Research Article

Diversity Controlling Genetic Algorithm for Order Acceptance and Scheduling Problem

Cheng Chen, Zhenyu Yang, Yuejin Tan, and Renjie He

College of Information System and Management, National University of Defense Technology, Changsha 410073, China

Correspondence should be addressed to Cheng Chen; chencheng.prc@gmail.com

Received 2 November 2013; Accepted 9 January 2014; Published 23 February 2014

Academic Editor: Wei-Chiang Hong

Copyright © 2014 Cheng Chen 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.

Selection and scheduling are an important topic in production systems. To tackle the order acceptance and scheduling problem on a single machine with release dates, tardiness penalty, and sequence-dependent setup times, in this paper a diversity controlling genetic algorithm (DCGA) is proposed, in which a diversified population is maintained during the whole search process through survival selection considering both the fitness and the diversity of individuals. To measure the similarity between individuals, a modified Hamming distance without considering the unaccepted orders in the chromosome is adopted. The proposed DCGA was validated on 1500 benchmark instances with up to 100 orders. Compared with the state-of-the-art algorithms, the experimental results show that DCGA improves the solution quality obtained significantly, in terms of the deviation from upper bound.

1. Introduction

In make-to-order production systems with limited capacity and tight delivery requirements, the order acceptance and scheduling problem has been considered as an important topic. Order acceptance decides which orders are to be accepted from all of the orders submitted by customers, while scheduling fixes on the start time for processing each accepted order. Accepting orders without taking into account the capacity and delivery requirements may postpone some of the orders and decrease the revenue. Therefore, joint decisions for acceptance and scheduling have to be made simultaneously in order to maximize the total net revenue.

Diverse problems that combine the decisions of selection and scheduling have been studied by both researchers as well as practitioners over the past two decades. Slotnick [1] summarized variants of acceptance and scheduling problems under different settings and objectives from the literature. In this study, we focus on a static single machine environment, which is the fundamental of other researches in this area. Typical exact algorithms have been applied to solve the problem. For example, Hall and Magazine [2] incorporated Lagrangian relaxation into a dynamic programming without considering sequence-dependent setup times and tardiness

penalty. Gordon and Strusevich [3] presented a dynamic programming for similar problem where the processing time of each job depended on its position in the processing sequence. Charnsirisakskul et al. [4, 5] put forward a mixed-integer programming method for single machine production system where setup times were negligible. Oğuz et al. [6] gave a mixed-integer linear programming formulation of order acceptance and scheduling problem, which can be solved optimally for problems with up to 15 orders. Nobibon and Leus [7] devised two exact branch-and-bound procedures for a generalization of the order acceptance and scheduling problem. Slotnick and Morton [8] developed an optimal branch-and-bound procedure using a linear(integer) relaxation for bounding and performed job acceptance and sequencing jointly. However, due to the strongly NP-hard complexity of the problem [6], only small or moderate size order acceptance and scheduling problems can be optimally solved using exact algorithms. Thus, it is crucial to apply advanced approximate algorithms to solve problems in practical applications with large numbers of orders.

In terms of approximate algorithms, Cesaret et al. [9] proposed a tabu search algorithm supported with a probabilistic local search procedure for order acceptance and scheduling problem with an objective of maximizing total revenue minus

tardiness penalty. Lin and Ying [10] presented an artificial bee colony algorithm combined with an iterated greedy heuristic for exactly the same problem. Rom and Slotnick [11] proposed a genetic algorithm complemented with a probabilistic local search for order acceptance and scheduling under the assumption that all orders were released at the same time. Xiao et al. [12] extended the order acceptance problem to the case of permutation flow shop scheduling and put forward a simulated annealing algorithm coupled with partial optimization. Yang and Geunes [13] studied a single resource scheduling problem with job-selection flexibility, tardiness costs, and controllable processing times and proposed a greedy randomized adaptive search procedure.

In our previous work [14], we proposed an improved genetic algorithm with local search (IGAL) for order acceptance and scheduling problem. Genetic algorithm is an effective metaheuristic algorithm for global optimization. However, traditional genetic algorithm often takes comparatively long time to converge to global optima [15, 16]. Hastily congregating individuals within a small region of the search space lead to premature population convergence [17], which makes the algorithm stuck into local optima. Maintaining a diversified population is an effective way to prevent premature convergence. Diversity can be measured either at the genotype level or the phenotype level [18]. Vidal [19, 20] characterized each individual in the population by a biased fitness composed of its solution cost as well as its diversity contribution for the vehicle routing problems with time windows. The diversity contribution was defined as the average Hamming distance from one individual to its closest neighbors. For the numerical optimization problems, Mc Ginley [17] introduced standard population diversity (SPD) and healthy population diversity (HPD) to adapt the genetic operators. SPD described the level of solution space diversity while HPD described the level of fitness-weighted diversity. The gene-wise Euclidean distance and fitness-weighted distance are used as the measure of SPD and HPD, respectively. Masisi et al. [21] and Misevičius [22] applied the entropy as genotype diversity measurement. Actually, as Burke et al. [23] has pointed out, the diversity measurements are problem specific. There is no single measurement that fits all problems [18].

