannot be finished in time when the buffer is insufficient and unreasonable. In this paper, our method is based on probability distribution theory of discrete events, and is used to calculate the buffer space required by a given task set. In addition, an example is used to illustrate the effectiveness of HHDM method.

4.4.1 Discrete random variables' probability distribution and mathematical expectation

Definition 1: Assume that the all possible values of discrete random variables X is $x_d(1, 2,\ldots)$, the probability function of X can be described as following.

$$P\{X = x_d\} = p_d, (d = 1,2,\ldots) \tag{6}$$

Also, the probability distribution could be denoted as below:

X $x_1\ x_2 \ldots x_d \ldots$

P_d $p_1\ p_2 \ldots p_d \ldots$

In this case, the mathematical expectation can be figured out according to equation (7):

$$E(X) = x_1 \times p_1 + x_2 \times p_2 + \ldots + x_d \times p_d + \ldots \tag{7}$$

4.4.2 Intermediate BSEA

In this section, discrete random variables' probability distribution and mathematical expectation theory are used to estimate the intermediate buffer size. The intuition of the algorithm is to accurately model the randomness of the HFSP job assignment for the buffer size selection, and the details are described as follows:

Step 1 Choose a task set to be processed and proceed a pre-scheduling. In the pre-scheduling process, SADE algorithm is used as a global optimisation algorithm. Firstly, randomly generate an initial population which includes 30 individuals (NP =3 0), as we know, each individual in the population can stand for a solution for the problem. Each individual in population refers to the online processing sequence in the first stage of the job. According to the above coding and decoding rules, each individual can be converted into a complete solution for scheduling. Secondly, each individual can be transformed into a complete scheduling solution using the HHDM method. Intermediate

buffer size set ($\beta_{1,e}$, ($\beta_{2,e}$, …, ($\beta_{m-1,e}$) will be recorded after each individual is converted into a complete solution.

From above set $\beta_{j,e}$, $j = \{1, 2, \ldots, m-1\}$ and $e = \{1, 2, \ldots, 300\}$ refers to intermediate buffer size at stage j of the e^{th} complete schedule solution.

Its calculation is shown as equation (8). If $\beta_{j,e}$, $\beta_{1,e}$, $\beta_{2,e}$, …, $\beta_{m-1,e}$ denotes this intermediate buffer can store two jobs. And then, SADE algorithm is adopted as a global optimisation algorithm to iterative update operation for each individual. The iteration times is 10 (*iteration* = 10), and the intermediate buffer usage ($\beta_{1,e}$, $\beta_{2,e}$, …, $\beta_{m-1,e}$) for new individuals is recorded. In such way, there are 30 * 10 intermediate buffer size sets which corresponded to feasible scheduling solution.

$$\beta_{j,e} = \text{maximum}\left(I_1, I_2, \ldots, I_n\right) \tag{8}$$

$I_i (i \in \{1, 2, \ldots, n\})$ is the job quantity, which can be stored in the buffer bu_j, when the job J_i enters the buffer bu_j until it departs from the buffer. When $Td_{i,j,k} < S_{i,j+1,k'}$, set $y = 1$.

$$\begin{aligned} Nc_i = card\Big(i' \Big| &\left[Td_{i',j,k}, S_{i',j+1,k}\right] \\ &\cup \left[Td_{i,j,k}, S_{i,j+1,k}\right] \neq \varnothing\Big) \\ &i' \in \{1, 2, \ldots, n\}, i' \neq i \end{aligned} \tag{9}$$

As is shown in equation (9), NC_i shows other jobs that exist at the same time in the buffer bu_j when job J_i enters the buffer bu_j until it leaves buffer. $y = y + Nc$, $I_i = y$.

$$x_j = \left(\beta_{j,1}, \beta_{j,2}, \ldots, \beta_{j,300}\right), \quad j \in \{1, 2, \ldots, m-1\} \tag{10}$$

In equation (10), x_j refers to the value of intermediate buffer size between stage j and stage $j + 1$ in 300 solutions.

