Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 986218, 10 pages
http://dx.doi.org/10.1155/2013/986218

Research Article

Hybrid Discrete Differential Evolution Algorithm for Lot Splitting with Capacity Constraints in Flexible Job Scheduling

Xinli Xu,[1] Li Li,[1] Lixia Fan,[2] Jing Zhang,[2] Xuhua Yang,[1] and Wanliang Wang[1]

[1] *College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China*
[2] *College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China*

Correspondence should be addressed to Xinli Xu; 10165956@qq.com

Received 6 December 2012; Revised 31 January 2013; Accepted 20 February 2013

Academic Editor: Shengyong Chen

Copyright © 2013 Xinli Xu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

A two-level batch chromosome coding scheme is proposed to solve the lot splitting problem with equipment capacity constraints in flexible job shop scheduling, which includes a lot splitting chromosome and a lot scheduling chromosome. To balance global search and local exploration of the differential evolution algorithm, a hybrid discrete differential evolution algorithm (HDDE) is presented, in which the local strategy with dynamic random searching based on the critical path and a random mutation operator is developed. The performance of HDDE was experimented with 14 benchmark problems and the practical dye vat scheduling problem. The simulation results showed that the proposed algorithm has the strong global search capability and can effectively solve the practical lot splitting problems with equipment capacity constraints.

1. Introduction

In a practical production system, with much change of market and diversification of customer needs, variety and small batch production mode has gradually become the main way of manufacturing. In this mode, batch splitting scheduling problem has become an urgent issue addressed in the actual production. Batch splitting, also called lot streaming in many researches, is the process of splitting given jobs, each consisting of a batch of identical parts, into many smaller subbatches to allow their overlapping processing on alternative machines to get a better performance. An overview of the models and methods for batch splitting has been reported in the literature [1–3]. Job shop scheduling problem with batch splitting is commonly more complex than traditional job shop scheduling problem and usually more close to the actual manufacturing system.

In recent years, a growing body of literature suggests the research of job shop scheduling problems with batch splitting" for the sake of correctness. Low et al. [4] and Buscher and Shen [5] adopted integer programming methods to solve the batch splitting in job shop scheduling problem. Buscher and Shen (2009) presented consistent-sized batch splitting strategy under the conditions of given subbatches and put forward three-phase algorithm to solve lot streaming problem in job shops, which consists of predetermination of sublot sizes, the determination of schedules based on tabu search, and the variation of sublot sizes [6].

To minimize makespan of a batch splitting scheduling problem with setup time and alternative machines, Pan and Zhu (2004) split an original batch into equal sized subbatches, with the batch size for each subbatch fixed in advance, and used genetic algorithm to find the optimal sequence in which operations have to be machined [7]. Sun et al. [8] and An [9] put forward a batch splitting algorithm and a sublot scheduling algorithm based on genetic algorithm, optimized the number of subbatches of each job, and determined the size of each subbatch based on splitting strategy of equal-sized subbatches. Considering the production batch and the setup time of machine, Lin et al. (2007) proposed a hybrid genetic algorithm and combined the heuristic rules with simulated annealing algorithm to solve the flexible job shop scheduling problem by fixing the number of subbatches [10]. Ju and Zhu (2007) regarded each job as a batch, merged the same type of jobs sorted which are adjacent to each other into a subbatch in the evolution, combined particle swarm optimization and

genetic algorithm, and proposed the strategy of job shop scheduling optimization of batch production [11].

Bai et al. [12] proposed a flexible size lot splitting approach based on "cursors," designed a novel particle coding scheme combining the lot splitting with the sublot scheduling in particle swarm algorithm to solve the multiobjective flexible job shop scheduling with lot splitting. Huang [13] discussed multiobjective job shop scheduling with lot splitting production when the number of lots split is 1, 2, 3, and 4, in which subbatches are equal, and used ant colony algorithm to optimize the processing sequences. However, these recent researches focused on that the batch sizes of subbatch and sub-subbatches are determined in advance, and these research results have provided an important basis for intensive study solving the lot splitting problem in job shop scheduling.

Differential evolution algorithm (DE) [14] introduced by Storn and Price (1995) is a novel and efficient parallel direct search method, which has good robustness as well as fast convergence. Consequently, over the past few years, several researchers have demonstrated the applicability of DE in job shop scheduling problem [15–20]. In particular, Sang et al. (2010) gave an equal-sized batch scheme in advance and used DE to optimize the sequence of nonintermingling subbatches for the flow shop problems with the objective to minimize the total weighted earliness and tardiness [17]. Zhao et al. (2010) used the strategy of small production batch to shorten the production cycle. Considering mutation operator with the characteristics of the same sum of the individual values to meet the batch splitting constraints, DE was firstly applied in the lot splitting scheduling problem, and job shop scheduling problem with consistent-sized batch splitting [18], variable-sized batch splitting [19], and lot streaming under multiple-resource constraints [20] were, respectively, studied. However, the above researches did not involve the constraints of equipment capacity.

For solving the lot splitting with equipment capacity constraints in flexible job shop scheduling, it needs to split an original lot into many appropriate smaller processing subbatches, find the optimal sequence of those subbatches, and also consider equipment processing capacity constraints. It is a more complex NP hard problem which is motivated by practical engineering applications. In this paper, we proposed a hybrid discrete differential evolution algorithm for lot splitting with equipment capacity constraints in flexible job shop scheduling problem and used a two-level coding scheme to map the solution space into the chromosome space. To improve the performance of DE and avoid its trapping into local optimal solution, a local search strategy with dynamic random searching based on the critical path and random mutation operator is developed. The simulation results indicated that the algorithm has strong global search capability and can effectively solve the practical batch scheduling problem with the equipment capacity constraints.

The remainder of this paper is organized as follows. Section 2 provides the formulation of lot splitting with equipment capacity constraints in job shop scheduling problem, including subbatch size constraints, product and requirement constraints, setup costs, and setup time constraints and delivery duration constraints. Section 3 proposes a hybrid parallel algorithm combining with a local search strategy with dynamic random searching based on the critical path and random mutation operator to solve both the batch splitting problem and the batch scheduling problem. Section 4 gives the computational experiments and analyzes the results, and Section 5 makes the conclusion.