In this paper, we intend to study how to improve the IGAL by introducing a diversity controlling mechanism into genetic algorithm. A new diversity controlling genetic algorithm (DCGA), which extends our previous work in [14] by introducing a diversity controlling mechanism, is presented. The diversity measure designed for the order acceptance and scheduling problem is different from the measures mentioned in the last paragraph. Not all the orders in the chromosome are considered for calculating the diversity. We do not take into account the orders that are not accepted. Only the accepted orders which contribute to the fitness in the solution are considered. In the survival selection operator, instead of the commonly used “no fitness duplicates” [24] strategy, we allow for individuals with duplicate fitness only if they are distinct from each other to a certain extent according to the diversity measurement. Based on the diversity controlling mechanism, a healthy diverse population is

maintained during the search process. Extensive experiments are conducted on 1500 instances with up to 100 orders. The experimental results illustrate that the performance of genetic algorithm is improved significantly through diversity controlling during the search process. The efficacy of the proposed algorithm is also verified by comparing with several other algorithms from the literature.

The remainder of this paper is organized as follows. Section 2 describes the order acceptance and scheduling problem formally. Section 3 details the proposed diversity controlling genetic algorithm. Section 4 presents the experimental studies. Finally, Section 5 concludes this paper with some remarks and future research directions.

2. Order Acceptance and Scheduling Problem

The order acceptance and scheduling problem can be described as follows. In a single machine environment, there are n incoming orders to be processed. No precedence constraints exist between the orders. The machine can only process one order at any time without preemption. The machine needs processing time p_i to process order i . Order i is released at time r_i , after which the order is available for processing. If order i is accepted, the completion time for processing the order should be no later than its deadline dl_i . Otherwise, the customer will not pay for the order for its too late delivery. If order i is accepted and delivered without tardiness, that is to say, the completion time c_i is no later than the due date d_i , revenue e_i is obtained. If tardiness happens to order i , the revenue of it decreases by $w_i = e_i/(dl_i - d_i)$ per unit time delay beyond due date d_i ($d_i < dl_i$). We denote the revenue gained for accepting order i by $re_i = x_i \cdot (e_i - w_i \cdot T_i)$, where x_i is an indicator which equals to 1 if order i is accepted, and 0 otherwise. $T_i = \max\{0, c_i - d_i\}$ is the tardiness of the order. The objective is to find a subset of orders and fix on start processing time of them, which maximize the total net revenue denoted by $\sum_{i=1}^n re_i$. This problem has been proved to be of strongly NP-hard complexity [6].

3. Diversity Controlling Genetic Algorithm

The proposed diversity controlling genetic algorithm starts with initializing the population to a group of chromosomes, each of which represents a feasible solution. The fitness of each chromosome is calculated to evaluate the quality of the solution. A subset of individuals from the population are selected, two of which are chosen for mating operation (crossover, mutation) to generate offspring. This procedure is repeated until it has reached the maximum number of children in the subpopulation. Then survival selections are performed to generate population for the next generation. The algorithm stops and returns the best feasible solution until it has reached the stopping criterion. Algorithm 1 illustrates the procedure of diversity controlling genetic algorithm.

The population initialization, fitness evaluation, crossover operator, mutation operator, and local search methods are exactly the same as our previous work [14]. They are summarized from Section 3.1 to Section 3.5.

Algorithm DCGAInitialize parent population G_F (N individuals) and subpopulation $G_S = \text{NULL}$ While number of iterations $< It_{\max}$ and number of iterations without improvement $< It_{ni}$ Add the ne fittest individuals in G_F into G_S While the number of individuals in $G_S < M$ Select a subgroup G_m (λ individuals) without replacement through tournament selection Set the fittest individual in G_m as a parent P_1 and the most different individual from P_1 as parent P_2 Generate offspring from P_1 and P_2 (crossover, mutation) Add the generated offspring into G_S Select N survivors from G_S and replace all the individuals in G_F with them Set $G_S = \text{NULL}$ Perform local search on randomly selected nl individuals among the 30% fittest individuals in G_F

Return the best feasible solution

ALGORITHM 1: Diversity controlling genetic algorithm.

3.1. Population Initialization. We adopt the permutation representation as a chromosome for an individual. For a problem with n orders, a sequence of integers ranging from 1 to n represent a solution where each gene corresponds to an order. Random key representation [25] is applied here to generate initial chromosome. We first generate a sequence of n random decimal numbers in $(0, 1)$ and then sort them in ascending order. The position of each decimal number in the original sequence is recorded as an integer, which represents an order. For example, according to this method, a decimal sequence (0.33, 0.92, 0.48, 0.18, 0.76) represents an integer sequence (4, 1, 3, 5, 2). We randomly generate $2 \times N$ individuals and select best half of them as the initial parent population G_F .

3.2. Fitness Evaluation. The processing sequence of each order is fixed according to its position in the chromosome. The fitness evaluation for each chromosome is described as follows.