Step 2 Calculate the distribution probability of each intermediate buffer size x_j. $b_{j,N}$ refers to the value of intermediate buffer size $x_j$and there are totally N values.

$$x_j \quad b_{j,1}\ b_{j,2} \ldots b_{j,}N$$

$$P_j \quad p_1\ p_2 \ldots p_N$$

Step 3 Calculate the expectation value of intermediate buffer size $x_j (j \in \{1, 2, \ldots, m-1\})$ according to equation (11):

$$E\left(x_j\right) = b_{j,1} \times p_1 + b_{j,2} \times p_2 + \ldots + b_{j,N} \times p_N \tag{11}$$

Based on the above step, the expectation value of intermediate buffer size after each stage can be calculated, and the expectation value can be set as the value of this intermediate buffer size Ebc_j, as it shows in equation (12):

$$Ebc_j = E\left(x_j\right), \quad j = \{1, 2, \ldots m-1\} \tag{12}$$

As a result, for the current given task set, the reasonable configure method of intermediate buffer size is shown in equation (13):

$$EBC = \left(Ebc_1, Ebc_2, \ldots, Ebc_{m-1}\right) \tag{13}$$

To illustrate the effectiveness above HHDM method, we take an example as shown in Figure 2. Based on this example, there are 12 jobs and three stages (Wang et al., 2013), the number of parallel machines in each stage is 3, 2 and 4, respectively. The processing time of the jobs in each stage is given in Table 2. The jobs queue A_1 in the first stage is {6, 1, 11, 5, 2, 8, 10, 12, 9, 3, 4, 7}. Then, the corresponding complete schedule of this job queue can be obtained using the HHDM method (Figure 2). It is obvious that the intermediate buffer stage size in stage 1 is 1, the area where blue rectangle stands for the occupied buffer stages, and the red rectangle refers to the blocking time of machines in each stage.

Figure 2 Schedule result GANTT by HHDM method (see online version for colours)

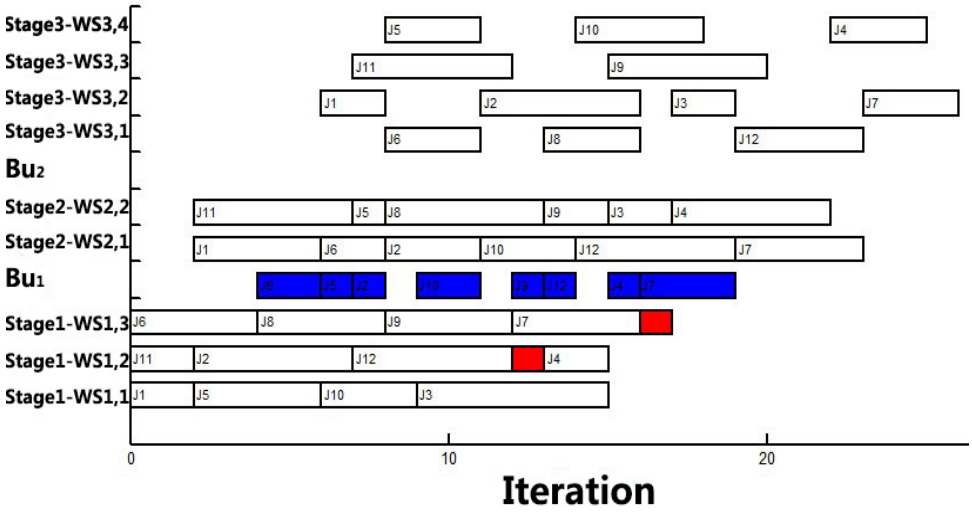


Figure 3 Mean S/N ratio plot for SADE (see online version for colours)