2. Problem Description and Formulation

2.1. Problem Description. Given a job shop production system with N kinds of products and R kinds of machines, where the capacity of each kind of machine is different, that is, each kind of machine has minimum and maximum output, the assumptions are that (1) any product can be divided into S batches processed individually, (2) each product has several operations that are waiting to be processed, (3) each machine can process different operation of any type of product, and (4) the setup time and costs for each operation cannot be omitted. Lot splitting with equipment capacity constraints in job shop scheduling problem is how to merge and split orders received, determine the number of sublots and sublot sizes of each product, the operational sequencing and working equipment of each batch, and minimize the objective function to satisfy with the constraints.

2.2. Notations

j: Machine index

o: Order index

p: Product type index

L: Number of orders

M: Number of machines

N: Number of production types

n_p: Number of operations of product p

J: A set of machines

O: A set of orders

P: A set of products

D_p: Demand of product p

$BD_{j,p}$: Sublot size of product p processed on machine j

BN_p: Number of sublots of product p

$\min V_j$: Minimum lot size allowed on machine j, that is, minimum output of machine j

$\max V_j$: Maximum lot size allowed on machine j, that is, maximum output of machine j

$\text{maxCost}_{j,p}$: Processing cost of product p on machine j with the maximum output

$\text{conCost}_{j,p}$: Constant cost of product p processed on machine j

$\text{unitCost}_{j,p}$: Unit cost of product p processed on machine j

$\max\text{Time}_{j,p}$: Processing time of product p processed on machine j with the maximum output

$\text{conTime}_{j,p}$: Constant time of product p processed on machine j

$\text{unitTime}_{j,p}$: Unit Time of product p processed on machine j

$\text{trCost}_{j,p,p'}$: Setup costs of product p and p' successively processed on machine j

$\text{trTime}_{j,p,p'}$: Setup time of product p and p' successively processed on machine j

$\beta_{p,o}$: Tardiness penalty coefficient of product p in order o

$d_{p,o}$: Due date of product p in order o

$TS_{j,p}$: Start time of product p processed on machine j

$TE_{j,p}$: Finishing time of product p processed on machine j

$Pe_{p,o}$: Completion time of product p in order o

$\text{poCost}_{p,o}$: Tardiness penalty cost of product p in order o.

2.3. Formulation. Lot splitting with equipment capacity constraints in job shop scheduling problem has the following constraints.

(1) Subbatch Size Constraint. In general, several products are taken in a batch as a unit to be processed during batch production. For the problems with equipment capacity constraints, each machine has its minimum output ($\min V_j$) and maximum output ($\max V_j$). The subbatch size ($BD_{j,p}$) of product p processed on machine j must satisfy the formula (1) and achieve the maximum output as much as possible

$$\min V_j \le BD_{j,p} \le \max V_j, \quad \forall j \in J,\ p \in P. \tag{1}$$

(2) Product Requirement Constraint. The number of the pth type of product processed on all machines should be equal to the total number of its requirement (D_p)

$$\sum_{j \in J} BD_{j,p} = D_p, \quad \forall p \in P. \tag{2}$$

(3) Setup Costs and Setup Time Constraint. In batch production, each machine is a flexible production line as it can process any type of product. Setup between two different products on the same machine usually generates setup costs. For example, the dye vat commonly must be cleaned in order to guarantee the quality of the next product, so it results in setup costs, that is, cleaning costs

$$TS_{j,p'} - TS_{j,p} \ge \text{trTime}_{j,p,p'}, \quad \forall j \in J,\ p \in P,\ p' \in P. \tag{3}$$

Formula (3) represents that the start time difference ($TS_{j,p'} - TS_{j,p}$) between the earlier product p and the later product p' processed on machine j should be bigger than the setup time ($\text{trTime}_{j,p,p'}$). Setup costs are affected by two successive products processed and equipment capacity; that is, there are the setup costs between the earlier task and the later task on the same equipment.

(4) Delivery Duration Constraint. Delivery dates of the orders from different customers always are different. In the same order, the customer sometimes even may have different requirements that some productions need to be processed urgently, so the tardiness penalty coefficient ($\beta_{p,o}$) of production p is different. Generally, $\beta_{p,o}$ of important customers is bigger than that of general customers. If the completion time ($Pe_{p,o}$) is beyond the delivery time ($d_{p,o}$), there will be the tardiness penalty costs ($\text{poCost}_{p,o}$) just like the following:

$$\text{poCost}_{p,o} = \beta_{p,o} \times \max\left(0, Pe_{p,o} - d_{p,o}\right), \quad \forall p \in P,\ o \in O. \tag{4}$$

Based on above constraints, the batches splitting of all products must be determined according to the received orders, moreover the number of sublots and sublot sizes of each procedure for each product must be arranged on different machine to minimize the total production costs of enterprises under the precondition of meeting the orders requirements as far as possible. Objective function of the lot splitting with equipment capacity constraints in flexible job shop scheduling problem can be seen as follows:

$$\begin{aligned} \text{minimize}\quad f = & \sum_{j \in J,\ p \in P} \left(\text{conCost}_{j,p} + \text{unitCost}_{j,p} \times BD_{j,p}\right) \\ & + \sum_{j \in J,\ p \in P,\ p' \in P} \text{trCost}_{j,p,p'} \\ & + \gamma \sum_{p \in P,\ o \in O} \beta_{p,o} \times \max\left(0, Pe_{p,o} - d_{p,o}\right). \end{aligned} \tag{5}$$

In the above formula, the total production costs include processing costs ($\sum_{j \in J, p \in P}(\text{conCost}_{j,p} + \text{unitCost}_{j,p} \times BD_{j,p})$), setup costs ($\sum_{j \in J, p \in P, p' \in P} \text{trCost}_{j,p,p'}$), and tardiness penalty costs ($\sum_{p \in P, o \in O} \beta_{p,o} \times \max(0, Pe_{p,o} - d_{p,o})$). γ is the ratio coefficient between tardiness penalty costs and the other two costs. Processing costs are made up of constant costs ($\sum_{j \in J, p \in P} \text{conCost}_{j,p}$) and variable costs ($\sum_{j \in J, p \in P} \text{unitCost}_{j,p} \times BD_{j,p}$). In the practical production, processing costs and processing time of product p processed on machine j with maximum output can be obtained by experience.

3. A Hybrid Discrete Differential Evolution Algorithm

3.1. Encoding and Decoding. In the lot splitting scheduling problem, the number of sublots and sublot sizes of each product must be determined, and the working machine and operational sequencing of each sublot must be also arranged. In this paper, a two-level coding scheme is developed to map the solution space into the chromosome space, where

the first level code represents lot splitting chromosome (chrome_1), and the second level code denotes lot scheduling chromosome (chrome_2). chrome_1 is used to determine the number of subbatches and subbatch sizes of each product, and chrome_2 is used to prioritize the operational sequencing arrangement of each subbatch.