- (i) Choose next order in the sequence to evaluate until it has reached the end of the sequence.
- (ii) Try to set the start time of chosen order at the earliest possible start time. The order is accepted if the end time does not go beyond the deadline and rejected otherwise. If the chosen order is accepted then the revenue gained by accepting it is recorded. In a situation where order i is accepted and order j is the succeeding order to be evaluated, the earliest possible start time of order j is $\max\{et_i, r_j\} + s_{ij}$; et_i is the end time of order i .
- (iii) The revenue of each accepted order adds to the total net revenue of the solution.

3.3. Crossover. A crossover operator recombines the genes of two chromosomes and generates offspring that inherit part of the characters of the parents. A diversity of crossover operators have been reviewed by Potts et al. [26] and by Poon and Carter [27]. In this study, we apply the same-site-copy-first principle [28], which was proposed to solve the production scheduling problem at first, to perform crossover

FIGURE 1: Illustration of crossover with the same-site-copy-first principle.

operation in order acceptance and scheduling problem. The crossover procedure is described as follows.

- (i) Any gene code which takes up the same position of both parents is also fixed on that position of the offspring.
- (ii) The remaining positions in the offspring are assigned by the order of all gene codes in Parent 1 within the sequence bounded by two randomly selected points.
- (iii) The remaining unassigned positions are placed in the order of appearance in Parent 2.

The crossover with same-site-copy-first principle is illustrated in Figure 1.

3.4. Mutation. Mutation is adopted to prevent the newly formed children from being trapped into their particular local optima. Several mutation operators have been invented for permutation problems, such as adjacent two-change, arbitrary two-change, arbitrary three-change, and shift-change [29]. Shift-change is applied to the mutation operation here because it changes the actual positions of some gene codes to obtain diversity while keeps the relative positions of some gene codes to inherit gene traits of parents. For a shift-change procedure, select two positions randomly and replace one selected position with the other one. Then shift all positions within the sequence bounded by the two selected positions. The shift-change mutation procedure is illustrated in Figure 2.

FIGURE 2: Illustration of shift-change mutation.

3.5. Local Search. To improve the generated solutions, genetic algorithms have been usually complemented by local search. Local search can improve current solution by selecting the best from the neighborhood, but it can also add to the computational cost for large size instances. Considering this, we apply two simple but effective local search strategies to improve the solutions generated at each iteration. For the first local search procedure, we successively interchange the positions of two immediate orders in the sequence and record the best interchange as a new solution. For the second local search procedure, we apply the iterated greedy procedure proposed by Ruiz and Stützle [30]. The iterated greedy procedure applies the destruction phase and construction phase iteratively. The destruction phase eliminates some orders from the incumbent solution while the construction phase reinserts the eliminated orders sequentially into the sequence. The iterated greedy procedure is also applied by Lin and Ying [10]. For detailed descriptions and illustrations of the procedure, the reader is referred to the work of Ruiz and Stützle [30] and the work of Lin and Ying [10].

3.6. Diversity Controlling. Comparing with our previous work in [14], the algorithm proposed in this paper adopts a diversity controlling mechanism in the search process. An improved distance measurement is proposed to measure the difference between individuals. In the parent selection procedure, we take into account not only the fitness of individuals but also the diversity between them. In the survival selection procedure, we select a diversity of individuals which are distinct from each other even though they might share same fitness values.

3.6.1. Difference Measurement. The Hamming distance is a frequently used measurement indicating the difference between two individuals. The difference between two individuals represented by a normalized Hamming distance [20] is defined as

$$\text{diff}(u, v) = \frac{1}{n} \sum_{i=1}^n 1(u[i] \neq v[i]), \quad (1)$$

where $u[i]$ and $v[i]$ are the i th and j th genes of individual u and v . $1(\text{cond})$ returns 1 if the condition cond is true, 0, otherwise. However, in order acceptance and scheduling problem, not all the orders represented in chromosomes can be accepted. When calculating the Hamming distance between two individuals, taking into account all the orders, a subset of which are not accepted and scheduled, might not reflect the real difference between them.

For example, Figure 3 illustrates two individuals composed of 10 orders. The integer in shadow indicates that

Individual A	6	1	2	4	7	9	3	10	5	8
Individual B	6	7	1	4	2	9	3	10	5	8
Individual C	6	7	2	3	1	9	4	10	8	5

FIGURE 3: Illustration of difference measurement.

the corresponding order is accepted, and not accepted otherwise. From the Hamming distance perspective, $\text{diff}(A, B) = 30\%$, $\text{diff}(A, C) = 60\%$. According to this measurement, it seems that individuals A and B are more similar than individuals A and C . But in fact, the accepted orders of individual A (6, 1, 2, 7, 9, 10) and the accepted orders of individual B (6, 7, 1, 2, 9, 10) are less similar than that of individual A and that of individual C (6, 7, 2, 1, 9, 10).