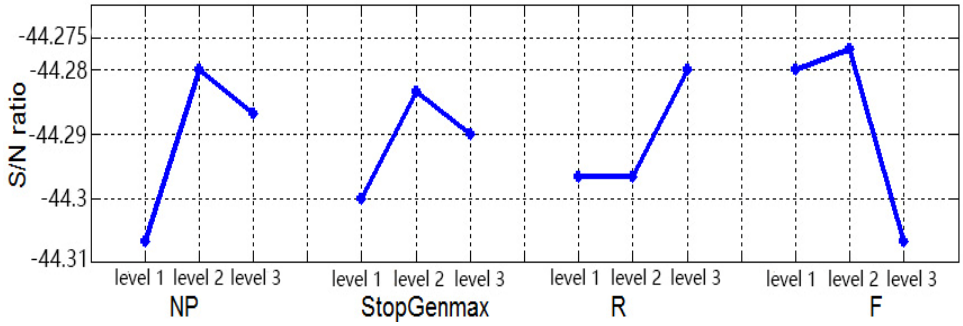


Table 2 Processing time for machine

OP	*WS*	*J*											
		J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9	J_{10}	J_{11}	J_{12}
OP_1	$WS_{1,1}$	2	4	6	4	4	6	5	3	2	3	5	6
	$WS_{1,2}$	2	5	5	3	5	5	2	5	5	6	2	5
	$WS_{1,3}$	3	4	4	4	3	4	4	4	4	4	4	4
OP_2	$WS_{2,1}$	4	3	4	6	3	2	4	7	1	3	3	5
	$WS_{2,2}$	5	4	2	5	1	3	6	5	2	4	5	4
OP_3	$WS_{3,1}$	2	3	3	3	3	4	3	3	7	4	6	3
	$WS_{3,2}$	3	4	4	6	4	3	4	3	8	8	7	4
	$WS_{3,3}$	2	5	2	5	6	9	3	6	6	6	6	7
	$WS_{3,4}$	3	4	5	8	5	5	5	4	7	7	5	5

5 Computational result

This section describes the testing data in Section 5.1, the simulation process of algorithm's parameter calibration in Section 5.2, and experimental results in Section 5.3 verifying the effect of our designed SADE algorithm as well as demonstrating whether the intermediate buffer size setting method is reasonable. All algorithms are implemented using MATLAB 2012b, running on the PC with Windows 7 system, Core i5 CPU 2.4 Hz and 4 G memory.

Table 3 Configurations of test instances

No.	*Instance*	*n*	*m*	*Parallel machines*	*Processing time between (unit: s)*
1	J10c5a1	10	5	(3, 3, 2, 3, 3)	[3, 20]
2	J10c5a2	10	5	(3, 3, 3, 3, 3)	[3, 20]
3	J15C5a1	15	5	(3, 3, 2, 3, 1)	[3, 20]
4	J15C5a2	15	5	(3, 3, 3, 3, 3)	[3, 20]
5	J20c4	20	4	(3, 2, 3, 3)	[5, 25]
6	J30c3	30	3	(4, 3, 4)	[5, 25]
7	J30c4	30	4	(4, 3, 4, 4)	[5, 25]
8	J50c2	50	2	(5, 5)	[5, 25]
9	J50c3	50	3	(4, 3, 4)	[5, 25]
10	J100c2	100	2	(5, 5)	[5, 25]

5.1 Testing data

Since different task sets need different intermediate buffer size, in order to validate the rationality and practicality of the methods to set intermediate buffer size and the effectiveness of the global search algorithm SADE, different scales of instances are used to test in experiment and generate the required data including quantity of jobs, processing time, quantity of stages and quantity of machines. Table 3 illustrates eight groups of instances. It is important to note that the deviation of processing time of jobs on parallel machines is less than five units.