For the problem with equipment constraints, a new coding scheme is proposed to describe chrome_1 considering the requirements of maximum output as far as possible. chrome_1 is divided into two parts, where the first part is the lot splitting scheme number of product p, the second part describes sublot sizes of product p, and the meaningless "0" is as the interval between them. The number of lot splitting scheme for product p (K_p) is less than or equal to the smaller one between its demand (D_p) and the number of machines (M). For example, there are two products divided into two subbatches. The batch size of each product is 8, and it can be divided into four schemes such as (1,7), (2,6), (3,5), and (4,4). So one of the lot splitting chromosomes may be [3, 1|0|3, 5, 1, 7]. As shown in Figure 1, BP_{P,BN_p} represents sublot size of the BN_pth batch of product p.

Based on the coding scheme of processing procedure, chrome_2 represents the operational sequencing of each procedure in each batch and the length of each chromosome $\text{len}_1 = \sum_{p=1}^{N} BN_p \times n_p$. For example, chrome_2 is [21, 11, 22, 12, 12, 21, 22, 11, 11, 22, 21, 12], where "21" means the first batch of the second product and three "21" indicate three different successive procedures of the second product.

Based on chrome_1 and chrome_2, the decoding scheme is as follows.

Step 1. Set $l = 1$ and the allowed start time of each machine $MST_j = 0$, where $j = 1, \ldots, M$.

Step 2. Select the lth gene of chrome_2 from left to right, acquire the processing product number p and the subbatch number k, and then obtain the subbatch size ($BP_{p,k}$) of the kth batch of production p combining with chrome_1 based on the gene location.

Step 3. Allocate working machines for the unscheduled tasks according to Substeps 3.1 and 3.2 and execute preparation and processing operation.

Substep 3.1. Fully considering the utilization of equipments, select the appropriate type of machine according to the subbatch size ($BP_{p,k}$). If $BP_{p,k}$ equals the maximum output of machine, then arrange the kth batch of production p to the type of equipment. Otherwise, arrange the task to process on the type of equipment whose maximum output is the nearest to $BP_{p,k}$.

Substep 3.2. After confirming the type of equipment, select a specific equipment according to the dispatching rules that first completed equipment is firstly arranged.

Firstly calculate the processing operation number s corresponding to the lth gene of chrome_2 and compare the finish time of the preceding operation with the earliest allowed start time of the processing equipment to obtain the start time of the kth batch of production p. According to the operation of the production p and the preceding task processed on the machine, determine whether they need setup. If this type of machine is the only one, then directly arrange the kth batch of production p to process on this machine. Otherwise, select the machine which can earliest complete the other tasks. Finally, calculate start time and finish time of the kth batch of production p.

Step 4. Refresh the earliest allowed start time of each machine.

Step 5. Set $l = l + 1$. If $l \le \text{len}_1$, then go to Step 2. Otherwise, quit.

3.2. Global Evolution Procedure. DE is a parallel evolutionary algorithm based on population, which utilizes N_p parameter vectors as a population $X^t = [X_1^t, X_2^t, \ldots, X_{N_p}^t]$ for each generation t. Each vector called each individual of the population represents potential solution for the optimization problem. DE generates a mutant vector V_i^{t+1} for each vector X_i^t in the population by adding the weighted difference between two randomly selected vectors ($X_{r_2}^t$ and $X_{r_3}^t$) to a third one $X_{r_1}^t$. A trial vector U_i^{t+1} is then generated by using the crossover operator which mixes the components of the mutant vector V_i^{t+1} and the original one X_i^t. In the last step of each iteration, the selection operator with greedy strategy chooses the better one for the next generation by comparing U_i^{t+1} with X_i^t.

Since the DE algorithm was originally designed to work with continuous variables, rather most engineering problems have either an integer objective function or discrete objective function. Here, we adopt the following method as the global search strategy of a hybrid discrete differential evolution algorithm (HDDE) in this paper, which makes the individuals evolve directly in the discrete domain and can effectively speed up the algorithm.

(1) Consider

$$V_i^{t+1} = F \otimes G\left(F \otimes G\left(X_{r_2}^t, X_{r_3}^t\right), X_{r_1}^t\right), \tag{6}$$

where $i, r_1, r_2, r_3 \in [1, N_p]$, randomly selected; expect $r_1 \neq r_2 \neq r_3 \neq i$. $X_{r_1}^t$, $X_{r_2}^t$, and $X_{r_3}^t$ are three different randomly selected individuals, V_i^{t+1} is a mutant individual, and $F \in [0, 1]$. Formula (6) consists of two parts as follows.

(a) Consider

$$\begin{aligned} Y_i^{t+1} &= F \otimes G\left(X_{r_2}^t, X_{r_3}^t\right) \\ &= \begin{cases} G\left(X_{r_2}^t, X_{r_3}^t\right), & \text{if rand} < F \\ X_{r_2}^t, & \text{else.} \end{cases} \end{aligned} \tag{7}$$

In formula (7), rand represents a uniformly distributed random value that ranges from zero to one. Here generate a random number (rand), if rand $< F$, then set the subsequent individual $Y_i^{t+1} = G(X_{r_2}^t, X_{r_3}^t)$. Otherwise, $Y_i^{t+1} = X_{r_2}^t$. For the lot scheduling chromosome, $Y_i^{t+1} = G(X_{r_2}^t, X_{r_3}^t)$ means

$$\text{chrome}_1 \quad K_1, \ldots, K_p, \ldots, K_N | 0 | BP_{1,1}, \ldots, BP_{1,BN_1}, \ldots, BP_{p,1}, \ldots, BP_{p,BN_p}, \ldots, BP_{N,1}, \ldots, BP_{N,BN_N}$$

Figure 1: Lot splitting chromosome example of two products ($N = 2$).

to use POX crossover operator [21] that only crosses the processing sequence of the operations of parent chromosomes. For the lot splitting chromosome, $Y_i^{t+1} = G(X_{r_2}^t, X_{r_3}^t)$ indicates a simple two-point crossover that only operates the first part of the lot splitting chromosome.