We put forward a new difference measurement taking into consideration only the orders that are accepted. It can be defined as

$$\begin{aligned} \overline{\text{diff}}(u, v) \\ = \frac{1}{n} \sum_{i=1}^n 1(u[i] \neq v[i]) (x[u[i]] = 1) (x[v[i]] = 1), \end{aligned} \quad (2)$$

where $x[u[i]] = 1$ and $x[v[i]] = 1$ indicate that the orders in i th gene of individuals u and v are accepted, respectively. According to this new measurement, $\overline{\text{diff}}(A, B) = 30\%$, $\overline{\text{diff}}(A, C) = 20\%$, which explains that individuals A and C are actually more similar than individuals A and B .

3.6.2. Survival Selection. Survival selection decides which individuals in the subpopulation are to be selected as the parent population in the next generation. The objective of this problem is to maximize the total net revenue of accepted orders, so we sort the individuals in the subpopulation by fitness in descending order. If a subset of individuals that share same fitness exist, we remove some of them that are too similar with each other according to the difference measurement. If the number of remaining individuals in the subpopulation is less than the size of parent population then we randomly generate new individuals together with the remaining individuals in the subpopulation to form father population, else N best fit remaining individuals in the subpopulation are selected as the father population in the next generation. The survival selection procedure is described in Algorithm 2.

To observe the diversity change during the search process, we define the population diversity as

$$\text{Div}(G) = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \overline{\text{diff}}(G[i], G[j]), \quad i \neq j, \quad (3)$$

where $G[i]$ and $G[j]$ are the i th and j th individuals in population G .

TABLE 1: Results for problem set $n = 10$.

τ	R	Deviation from UB (%)																	
		m-ATCS			ISFAN			TS			ABC			IGAL			DCGA		
		Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.
0.1	0.1	25	9	0	10	3	0	3	1	0	0	0	0	0	0	0	0	0	0
	0.3	19	9	2	9	3	0	3	1	0	2	0	0	0	0	0	0	0	0
	0.5	27	9	0	8	3	0	8	1	0	6	1	0	6	1	0	6	1	0
	0.7	23	10	0	10	3	0	3	0	0	0	0	0	0	0	0	0	0	0
	0.9	21	8	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0.3	0.1	19	10	0	12	4	0	2	0	0	0	0	0	0	0	0	0	0	0
	0.3	33	14	0	10	2	0	2	1	0	0	0	0	0	0	0	0	0	0
	0.5	40	16	0	19	7	0	0	0	0	2	0	0	0	0	0	0	0	0
	0.7	36	12	1	16	6	0	2	1	0	0	0	0	0	0	0	0	0	0
	0.9	42	21	5	29	6	0	2	0	0	0	0	0	0	0	0	0	0	0
0.5	0.1	24	14	4	20	9	0	6	1	0	0	0	0	0	0	0	0	0	0
	0.3	34	14	0	20	5	0	5	1	0	0	0	0	0	0	0	0	0	0
	0.5	23	15	3	9	3	0	0	0	0	0	0	0	0	0	0	0	0	0
	0.7	40	16	1	32	6	0	2	0	0	0	0	0	0	0	0	0	0	0
	0.9	35	15	1	28	6	0	0	0	0	0	0	0	0	0	0	0	0	0
0.7	0.1	34	14	0	23	6	0	4	0	0	0	0	0	0	0	0	0	0	0
	0.3	27	14	0	23	7	0	0	0	0	0	0	0	0	0	0	0	0	0
	0.5	49	24	4	12	4	0	1	0	0	0	0	0	0	0	0	0	0	0
	0.7	26	13	0	19	8	0	0	0	0	0	0	0	0	0	0	0	0	0
	0.9	36	20	9	19	6	0	4	0	0	0	0	0	0	0	0	0	0	0
0.9	0.1	17	8	0	11	2	0	0	0	0	0	0	0	0	0	0	0	0	0
	0.3	29	11	0	18	7	0	0	0	0	0	0	0	0	0	0	0	0	0
	0.5	22	9	0	27	8	0	0	0	0	0	0	0	0	0	0	0	0	0
	0.7	34	15	6	39	11	0	0	0	0	0	0	0	0	0	0	0	0	0
	0.9	40	19	2	27	6	0	0	0	0	0	0	0	0	0	0	0	0	0

Algorithm Survival Selection
Sort individuals in G_s by fitness descending;
Set $G_F = \text{NULL}$
Initialize a Boolean array $\text{needRemove}[i] = \text{false}, i = 1, \dots, M$;
For $i = 1$ to M
 If $\text{needRemove}[i]$ continue;
 For $j = i + 1$ to M
 If $\text{needRemove}[j]$ continue;
 If fitness of individual $i >$ fitness of individual j continue;
 If $\text{diff}(\text{indiv } i, \text{indiv } j) < \varepsilon$
 $\text{needRemove}[j] = \text{true}$;
 $M = M - 1$;
 If (number of individuals in $G_F < N$ and $\text{needRemove}[i] = \text{false}$)
 Add individual i to G_F
While (number of individuals in $G_F < N$)
 Randomly generate a new individual and add it to G_F

ALGORITHM 2: Survival selection.

TABLE 2: Results for problem set $n = 15$.