5.2 Parameter calibration

There are four parameters in SADE algorithm: population size *NP*, the maximal stop generation $StopGen_{\max}$, the cross parameter *CR* and mutation *F*. Since the choice of parameters strikingly impacts on performance of algorithms, a preliminary calibration experiment (Montgomery, 2005) is carried out for investigating the effect of SADE in different levels of parameters and finding the most suitable levelling value of these parameters. The method in this experiment is Taguchi method which was developed by Dr. Gencini Taguchi in 1960s. In this technique, the values of quality characteristics obtained through the experiments are transformed into a measure called signal-to-noise ratio (S/N ratio) which is classified into three groups: the-smaller-the-better, the-larger-the-better, and nominal-is-best. Since the type of performance in this paper is minimum optimisation, the corresponding smaller-the-better S/N ratio is calculated by equation (14).

$$S/N\ ratio = -10\log_{10}(objective\ function)^2 \tag{14}$$

Four levels of the parameters are depicted in Table 4. The orthogonal array design for this is shown in Table 5. The degree of effect on the optimisation performance of SADE of each parameter is shown in Table 6. The results are shown in Figure 3. By selecting the maximum amount of S/N ratios in Figure 3, the optimum levels of parameters are delineated as follows:

- quantity of individuals generated in every generation $NP = 30$
- the maximal stop generation $StopGen_{\max} = 15$

- the range of cross parameter value $CR \in [0.3, 0.7]$
- the range of mutation parameter value $F \in [0.5, 0.9]$.

Table 4 Parameter levels for SADE

Parameters	*Factor*		
	1	*2*	*3*
NP	20	30	40
$StopGen_{max}$	10	15	20
CR	[0.1, 0.5]	[0.2, 0.6]	[0.3, 0.7]
F	[0.4, 0.8]	[0.5, 0.9]	[0.6, 1]

Table 5 The orthogonal array and AVG values

Combinations	*NP*	$StopGen_{max}$	*CR*	*F*	*AVG*	*S/N ratio*
1	1	1	1	1	164.33	–44.32
2	1	2	2	2	163.87	–44.29
3	1	3	3	3	164.21	–44.31
4	2	1	2	3	164.30	–44.31
5	2	2	3	1	163.27	–44.26
6	2	3	1	2	163.59	–44.27
7	3	1	3	2	163.54	–44.27
8	3	2	1	3	164.07	–44.30
9	3	3	2	1	163.83	–44.29

Table 6 Parameter's response table

Level	*NP*	$StopGen_{max}$	*CR*	*F*
1	–44.3067	–44.30	–44.2967	–44.28
2	–44.28	–44.2833	–44.2967	–44.2767
3	–44.2867	–44.29	–44.28	–44.3067
Range	0.0267	0.0167	0.0267	0.03
Rank	3	4	2	1

5.3 Results analysis

In this section, ten groups of instances and five schemes are designed to verify the effectiveness of the proposed SADE algorithm can achieve a better performance than both the DE algorithm and the GA algorithm. We also verify the performance of BSEA which calculates the size of the intermediate buffer. The details of each scheme are shown in Table 7.

Table 7 Information of five set of simulation schemes

Scheme	*Global optimisation algorithm*	*Buffer configuration*	*Intermediate buffer calculation method*
1	GA	Infinite buffer space	
2	DE	Infinite buffer space	
3	DE	Finite buffer space	BSEA
4	SADE	Infinite buffer space	
5	SADE	Finite buffer space	BSEA

Usually, the purpose of using SADE algorithm is to improve the quality of solutions and accelerate the convergence speed. Each instance for each algorithm will respectively perform 20 times to obtain the results, and then, the mean value of 20 experiments is taken as the experimental result of the scheme. The maximum iteration is set 200. In order to clearly reflect difference of scheduling results in different experiment schemes, relative error function is set as *PRE* (the percentage relative error), which is shown as equation (15).

$$PRE(A, B) = \frac{|C_A - C_B|}{C_B} \times 100\% \tag{15}$$

Based on above equation, C_A is the makespan value obtained by algorithm A, and C_B is the makespan obtained by algorithm B.