(b) Consider

$$V_i^{t+1} = F \otimes G\left(Y_i^{t+1}, X_{r_1}^t\right) = \begin{cases} G\left(Y_i^{t+1}, X_{r_1}^t\right), & \text{if rand} < F \\ Y_i^t, & \text{else.} \end{cases} \tag{8}$$

Formula (8) describes that a mutant individual V_i^{t+1} is generated by the randomly selected individual $X_{r_1}^t$ and the above individual Y_i^{t+1}. Operation $G(Y_i^{t+1}, X_{r_1}^t)$ is the same as $G(X_{r_2}^t, X_{r_3}^t)$.

(2) Consider

$$U_i^{t+1} = Cr \otimes G\left(V_i^{t+1}, X_i^t\right) = \begin{cases} G\left(V_i^{t+1}, X_i^t\right), & \text{if rand} < Cr \\ V_i^{t+1}, & \text{else.} \end{cases} \tag{9}$$

Formula (9) means that a trial individual U_i^{t+1} is generated by the crossover of the original individual X_i^t and the mutant individual V_i^{t+1}, where Cr is the crossover probability ranged in $[0, 1]$. Operation $G(V_i^{t+1}, X_i^t)$ is the same as above.

3.3. Local Search Strategy. The above operators of global evolution procedure are directly carried out in discrete domain, which can effectively improve the search efficiency of the algorithm, but it is easy to fall into local extremum. In this paper, the local search strategy is developed to enhance the local exploration ability of the algorithm, which includes dynamic random search strategy based on the critical path used to adjust the lot scheduling chromosome and search neighborhood solutions of the operational sequencing in the case of the same subbatch divided and random mutation operator used to adjust the lot splitting chromosome and change the subbatch sizes again under the condition of the unchanged operational sequence of the sublots.

Dynamic random search [26] (DRS) is appropriate for continuous optimization problems and cannot be directly applied to the problem in this paper. So we improve DRS strategy to make it applicable for discrete operators. Furthermore, DRS involves the selection of the neighborhood, which has a larger influence on search quality and efficiency of the algorithm. Here the longest path of no time intervals between the operations is called the critical path of a feasible scheduling. The change of any operation on the critical path is the key to the change of the maximum finish time, and it is also an important part of the neighborhood structure constructed. Neighborhood based on the critical path can effectively avoid unnecessary search and improve the performance of algorithm. Therefore, the local search with two neighborhood structures including INTERCHANGE and INSERT based on the critical path is embedded in the framework of HDDE, which can adjust the operational sequencing under the condition of the lot splitting chromosome unchanged. The procedure of DRS is as follows.

Step 1. Set the initial values: *epoch* = 0 and Xcurrent = X, where Xcurrent and X are the current individual and the candidate individual, respectively.

Step 2. Reset the iteration counter, $n = 0$.

Step 3. Generate a new individual Xnew = INTERCHANGE (Xcurrent); that is, Xnew is generated by using exchange between the kth component and the qth component of Xcurrent and keeping the else components unchanged.

Step 4. Calculate the objective value of Xnew. Judge the dominance relation among Xnew, X, and Xcurrent. If Xnew dominates X, then update X = Xnew and go to Step 7. Otherwise, if Xnew dominates Xcurrent, then update Xcurrent = Xnew and go to Step 7.

Step 5. Generate a new individual, Xnew = INSERT(Xcurrent); that is, Xnew is generated by shifting one of the components from the kth component to the $q - 1$th component of Xcurrent in turn backward, and moving the qth component to the kth component.

Step 6. Calculate the objective value of Xnew. Judge the dominance relation among Xnew, X, and Xcurrent. If Xnew dominates X, then update X = Xnew and go to Step 7. Otherwise, if Xnew dominates Xcurrent, then update Xcurrent = Xnew and go to Step 7.

Step 7. Increase iteration number by one, $n = n+1$. If iteration counter is less than its maximum value ($n < Ns$), then go to Step 3.

Step 8. Update *epoch*, *epoch* = *epoch* + 1. If local search stop criterion is not reached (*epoch* < *Epoch*), then go to Step 2. Otherwise, quit.

At the same time, random mutation operator is introduced in the local search of lot splitting chromosome, which is used to search the neighborhood solution of the subbatches splitting under the unchanged operational sequence of the subbatches. The specific procedure is as follows: randomly select a chromosome and its mutant bit and then divide the subbatch sizes again.

Table 1: Results of HDDE and the other algorithms.

Problem	HDDE	KBACO [22]		PVNS [23]		hGA [24]		eGA [25]	
	C^*	$C_{\max}$	dev (%)	$C_{\max}$	dev (%)	$C_{\max}$	dev (%)	$C_{\max}$	dev (%)
C_1	14	14	0	14	0	14	0	- -	- -
C_2	11	11	0	- -	- -	- -	- -	- -	0
C_3	7	7	0	7	0	7	0	- -	- -
C_4	11	11	0	12	8.33	- -	- -	- -	- -
C_5	40	39	−2.56	40	0	40	0	40	0
C_6	26	29	10.34	26	0	26	0	26	0
C_7	204	204	0	204	0	204	0	204	0
C_8	60	65	7.69	60	0	60	0	60	0
C_9	173	173	0	173	0	173	0	173	0
C_{10}	63	67	5.97	60	−5.00	60	−5.00	58	−8.62
C_{11}	139	144	3.47	141	1.42	140	0.71	144	3.47
C_{12}	523	523	0	523	0	523	0	523	0
C_{13}	311	311	0	307	−1.30	307	−1.30	307	−1.30
C_{14}	221	229	3.49	208	−6.25	205	−7.80	198	−11.62

3.4. Algorithm Procedure. In this paper, the HDDE algorithm is presented based on the combination of the DE algorithm and the local search strategy. For simplicity, population evolution of DE is denoted as operation Oper_1, and the local search strategy is denoted as operation Oper_2. The procedure of HDDE solving the lot splitting with equipment capacity constraints in flexible job shop scheduling problem is as follows.

Step 1. Set parameters, such as Np, F, Cr, $G_{\max}$, *Epoch,* and Ns.

Step 2. Population initialization. Use random initialization to generate each individual including the lot splitting chromosome and the lot scheduling chromosome in the population. Moreover set generation $t = 0$.

Step 3. Operation Oper_1. Execute mutation operator and crossover operator on the lot splitting chromosome and the lot scheduling chromosome successively and then generate the mutant individual (V_i^{t+1}) and the trial individual (U_i^{t+1}), respectively.

Step 4. Calculate the fitness values, select to generate the next population (X^{t+1}), and update the optimal value and the optimal individual.

Step 5. Operation Oper_2. Carry out the local search of 10% individuals in the population X^{t+1} to find their neighborhood solution. Then calculate the fitness function and update the optimal value and the optimal individual.