τ	R	Deviation from UB (%)																	
		m-ATCS			ISFAN			TS			ABC			IGAL			DCGA		
		Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.
0.1	0.1	17	10	4	15	8	1	4	3	1	4	2	0	4	2	0	3	2	0
	0.3	23	16	8	12	6	2	8	3	0	7	3	1	6	3	0	6	2	0
	0.5	18	11	4	8	4	2	4	2	0	4	2	0	4	1	0	3	1	0
	0.7	29	11	3	9	4	0	4	2	0	4	1	0	4	1	0	4	1	0
	0.9	28	10	0	11	5	0	6	1	0	6	1	0	6	1	0	6	1	0
0.3	0.1	24	15	7	8	6	3	8	4	3	6	3	0	6	3	0	6	3	0
	0.3	41	19	6	19	9	4	11	5	2	11	4	1	11	4	0	11	4	0
	0.5	29	19	9	12	7	5	8	5	1	8	4	1	7	4	1	7	4	1
	0.7	36	20	6	34	11	3	8	4	2	8	3	1	8	3	1	8	3	1
	0.9	31	18	3	13	6	0	7	4	0	7	3	0	7	3	0	7	3	0
0.5	0.1	27	20	12	19	11	2	13	7	2	13	7	2	13	7	2	13	7	2
	0.3	33	22	8	18	10	4	14	8	4	14	8	4	14	7	3	14	7	3
	0.5	35	26	10	20	12	4	15	9	6	15	9	4	15	9	4	15	9	4
	0.7	44	26	11	25	9	4	11	6	1	11	6	0	10	6	0	10	6	0
	0.9	40	21	7	21	8	0	18	6	0	18	6	0	18	6	0	18	6	0
0.7	0.1	26	15	0	7	3	0	4	1	0	2	0	0	0	0	0	0	0	0
	0.3	30	14	0	5	3	0	3	1	0	0	0	0	0	0	0	0	0	0
	0.5	44	20	12	12	4	0	2	0	0	0	0	0	0	0	0	0	0	0
	0.7	33	19	8	20	7	0	8	2	0	8	1	0	8	1	0	8	1	0
	0.9	29	17	6	16	7	0	0	0	0	0	0	0	0	0	0	0	0	0
0.9	0.1	27	10	0	14	4	0	3	0	0	0	0	0	0	0	0	0	0	0
	0.3	35	17	8	10	4	0	5	0	0	0	0	0	0	0	0	0	0	0
	0.5	24	17	2	12	4	0	0	0	0	0	0	0	0	0	0	0	0	0
	0.7	25	16	3	33	9	0	0	0	0	0	0	0	0	0	0	0	0	0
	0.9	39	21	3	28	9	0	0	0	0	0	0	0	0	0	0	0	0	0

FIGURE 4: Illustration of population diversity change.

FIGURE 5: Illustration of change of average number of accepted orders.

Figure 4 illustrates the population diversity change on test dataset Dataslack_100orders_Tao9R9_1 after 1500 generations. At the first few generations, the population diversity is relatively small. This is because the population is randomly initialized and only few orders are accepted in the random

generated chromosomes. Figure 5 explains the change of the average number of accepted orders and Figure 6 shows the change of best fitness. We can see that as the search proceeds, the population diversity does not show obvious

TABLE 3: Results for problem set $n = 20$.

τ	R	Deviation from UB (%)																	
		m-ATCS			ISFAN			TS			ABC			IGAL			DCGA		
		Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.
0.1	0.1	17	11	6	10	7	2	5	3	1	4	2	0	4	2	0	2	2	0
	0.3	30	17	5	13	9	3	5	3	1	2	2	1	2	2	1	2	2	1
	0.5	23	17	4	8	5	1	3	2	1	3	2	0	3	1	0	3	1	0
	0.7	23	12	6	10	5	0	2	1	0	1	0	0	1	0	0	1	0	0
	0.9	24	10	0	17	6	0	4	1	0	3	1	0	3	0	0	3	0	0
0.3	0.1	28	20	9	18	11	7	8	5	2	7	4	2	7	3	0	7	3	0
	0.3	27	20	12	15	9	5	6	5	3	6	4	1	6	3	1	5	3	1
	0.5	26	17	9	12	8	5	8	5	2	6	4	1	6	3	1	5	3	1
	0.7	31	17	5	12	7	2	7	3	1	7	2	1	7	2	0	7	2	0
	0.9	28	18	5	13	7	1	5	2	0	5	2	0	5	1	0	5	1	0
0.5	0.1	26	20	12	13	10	7	8	6	4	6	5	2	6	4	2	5	4	2
	0.3	63	26	11	18	12	5	9	6	3	9	5	1	8	5	1	8	5	1
	0.5	37	27	19	17	13	9	10	7	3	9	5	2	9	5	2	9	5	2
	0.7	35	23	13	17	12	7	11	5	1	11	5	1	11	4	1	11	4	1
	0.9	38	26	10	23	14	5	11	6	0	11	6	0	11	5	0	11	5	0
0.7	0.1	30	22	17	20	14	8	15	10	5	14	9	5	14	8	2	14	8	2
	0.3	33	24	10	18	13	10	14	9	6	13	8	6	12	8	6	12	8	6
	0.5	47	35	19	24	14	5	20	9	0	20	9	0	20	9	0	20	9	0
	0.7	40	25	9	24	13	0	12	7	0	12	6	0	12	6	0	12	6	0
	0.9	44	29	7	34	17	8	13	8	0	12	7	0	12	7	0	12	7	0
0.9	0.1	18	10	1	8	4	0	1	0	0	0	0	0	0	0	0	0	0	0
	0.3	26	15	7	24	7	0	1	0	0	1	0	0	0	0	0	0	0	0
	0.5	43	20	8	24	8	1	2	0	0	0	0	0	0	0	0	0	0	0
	0.7	46	21	11	13	6	0	1	0	0	0	0	0	0	0	0	0	0	0
	0.9	39	27	12	18	12	0	13	2	0	13	1	0	13	1	0	13	1	0