Based on the results from Table 8 to Table 10, the following conclusions can be drawn on our proposed SADE-BSEA algorithm.

1 Adopting HHDM not only obtains the complete solution of the hybrid flow shop with finite intermediate buffer scheduling problem but also clearly reflects the processing situation of each processing job in each stage including buffer stage as well as real-time status of each stage in various processing machines, such as blocking status, etc.

2 Evolving parameter strategy is introduced, which can be self-adaptively adjusted DE parameter with stop iteration. This can enhance the ability of DE algorithm to escape local extremum so that DE algorithm always remains the ability to continue seeking better solution. At the same time, GA algorithm is also introduced as the contrast algorithm. From the analysis in Table 8, SADE algorithm can obtain the best optimisation result among the three algorithms.

   PRE(*GA*, *SADE*) is 13.32% and *PRE*(*DE*, *SADE*) is 9.88% under the small scale data. *PRE*(*GA*, *SADE*) is 21.70% and *PRE*(*DE*, *SADE*) is 15.74% under the large scale data. This indicates that the optimising effect will be more obvious along with the expansion of data scale.

3 The intermediate buffer size calculation method is effective. As is shown in Table 9 and Table 10, the intermediate buffer size value is set as the calculated value obtained by BSEA method, and the optimisation results are either not affected or the impact is very small in comparison with this infinite buffer.

4 The intermediate buffer size is calculated by BSEA method and this size will obtain different results on the basis of different task set. This is convenient for staff to predict intermediate buffer size before this processing task is arranged so as to reasonably distribute processing resources.

Table 8 Simulating results of scheme 2 and scheme 4

Instances	C_{GA}	*CPU(s)*	C_{DE}	*CPU(s)*	C_{SADE}	*CPU(s)*	*PRE(GA, SADE)*	*PRE(DE, SADE)*
J10c5a1	79.54	45.32	74.93	26.51	71.2	34.73	11.71%	5.24%
J10c5a2	72.78	42.67	69.47	23.36	63.9	32.90	13.20%	8.72%
J15c5a1	96.82	67.91	93.40	54.49	85	60.57	13.32%	9.88%
J15c5a2	185.63	72.08	179.15	58.02	166	66.53	11.83%	7.92%
J20c4	182.38	124.75	179.74	98.38	162	116.42	12.58%	10.95%
J30c3	179.47	130.21	175.37	108.35	153	125.51	17.30%	14.62%
J30c4	186.96	126.73	181.28	100.55	159.75	119.47	17.03%	13.48%
J50c2	289.67	238.92	283.76	191.83	251	224.56	15.41%	13.05%
J50c3	287.56	219.38	275.60	173.35	240	208.17	19.82%	14.83%
J100c2	369.24	308.28	350.85	260.19	303.4	297.82	21.70%	15.74%

Table 9 Simulating results of scheme 4 and scheme 5

	Instances							
λ	*J10c5a1*		*J10c5a2*		*J15c5a1*		*J15c5a2*	
	0	*1*	*0*	*1*	*0*	*1*	*0*	*1*
C_{max}	71.2	72.4	63.9	64.3	85	82.6	166	166.5
CPU(s)	34.73	40.52	32.90	38.5	60.57	63.74	66.53	67.16

Note: λ is a binary variable, $\lambda = 1$ represent the buffer size is finite, otherwise, the buffer size is infinite.