Step 6. If the optimal value of the population has no continuous change for the appointed iterations, then reinitialize 10% individuals of the population and have the global optimal solution instead of one solution in the population.

Table 2: Information of orders received.

Order number	Product number	Color depth	Demand (kg)	Delivery (hour)	Weight of tardiness penalty
o_1	1	24	90	26	2
	2	60	40	30	1.5
	3	89	100	27	1
o_2	1	24	50	26	1.1
	3	89	90	35	1.6
o_3	1	24	130	48	1.3
	3	89	90	40	1.2
o_4	2	60	110	60	1.9
o_5	4	25	60	9	8
	5	62	120	16	8.5
o_6	6	88	500	48	1
o_7	7	98	60	18	1.4
	8	100	80	36	1.2
o_8	7	98	88	50	2.3
	8	100	100	48	2.1
	9	120	135	64	1.8
o_9	10	145	90	72	3.5

Step 7. Set $t = t + 1$. If $t < G_{\max}$, then go to Step 3. Otherwise, quit.

3.5. Analysis of Complexity of the Proposed Algorithm. As shown in the above procedure, the computation time of HDDE spent is almost in the iterative process. Each individual in the population goes through main operators including difference evolution, decoding, selection, and local search every iteration. Considering a batch splitting scheduling

Table 3: Cleaning time (unit: hour) while each product of different color depth is processed in the vat with the maximum output.

Dye vat number/quantity	Minimum capacity, maximum capacity (kg)	Color depth of each product									
		24	60	89	25	62	88	98	100	120	145
v_1/3	1, 10	0.1	0.15	0.25	0.15	0.5	0.25	0.3	0.3	0.35	0.45
v_2/1	5, 30	0.1	0.15	0.3	0.35	0.25	0.25	0.4	0.45	0.5	0.55
v_3/5	5, 50	0.2	0.3	0.45	0.2	0.3	0.45	0.5	0.5	0.6	0.65
v_4/1	10, 85	0.25	0.35	0.45	0.25	0.35	0.45	0.55	0.6	0.65	0.75
v_5/1	10, 90	0.3	0.45	0.55	0.3	0.45	0.55	0.65	0.65	0.8	0.8
v_6/7	20, 100	0.35	0.5	0.65	0.35	0.5	0.65	0.75	0.85	1.0	1.05
v_7/2	50, 500	0.45	0.65	0.8	0.45	0.65	0.8	0.95	0.95	1.15	1.2

Table 4: Processing time (unit: hour) and cost of different product on different vat with its maximum output.

Dye vat	Color depth of each product														
	p1	p2	p3	p1	p2	p3	p1	p2	p3	p1	p2	p3	p1	p2	p3
		24			60			89			25			62	
v_1	1/10	1.9/25	1.2/12	1.5/18	2.8/30	1.6/20	2/20	4/42	2/22	1/10	1.9/25	1.2/12	1.5/18	2.8/30	1.6/20
v_2	1/12	2/30	1.3/15	1.5/20	3.5/35	1.8/23	2.2/24	4.5/50	2/25	1/12	2/30	1.3/15	1.5/20	3.5/35	1.8/23
v_3	1.5/15	2.5/35	1.8/19	1.8/23	3.5/40	2/28	2.5/28	4.8/55	2.3/30	1.5/15	2.5/35	1.8/19	1.8/23	3.5/40	2/28
v_4	2/18	3/40	2.3/25	2.2/25	4/50	2.6/30	2.5/30	5.2/60	3/33	2/18	3/40	2.2/25	2.2/25	4/50	2.5/30
v_5	2/20	3/45	2.3/28	2.2/30	4/55	2.6/35	2.5/35	5.2/68	3/38	2/20	3/45	2.3/28	2.2/30	4/55	2.6/35
v_6	2.2/25	3.5/50	2.5/30	2.5/35	4.5/60	2.8/40	2.8/40	5.5/70	3.5/45	2.2/25	3.5/50	2.5/30	2.5/35	4.5/60	2.8/40
v_7	2.5/25	4.5/60	3.5/40	3.5/45	5.5/70	3.8/50	3.8/50	6.5/80	4.5/55	2.7/25	4.7/60	3.7/40	3.6/45	5.7/70	3.9/50
		88			98			100			120			145	
v_1	2.2/20	4/42	2/22	2.4/21	4.5/48	2.4/24	2.5/21	4.6/50	2.4/24	3/25	5/68	2.8/26	3.5/26	5.5/75	3.2/27
v_2	2.2/24	4.5/50	2/25	2.5/25	4.7/55	2.5/26	2.6/25	4.8/58	2.5/27	3.2/28	5.5/75	3/30	3.8/30	6.2/82	3.5/32
v_3	2.5/28	4.8/55	2.3/30	2.8/30	5/62	2.6/32	2.9/30	5.2/65	2.7/33	3.5/32	6/78	3.2/34	4/35	6.5/88	3.7/36
v_4	2.5/30	5/60	2.8/33	2.8/32	5.2/70	3.1/35	3/32	5.4/75	3.2/35	3.6/35	6.2/85	3.8/38	4.2/38	6.8/100	4.4/40
v_5	2.5/35	5.2/68	3/38	2.8/38	5.4/80	3.2/40	3/38	5.5/85	3.3/40	3.7/40	6.3/95	3.9/45	4.4/45	7/120	4.5/58
v_6	2.8/40	5.5/70	3.5/45	3/45	5.6/100	3.2/50	3.3/48	6.2/10	3.4/53	3.8/55	6.8/120	4/60	4.5/62	7.2/135	4.6/65
v_7	3.8/50	6.5/80	4.5/55	4/50	6.2/110	4.2/55	4.2/50	6.5/115	4.4/55	4.5/60	7/145	4.8/65	4.8/65	7.5/165	5/70

problem with population size (N_p), the maximal iteration number ($G_{\max}$), dynamic random search times (*Epoch*), and the length of the lot scheduling chromosome (len_1), the time complexity of difference evolution operator is $O(N_p \times \text{len}_1)$, decoding operator is $O(N_p \times \text{len}_1)$, selection operator is $O(N_p)$, and local search operator is $O(\text{len}_1 \times Epoch \times 0.1)$. Therefore, the time complexity of whole algorithm is

$$\begin{aligned} &O\left(N_p, G_{\max}\right) \\ &\quad = G_{\max} \times \left(O\left(N_p \times \text{len}_1\right) + O\left(N_p \times \text{len}_1\right)\right. \\ &\qquad \left. + O\left(N_p\right) + O\left(\text{len}_1 \times Epoch \times 0.1\right)\right) \quad (10) \\ &\quad = G_{\max} \times \left(2O\left(N_p \times \text{len}_1\right)\right. \\ &\qquad \left. + O\left(\text{len}_1 \times Epoch \times 0.1\right) + O\left(N_p\right)\right). \end{aligned}$$

Obviously, the time complexity of HDDE algorithm is related to len_1, N_p, $G_{\max}$, and *Epoch*.