change. The population after 1500 generations is as diversified as it is when the search just starts.

3.6.3. Parent Selection for Mating. The Best-Last Mating [31] with similarity preference is applied here. We select λ individuals without replacement through tournament selection from parent population. The fittest one in the λ individuals is selected as parent P_1 . The other individual that is most different from P_1 is selected as parent P_2 . The selected parents P_1 and P_2 then undergo crossover and mutation with probabilistic p_c and p_m , respectively, to generate new offspring.

4. Experimental Studies

4.1. Test Instances. The proposed DCGA was evaluated on 1500 benchmark instances with various problem sizes and parameter values. The data sets were designed by Cesaret, and can be found in [9]. The instances include three factors: the number of orders (n), the tardiness factor (τ), and the due date range (R). The number of orders was set to 10, 15, 20, 25, 50, and 100. The tardiness factor and due date range were both set to 0.1, 0.3, 0.5, 0.7, and 0.9.

FIGURE 6: Illustration of fitness change.

The procedure to generate the data sets used by Cesaret [9] is described as follows. For each order i ($i = 1, 2, \dots, n$), a processing time p_i and a revenue e_i were randomly generated from the uniform distribution $[0, 20]$. A release date r_i was generated from $[0, \tau p_T]$ where p_T is the total processing

TABLE 4: Results for problem set $n = 25$.

τ	R	Deviation from UB (%)																	
		m-ATCS			ISFAN			TS			ABC			IGAL			DCGA		
		Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.
0.1	0.1	20	14	9	12	8	4	6	4	1	4	3	1	3	2	1	3	2	1
	0.3	32	16	6	11	7	3	6	3	2	6	2	1	4	2	1	3	1	1
	0.5	21	12	3	11	7	3	4	2	1	2	1	0	2	1	0	2	1	0
	0.7	23	13	5	9	5	2	4	1	0	3	1	0	2	0	0	1	0	0
	0.9	24	8	2	10	6	1	2	1	0	2	0	0	1	0	0	1	0	0
0.3	0.1	29	19	13	12	9	5	5	4	2	5	3	1	4	2	1	4	2	1
	0.3	24	18	7	12	10	7	7	5	3	6	3	1	5	3	1	4	3	1
	0.5	26	18	10	10	7	5	6	3	2	5	2	2	3	2	1	3	2	1
	0.7	32	17	8	24	11	6	6	2	1	5	2	1	5	2	1	4	1	0
	0.9	27	16	6	18	9	5	4	2	0	3	1	0	2	1	0	2	1	0
0.5	0.1	29	19	12	14	11	7	7	6	3	7	5	3	6	4	2	6	4	2
	0.3	29	21	11	17	11	6	9	5	3	7	4	3	7	4	2	7	4	2
	0.5	30	20	8	20	13	8	8	5	2	6	4	2	6	4	1	6	4	1
	0.7	38	26	17	17	13	7	11	6	2	12	3	0	7	4	1	7	4	1
	0.9	49	28	16	25	15	6	7	4	1	7	3	1	7	3	1	7	3	1
0.7	0.1	26	20	12	27	18	10	18	9	3	16	8	1	15	7	1	15	7	1
	0.3	33	26	18	23	17	9	14	10	7	12	9	5	12	8	5	12	8	5
	0.5	43	33	26	26	19	13	15	12	7	14	10	5	14	10	5	14	10	5
	0.7	40	29	15	21	15	9	14	8	2	12	7	2	12	6	2	12	6	2
	0.9	53	29	8	24	17	10	15	10	3	14	8	0	14	8	0	14	8	0
0.9	0.1	20	12	4	25	8	0	6	1	0	5	1	0	5	0	0	5	0	0
	0.3	22	17	10	8	5	1	0	0	0	1	0	0	0	0	0	0	0	0
	0.5	45	27	14	44	12	1	12	4	0	12	3	0	12	2	0	12	2	0
	0.7	45	29	17	31	13	0	25	8	0	21	7	0	21	7	0	21	7	0
	0.9	58	30	20	31	13	2	22	7	0	19	6	0	19	6	0	19	6	0

times of all orders. The sequence-dependent setup times s_{ij} ($j = 1, 2, \dots, n, j \neq i$) were generated randomly from the uniform distribution $[0, 10]$. A due date was generated as $d_i = r_i + \max_{j=0,1,\dots,n} s_{ji} + \max\{\text{slack}, p_i\}$, where slack was generated randomly from the uniform distribution $[p_T(1 - \tau - R/2), p_T(1 - \tau + R/2)]$. A deadline was generated as $dl_i = d_i + Rp_i$. A weight was calculated as $w_i = e_i/(dl_i - d_i)$. All of the generated parameters except w_i are integers.