Table 10 Experimental results of scheme 4 and scheme 5

	Instances							
λ	*J20c4*		*J30c4*		*J50c3*		*J100c2*	
	0	*1*	*0*	*1*	*0*	*1*	*0*	*1*
C_{max}	162	163.11	159.75	166.25	240	243.72	303.4	308.4
CPU(s)	116.42	151.13	119.47	172.77	208.17	239.78	297.82	360.33

Figure 4 Relationship between fitness value and training iterations of scheme 2, 4 in problem 5

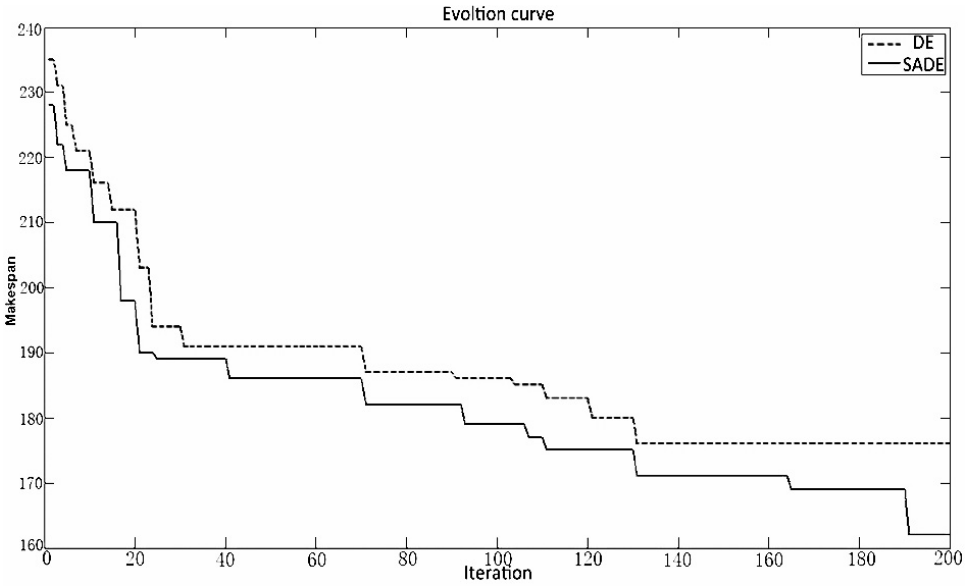
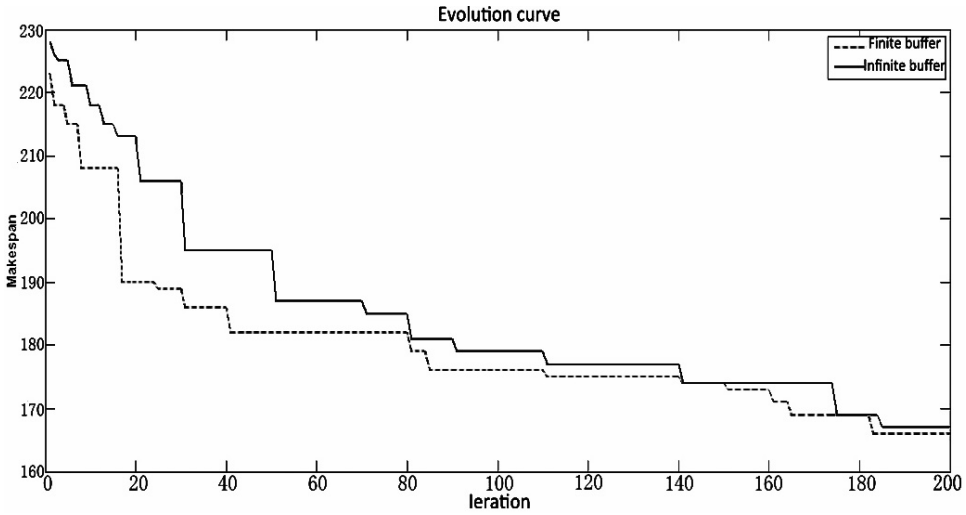


Figure 5 Relationship between fitness value and training iterations of scheme 4 and 5 in problem 5



From Figure 4, it is obvious that the SADE algorithm has a better ability for searching the optimal solution than DE algorithm. In the initial stages of the evolution, there is a significant decline of the fitness value conducted by the two algorithms. With the increase of the evolution generation, the diversity of the population decreases. The evolution ability of DE and SADE algorithm both become weak. Especially, after 130 generations, the DE evolution curve is in stagnation while the SADE keeps a persistent tendency to evolve, which proves that the self-adaptive parameter adjusting strategy can effectively help DE algorithm jump out of the local extreme value.