4. Simulation Results and Analysis

4.1. Test Problems and Comparison of Results. We evaluate performance of the HDDE algorithm on 14 test problems with different complexity, in which $C_1 \sim C_4$ designed by Kacem et al. [27, 28] are 8 × 8, 10 × 7, 10 × 10, and 15 × 10 problem, respectively, and $C_5 \sim C_{14}$ designed by Brandimarte [29] are from MK1 to MK10 problem. The proposed algorithm has been coded with VC++ 6.0 and runs on a PC with Intel Pentium CPU 2.66 GHz processor and 1 G of memory. The parameters of HDDE are set as follows: $N_p = 50$, $G_{\max} = 500$, $F = 0.5$, $Cr = 0.7$, $Epoch = 50$, and $Ns = 10$.

The results of HDDE, KBACO [22], PVNS [23], *hGA* [24], and eGA [25] are compared in Table 1, where C^* is the optimal value obtained by HDDE, $C_{\max}$ is the optimal value obtained by the algorithms from literatures, dev denotes the relative error percent between HDDE and the other algorithms, that is, $\text{dev} = (C_{\max} - C^*)/C_{\max} \times 100\%$, and "- -" means that corresponding experiment was not conducted in those literatures. As shown in Table 1, the HDDE algorithm is not worse than any of those algorithms, and it can obtain the global optimal solution for most of the problems, which

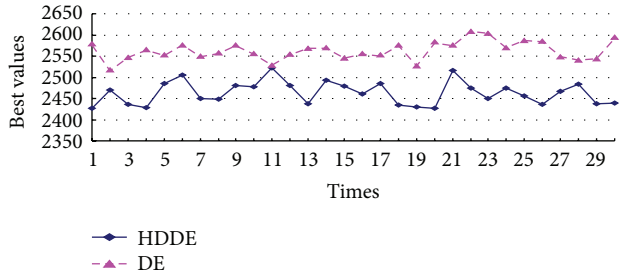


Figure 2: Best values of HDDE and DE obtained over 30 runs.

Table 5: Results of DE and HDDE.

Algorithm	Best	Worst	Average	Standard deviation	Runtime (s)
DE	2518.21	2607.86	2563.29	21.65	1.3
HDDE	2426.6	2521.89	2463.16	26.76	3.2

indicates that the proposed algorithm has better global searching ability. Each algorithm has its own advantages, but it also has certain disadvantage on large scale problems. For instance, the HDDE algorithm has lower global search for C_{13} and C_{14}.

4.2. Practical Dye Vat Scheduling Problem and Discussion of Results. In this paper, we get a certain amount of data samples from actual production in some dyeing workshop. Suppose that the enterprise receives 9 orders, where 10 types of products need to be processed. The products, color depth, demand, delivery duration, and weight of tardiness penalty are shown in Table 2. There are 7 types of dye vats in the dyeing workshop, all of which have different capacity. Information of each vat, such as the number, quantity, minimum capacity, and maximum capacity of each dye vat, is shown in Table 3. There is also the cleaning time (unit: hour), that is, setup time, while the product of different color depth is processed in the vat with the maximum output in Table 3. The processing time (unit: hour) and cost of different product on different vat with its maximum output are shown in Table 4, where p1, p2, and p3 denote three main operations of dyeing process, that is, preprocessing, dyeing, and postprocessing, respectively. The setup time and cost of preprocessing operation of any product are 0. Moreover, suppose that switching cost factors $\alpha = 0.9$ and $\beta = 0.7$.

To evaluate the effectiveness of local search strategies, DE and HDDE run 30 times, independently, and the parameters are set as follows: $N_p = 50$, $G_{\max} = 200$, $F = 0.5$, $Cr = 0.7$, $Epoch = 10$, and $Ns = 3$. Figure 2 and Table 5 show the comparison of results obtained by DE and HDDE over 30 runs and Figure 3 is convergent graph of the optimal value obtained by DE and HDDE. As seen in Figure 2 and Table 5, the local search strategy leads to a significant improvement in the global searching ability of HDDE. The results obtained by HDDE over 30 times are better than those of DE. However, HDDE also consumes a certain amount of computation time. In addition, As we can see from Figure 3, DE does not evolve essentially in later stage with the increasing of iteration.

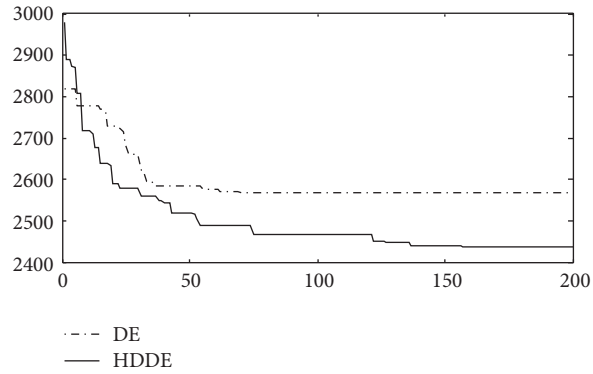


Figure 3: Convergent graph of the optimal solution for DE and HDDE.

On the contrary, HDDE can avoid trapping into the local optimum effectively, and it has higher optimization efficiency than DE.

In general, a larger standard deviation e represents that there is a greater difference between the most value and the average value of solutions, and a smaller standard deviation shows that the most value is close to the average value of solution. As we can see from Figure 3, no more changes are occurred in the optimal value obtained by DE in later iterations; that is, DE easily gets into the local extremum in the later evolution period. Furthermore, as shown in Figure 2 and Table 5, most value of solutions calculated for 30 times is close to the average value. Because of a local search strategy with dynamic random searching based on the critical path and random mutation operator, HDDE can evolve further in later iterations as seen in Figure 3; that is, the local search strategy helps HDDE being out of local minimum. On the other hand, randomness of the local search strategy also results in the bigger difference between some value of solution and the average value as shown in Figure 2 and Table 5.