4.2. Parameter Settings. Since the main purpose of this paper is to validate the efficacy of diversity controlling in GA for the order acceptance and scheduling problem, we did not apply any sophisticated parameter control strategies. Instead, we used simple parameter study procedure to select the most suitable fixed parameter settings for DCGA manually. Based on our preliminary experiments, the following parameter setting is suggested. The size of father population G_F and the size of subpopulation G_s are set to 40 and 120, respectively. Crossover probability and mutation probability are set to 0.8 and 0.2, respectively. λ is set to 8 and ϵ is set to 0.1. If the best fitness does not improve within 200 consecutive iterations or maximum 1500 iterations are reached, the algorithm stops and returns the best solutions.

4.3. Experimental Results and Discussions. The proposed diversity controlling genetic algorithm (DCGA) is implemented in C# programming language and run on a PC with Intel i5 CPU (2.4 GHz) and 2 GB RAM, Windows 7 OS. The percentage deviation of the total net revenue from upper bound (UB) is used as the performance measure. The deviation from UB is $100\% \times (\text{UB-feasible solution})/\text{UB}$. The upper bound for each instance was generated by Oğuz et al. [6]. Two bounds were generated and the tighter one was selected as the UB. The first bound was calculated by solving MILP using CPLEX with time limit of 1 hour. The second bound was generated by solving the LP relaxation of the MILP with valid inequalities. The deviation from upper bound on 1500 instances is recorded and compared with that of 5 other algorithms. They are MILP, m-ATCS, ISFAN, TS, and ABC. MILP, m-ATCS, and ISFAN are proposed by Oğuz et al. [6]. m-ATCS refers to modified apparent tardiness cost rule-based heuristic. ISFAN is iterative sequence first-accept next procedure based on simulated annealing. TS is tabu search algorithm proposed by Cesaret et al. [9]. ABC refers to artificial bee colony based algorithm proposed by Lin and Ying [10].

TABLE 5: Results for problem set $n = 50$.

τ	R	Deviation from UB (%)																	
		m-ATCS			ISFAN			TS			ABC			IGAL			DCGA		
		Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.
0.1	0.1	20	15	9	12	8	6	3	2	1	3	2	1	2	2	1	2	1	1
	0.3	18	15	9	12	9	7	4	2	1	2	2	1	2	2	1	1	1	1
	0.5	19	13	6	10	8	5	2	2	1	2	1	0	2	1	0	1	0	0
	0.7	24	14	7	22	9	4	16	3	0	16	2	0	16	2	0	16	2	0
	0.9	10	4	1	11	7	4	2	1	0	0	0	0	0	0	0	0	0	0
0.3	0.1	24	16	11	13	11	8	3	3	2	4	3	2	3	2	1	3	2	1
	0.3	30	20	11	15	12	9	5	4	3	4	3	2	3	2	2	3	2	1
	0.5	33	18	9	15	11	8	5	3	1	4	2	1	4	2	0	4	1	0
	0.7	24	14	5	18	11	7	3	1	0	2	1	0	1	0	0	0	0	0
	0.9	26	17	3	11	9	8	3	1	0	2	1	0	1	0	0	1	0	0
0.5	0.1	27	20	13	19	14	10	5	4	3	4	3	2	3	2	2	3	2	1
	0.3	30	25	18	20	16	11	8	6	3	6	4	3	6	3	2	5	3	2
	0.5	34	25	15	21	16	7	8	4	2	7	4	2	6	3	1	6	3	1
	0.7	31	24	12	18	15	9	5	3	2	4	2	1	4	2	0	4	2	0
	0.9	35	27	17	19	15	6	6	4	2	4	2	1	4	1	0	4	1	0
0.7	0.1	30	22	17	21	16	12	9	7	4	6	5	3	5	4	2	5	4	2
	0.3	33	24	16	20	16	11	11	7	4	9	6	4	8	5	3	8	5	3
	0.5	46	33	23	22	19	14	13	9	7	11	7	5	11	6	5	11	6	5
	0.7	35	31	24	29	19	12	18	9	2	15	8	2	14	7	1	14	7	1
	0.9	41	29	19	28	21	17	18	11	6	14	8	7	13	7	3	13	7	3
0.9	0.1	30	24	17	22	17	12	18	13	8	17	11	7	17	11	7	17	11	7
	0.3	42	32	21	28	23	18	23	17	13	18	14	9	18	13	9	18	13	9
	0.5	45	39	33	26	21	18	23	17	10	17	13	3	17	12	3	17	12	3
	0.7	46	37	26	72	27	17	21	16	11	18	12	6	18	11	6	18	11	6
	0.9	44	36	24	29	22	16	19	16	11	16	12	10	16	12	10	16	12	9

For small size problems, for example $n = 10$, we can find the optimality by enumerating all the feasible solutions within 2 seconds. But due to the combinatorial complexity, most of the moderate size and large size problems cannot be solved to optimality by those algorithms. For this reason, the percentage deviations from UB are reported here instead of the optimal solution numbers for each dataset.