As shown in Figure 5, SADE algorithm is used to globally optimise the hybrid flow shop with finite buffer constraints and the hybrid flow shop scheduling without buffer constraints. At initial evolution, because of the finite buffers constraints, the fitness function values on the hybrid flow shop with finite buffer constraints are smaller than those on the hybrid flow shop without finite buffer constraints. With the growing of evolutional generation, the influence of the finite buffer constraint on the optimisation effect becomes weakening. At the last stage of evolution, it is obvious that the two evolving curves all obtained the approximate optimal solutions, which proves that it is reasonable and effective to use BSEA method to set intermediate buffer size. Meanwhile, this method will not affect production efficiency in workshop.

6 Conclusions

In this paper, the hybrid flow shop with finite intermediate buffer scheduling problem is studied. A HHDM is proposed to process the encoding and decoding procedures. The intermediate buffer size is estimated by applying the probability distribution theory of discrete events. BSEA method is proposed to set the intermediate buffer size, and the global optimal solution is searched by a novel SADE algorithm which integrates a self-adaptive parameter adjusting strategy along with iterations. The proposed algorithm overcomes the premature convergence drawback while ensuring the algorithm evolution is on the reasonable direction. The simulation results validate that the BSEA method obtains a reasonable intermediate buffer size without affecting the production efficiency using different task sets. It is also shown that the SADE algorithm performs much better than classical DE algorithm in terms of solving the HFSP with the finite intermediate buffer constraint.

Acknowledgements

This work was supported by the Liaoning Provincial Science Foundation (No. 201602608) and the Natural Science Foundation of China (No. 61503259).

References

Abyaneh, S.H. and Zandieh, M. (2012) 'Bi-objective hybrid flow shop scheduling with sequence-dependent setup times and limited buffers', *The International Journal of Advanced Manufacturing Technology*, Vol. 58, Nos. 1–4, pp.309–325.

Agnetis, A., Rossi, F. and Gristina, G. (1998) 'An exact algorithm for the batch sequencing problem in a two-machine flow shop with limited buffer', *Naval Research Logistics*, Vol. 45, No. 2, pp.141–164.

Akrami, B., Karimi, B. and Hosseini, S.M. (2006) 'Two metaheuristic methods for the common cycle economic lot sizing and scheduling in flexible flow shops with limited intermediate buffers: the finite horizon case', *Applied Mathematics and Computation*, Vol. 183, No. 1, pp.634–645.

Arfi, F., Ilié, J.M. and Saidouni, D.E. (2016) 'Solving preemptive job-shop scheduling problems using a true concurrency model', *International Journal of Simulation and Process Modelling*, Vol. 11, Nos. 3–4, pp.292–304.

Eskandari, H. and Hosseinzadeh, A. (2014) 'A variable neighbourhood search for hybrid flow-shop scheduling problem with rework and set-up times', *Journal of the Operational Research Society*, Vol. 65, No. 8, pp.1221–1231.

Gupta, J.N. (1988) 'Two-stage, hybrid flowshop scheduling problem', *Journal of the Operational Research Society*, Vol. 39, No. 4, pp.359–364.

Han, Z., Ma, X., Yao, L. and Shi, H. (2012) 'Cost optimization problem of hybrid flow-shop based on PSO algorithm', *Advanced Materials Research*, Vols. 532–533, pp.1616–1620.

Han, Z., Zhu Y., Ma, X. and Chen, Z. (2016) 'Multiple rules with game theoretic analysis for flexible flow shop scheduling problem with component altering times', *International Journal of Modelling, Identification and Control*, Vol. 26, pp.1–18.