Finally, the batch number of every product is [3, 2, 3, 1, 2, 5, 2, 2, 2, 1], the optimal lot splitting chromosome is [85, 100, 85, 30, 120, 10, 10, 260, 60, 100, 20, 10, 10, 10, 50, 420, 10, 138, 10, 170, 10, 125, 90] and the optimal lot scheduling chromosome is [52, 81, 32, 64, 101, 101, 101, 31, 63, 32, 82, 33, 51, 92, 71, 91, 22, 12, 22, 41, 32, 63, 81, 21, 41, 81, 22, 72, 72, 31, 21, 62, 12, 72, 63, 31, 13, 64, 21, 65, 51, 62, 52, 12, 62, 41, 71, 33, 71, 13, 61, 82, 52, 11, 64, 91, 33, 82, 11, 61, 65, 65, 13, 61, 92, 11, 51, 92, 91]. As seen from the lot splitting chromosome, most of the subbatches satisfy the requirement of the maximum output of certain dye vat, and each product only has at most one batch unsatisfied. Simulation results show the feasibility of batch splitting and validity of the proposed algorithm.

5. Conclusion

Aiming at lot splitting with equipment capacity constraints in flexible job shop scheduling problem, a two-level coding scheme including the lot splitting chromosome and the lot scheduling chromosome is presented. The new encoding method of the lot splitting chromosome is applied to solve

the batch splitting optimization problem satisfying the maximum processing output of machine. Moreover, according to the faultiness that DE is apt to be trapped into the local optima, the local search strategy with dynamic random searching based on the critical path and random mutation operator is developed, which is, respectively, used to adjust the neighborhood solutions of the lot splitting chromosome and the lot scheduling chromosome. Based on the analysis and comparison of results of benchmark problems, it is shown that the proposed algorithm has better optimization performance beyond most of other global algorithms. Finally, the proposed algorithm is applied to solve the practical scheduling problem of some dyeing workshop. Simulation results show that the proposed algorithm has good global search ability and it can effectively solve the practical lot splitting with equipment capacity constraints in flexible job shop scheduling problem.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (NSFC 61070043, 61105073, and 61203371), the Subproject of National Science and Technology Support Plan (2012BAD10B0101), and the Open Fund Project of Zhejiang University of Technology (20120814).

References

[1] D. Trietsch and K. R. Baker, "Basic techniques for lot streaming," *Operations Research*, vol. 41, no. 6, pp. 1065–1076, 1993.

[2] W. Huang and S. Chen, "Epidemic metapopulation model with traffic routing in scale-free networks," *Journal of Statistical Mechanics*, vol. 2011, Article ID P12004, 19 pages, 2011.

[3] W. Han, J. Chen, and X. Bu, "Batch splitting activity in scheduling of virtual cell with capacity constraints based on Bi-level mathematical model," *Applied Mechanics and Materials*, vol. 263–266, pp. 1257–1264, 2012.

[4] C. Low, C. M. Hsu, and K. I. Huang, "Benefits of lot splitting in job-shop scheduling," *International Journal of Advanced Manufacturing Technology*, vol. 24, no. 9-10, pp. 773–780, 2004.

[5] U. Buscher and L. Shen, "An integer programming formulation for the lot streaming problem in a job shop environment with setups," in *Proceedings of the International MultiConference of Engineers and Computer Scientists (IMECS '11)*, pp. 1343–1348, Hong Kong, March 2011.

[6] U. Buscher and L. Shen, "An integrated tabu search algorithm for the lot streaming problem in job shops," *European Journal of Operational Research*, vol. 199, no. 2, pp. 385–399, 2009.

[7] Q. K. Pan and J. Y. Zhu, "Optimization method for a job-shop scheduling problem with alternative machines in the batch process," *Chinese Journal of Mechanical Engineering*, vol. 40, no. 4, pp. 36–39, 2004.

[8] Z. J. Sun, J. An, and W. Q. Huang, "Lot scheduling with multiple process routes in job shop," *China Mechanical Engineering*, vol. 19, no. 2, pp. 183–187, 2008.

[9] J. An, *Optimization of Job-Shop Scheduling Problem in the Batch Process*, Nanjing University of Aeronautics and Astronautics, Nanjing, China, 2005.

[10] N. Lin, B. Meng, and Y. Fan, "Hybrid genetic algorithm for multiple process and batch scheduling in job-shop," *Journal of Beijing University of Aeronautics and Astronautics*, vol. 33, no. 12, pp. 1471–1476, 2007.

[11] Q. Y. Ju and J. Y. Zhu, "Multi-objective flexible job shop scheduling of batch production," *Chinese Journal of Mechanical Engineering*, vol. 43, no. 8, pp. 148–154, 2007.

[12] J. J. Bai, Y. G. Gong, N. S. Wang, and D. B. Tang, "Multi-objective flexible Job Shop scheduling with lot-splitting," *Computer Integrated Manufacturing Systems*, vol. 16, no. 2, pp. 396–403, 2010.

[13] R. H. Huang, "Multi-objective job-shop scheduling with lot-splitting production," *International Journal of Production Economics*, vol. 124, no. 1, pp. 206–213, 2010.

[14] R. Storn and K. Price, *Differential Evolution—A Simple And Efficient Adaptive Scheme For global optimization Over Continuous spaces*, University of California, Berkeley, Calif, USA, 1995.

[15] Q. K. Pan, L. Wang, and B. Qian, "A novel differential evolution algorithm for bi-criteria no-wait flow shop scheduling problems," *Computers and Operations Research*, vol. 36, no. 8, pp. 2498–2511, 2009.

[16] H. Y. Wang, Y. W. Zhao, X. L. Xu, and W. L. Wang, "Scheduling batch and continuous process production based on an improved differential evolution algorithm," *System Engineering Theory and Practice*, vol. 29, no. 11, pp. 157–167, 2009.

[17] H. Y. Sang, Q. K. Pan, Y. X. Pan, and L. Wu, "A discrete differential evolution algorithm for lot-streaming flow shop scheduling problem," *Computer Simulation*, vol. 27, no. 7, pp. 292–295, 2010.

[18] Y. W. Zhao, H. Y. Wang, X. L. Xu, and W. L. Wang, "A new hybrid parallel algorithm for consistent-sized batch splitting job shop scheduling on alternative machines with forbidden intervals," *International Journal of Advanced Manufacturing Technology*, vol. 48, no. 9–12, pp. 1091–1105, 2010.

[19] Y. Zhao, H. Wang, W. Wang, and X. Xu, "New hybrid parallel algorithm for variable-sized batch splitting scheduling with alternative machines in job shops," *Chinese Journal of Mechanical Engineering*, vol. 23, no. 4, pp. 484–495, 2010.

[20] H. Y. Wang, Y. W. Zhao, W. L. Wang, and X. L. Xu, "New parallel algorithm based on DE for batch splitting job shop scheduling under multiple-resource constraints," *Control and Decision*, vol. 25, no. 11, pp. 1635–1644, 2010.

[21] G. Y. Shi, "A genetic algorithm applied to a classic job-shop scheduling problem," *International Journal of Systems Science*, vol. 28, no. 1, pp. 25–32, 1997.

[22] L. N. Xing, Y. W. Chen, P. Wang, Q. S. Zhao, and J. Xiong, "A knowledge-based ant colony optimization for flexible job shop scheduling problems," *Applied Soft Computing Journal*, vol. 10, no. 3, pp. 888–896, 2010.

[23] M. Yazdani, M. Amiri, and M. Zandieh, "Flexible job-shop scheduling with parallel variable neighborhood search algorithm," *Expert Systems with Applications*, vol. 37, no. 1, pp. 678–687, 2010.

[24] J. C. Tang, G. J. Zhang, B. B. Lin, and B. X. Zhang, "A hybrid algorithm for flexible Job-shop scheduling problem," in *Proceedings of the International Conference on Advanced in Control Engineering and Information Science*, pp. 3678–3683, Guangzhou, China, 2011.

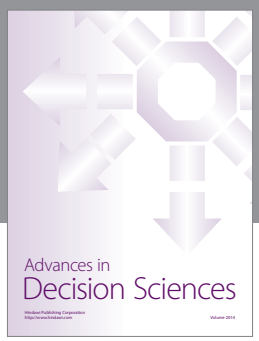
[25] G. Zhang, L. Gao, and Y. Shi, "An effective genetic algorithm for the flexible job-shop scheduling problem," *Expert Systems with Applications*, vol. 38, no. 4, pp. 3563–3573, 2011.

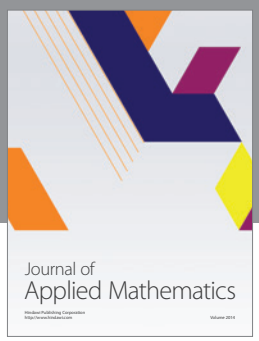
[26] H. Coskun and F. Kutay, "Continuous functions minimization by dynamic random search technique," *Applied Mathematical Modelling*, vol. 31, no. 10, pp. 2189–2198, 2007.

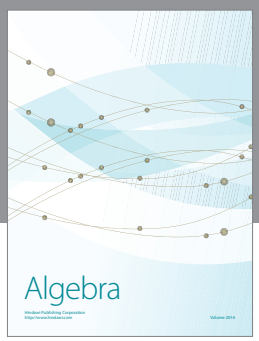
[27] I. Kacem, S. Hammadi, and P. Borne, "Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic," *Mathematics and Computers in Simulation*, vol. 60, no. 3–5, pp. 245–276, 2002.

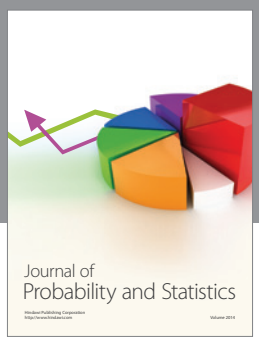
[28] I. Kacem, S. Hammadi, and P. Borne, "Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems," *IEEE Transactions on Systems, Man and Cybernetics C*, vol. 32, no. 1, pp. 1–13, 2002.

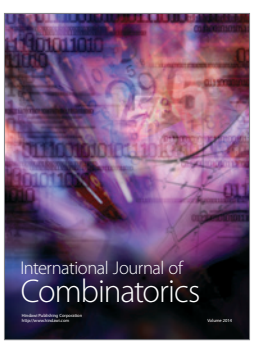
[29] P. Brandimarte, "Routing and scheduling in a flexible job shop by tabu search," *Annals of Operations Research*, vol. 41, no. 3, pp. 157–183, 1993.

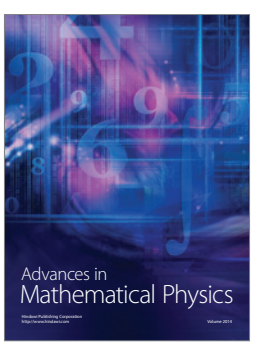


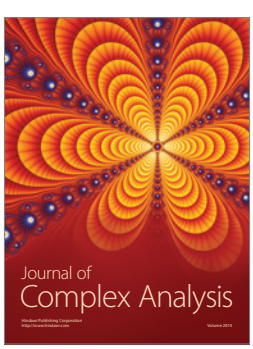


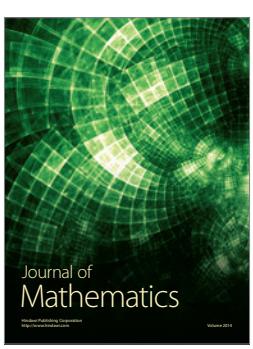


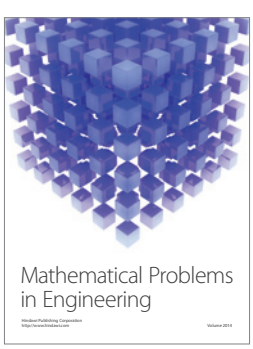


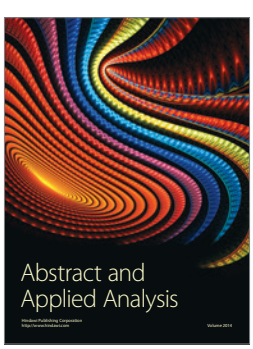


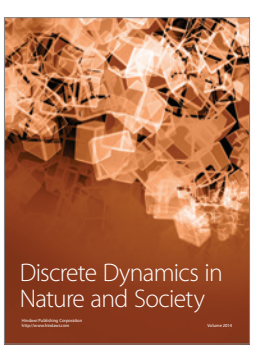


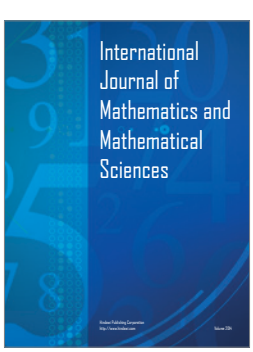












Hindawi

Submit your manuscripts at
http://www.hindawi.com