Irohara, T. (2010) 'Lagrangian relaxation algorithms for hybrid flow-shop scheduling problems with limited buffers', *Biomedical Fuzzy and Human Sciences: The Official Journal of The Biomedical Fuzzy Systems Association*, Vol. 15, No. 1, pp.21–28.

Komaki, G.M., Teymourian, E. and Kayvanfar, V. (2015) 'Minimising makespan in the two-stage assembly hybrid flow shop scheduling problem using artificial immune systems', *International Journal of Production Research*, Vol. 54, No. 4, pp.963–983.

Li, J.Q. and Pan, Q.K. (2015) 'Solving the large-scale hybrid flow shop scheduling problem with limited buffers by a hybrid artificial bee colony algorithm', *Information Sciences*, Vol. 316, pp.487–502.

Montgomery, D.C. (2005) *Design and Analysis of Experiments*, John Wiley and Sons, New York, NY.

Naderi, B., Gohari, S. and Yazdani, M. (2014) 'Hybrid flexible flowshop problems: Models and solution methods', *Applied Mathematical Modelling*, Vol. 38, No. 24, pp.5767–5780.

Qian, B., Wang, L., Huang, D.X., Wang, W.L. and Wang, X. (2006) 'An effective hybrid DE-based algorithm for multi-objective flow shop scheduling with limited buffers', *Computers & Operations Research*, Vol. 33, No. 10, pp.2960–2971.

Rane, A.B., Sunnapwar, V.K. and Khot, S.M. (2017) 'Cost models for improved vehicle assembly line performance', *International Journal of Simulation and Process Modelling*, Vol. 12, No. 2, pp.111–123.

Ruiz, R. and Vázquez-Rodríguez, J.A. (2010) 'The hybrid flow shop scheduling problem', *European Journal of Operational Research*, Vol. 205, No. 1, pp.1–18.

Smutnicki, C. (1998) 'A two-machine permutation flow shop scheduling problem with buffers', *OR Spectrum*, Vol. 20, No. 4, pp.229–235.

Soltani, S.A. and Karimi, B. (2015) 'Cyclic hybrid flow shop scheduling problem with limited buffers and machine eligibility constraints', *The International Journal of Advanced Manufacturing Technology*, Vol. 76, Nos. 9–12, pp.1739–1755.

Storn. R. and Price, K. (1997) 'Differential evolution – a simple and efficient heuristic strategy for global optimization over continuous spaces', *Journal of Global Optimization*, Vol. 11, No. 4, pp.341–359.

Tran, T.H. and Ng, K.M. (2013) 'A hybrid water flow algorithm for multi-objective flexible flow shop scheduling problems', *Engineering Optimization*, Vol. 45, No. 4, pp.483–502.

Wang, B. (2012) 'Scheduling hybrid flow shops by an improved memetic algorithm', *Proceedings of the 2012 International Conference on Computer Application and System Modeling*.

Wang, S., Wang, L. and Xu, Y. (2013) 'An estimation of distribution algorithm for solving hybrid flow-shop scheduling problem with stochastic processsing time', *32nd Chinese Control Conference*, Vol. 38, No. 3, pp.437–443.

Wang, X. and Tang, L. (2009) 'A tabu search heuristic for the hybrid flowshop scheduling with finite intermediate buffers', *Computers & Operations Research*, Vol. 36, No. 3, pp.907–918.

Wu, L., Wang, Y., Zhou, S. and Yuan, X. (2007) 'Research and application of pseudo parallel differential evolution algorithm with dual subpopulations', *Control Theory and Applications*, Vol. 24, No. 3, pp.453–458.

Yaurima, V., Burtseva, L. and Tchernykh, A. (2009) 'Hybrid flowshop with unrelated machines, sequence-dependent setup time, availability constraints and limited buffers', *Computers & Industrial Engineering*, Vol. 56, No. 4, pp.1452–1463.