Small size problems are very easy to solve. Tables 1 and 2 illustrate the results for small size problems. Problems with 10 orders can be solved to optimality by MILP and most of them can be solved to optimality by ABC. Our proposed DCGA finds all the optimal solutions for $n = 10$. For problems with 15 orders, the maximum deviations from UB of 2 datasets are improved; the average deviation from UB of 1 dataset is improved.

For moderate size problems, Table 3 shows that for problems with 20 orders, the maximum deviations from upper bound of 4 datasets are improved; the average deviation from upper bound of 1 dataset is improved. The results of problems with 25 orders are given in Table 4. The maximum deviations from upper bound of 4 datasets are improved; the average deviations from upper bound of 2 datasets are improved; and the minimum deviation from upper bound of 1 dataset is improved.

Tables 5 and 6 give the results of large size problems with 50 and 100 orders, respectively. For problems with 50 orders, the maximum deviations from upper bound of 4 datasets are improved; the average deviations from upper bound of 4 datasets are improved; and the minimum deviations from upper bound of 3 datasets are improved. For problems with 100 orders, the maximum deviations from upper bound of 7 datasets are improved, the average deviations from upper bound of 7 datasets are improved, and the minimum deviations from upper bound of 3 datasets are improved.

5. Conclusions

In this paper, we propose a diversity controlling genetic algorithm for the order acceptance and scheduling problem with tardiness penalties, distinct release dates, and sequence-dependent setup times. To measure the difference between individuals, a modified Hamming distance without considering the orders that are not accepted instead of the whole genes in the chromosome was adopted. A diversified population is maintained during the search process through survival selection. By comparing with the state-of-the-art algorithms, the experimental results suggest that the proposed diversity controlling mechanism is effective, and the quality of

TABLE 6: Results for problem set $n = 100$.

τ	R	Deviation from UB (%)																	
		m-ATCS			ISFAN			TS			ABC			IGAL			DCGA		
		Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.
0.1	0.1	19	15	9	13	9	8	3	2	1	2	2	1	2	2	1	1	1	1
	0.3	20	17	11	10	9	7	3	2	2	2	1	1	2	1	1	1	1	1
	0.5	19	14	11	11	9	6	3	1	0	1	1	0	1	1	0	0	0	0
	0.7	16	11	5	12	9	6	1	0	0	0	0	0	0	0	0	0	0	0
	0.9	11	6	0	16	12	8	1	0	0	0	0	0	0	0	0	0	0	0
0.3	0.1	24	19	15	13	12	10	4	3	1	3	2	1	3	2	1	2	1	1
	0.3	23	18	15	17	13	11	5	3	2	4	2	1	3	2	1	3	1	1
	0.5	21	18	14	17	14	10	4	2	1	2	2	1	2	1	1	2	1	0
	0.7	32	20	15	15	14	12	3	2	0	1	1	0	1	0	0	1	0	0
	0.9	18	15	10	17	13	9	2	1	0	1	0	0	0	0	0	0	0	0
0.5	0.1	24	22	17	18	16	12	5	4	2	4	3	3	4	3	2	3	2	2
	0.3	28	23	17	18	15	12	6	4	3	4	3	2	3	3	2	3	2	1
	0.5	31	24	20	19	17	14	5	4	3	4	3	2	4	3	1	3	2	1
	0.7	34	23	10	21	17	13	11	7	4	7	5	2	2	1	0	2	1	0
	0.9	36	24	15	24	18	12	5	2	1	3	1	1	2	1	0	2	1	0
0.7	0.1	28	22	17	19	17	13	6	5	3	5	4	3	4	3	3	4	3	2
	0.3	34	24	18	21	17	13	11	7	4	7	5	2	6	4	2	6	4	2
	0.5	37	26	20	24	18	14	13	6	4	12	5	3	11	5	2	11	5	2
	0.7	38	31	22	23	19	16	13	7	3	7	5	3	7	5	3	7	5	3
	0.9	37	31	23	24	18	15	13	8	5	8	6	4	7	5	3	7	5	3
0.9	0.1	25	22	18	20	17	14	12	9	7	9	7	5	9	6	4	9	6	4
	0.3	36	31	25	26	20	16	23	15	9	12	9	5	12	9	5	12	9	5
	0.5	39	33	25	25	21	19	18	16	13	17	12	9	17	11	8	16	11	8
	0.7	40	36	26	24	21	15	20	16	11	13	11	8	13	11	7	13	11	7
	0.9	40	35	28	28	21	13	22	16	12	17	11	4	17	11	4	17	11	4

obtained solutions was improved significantly on most of the used benchmark instances.

However, the parameters of DCGA were selected manually. We believe that adaptive parameter control strategies would further improve the performance of the algorithm. Adaptive parameter calibration could be a potential direction for future research.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors would like to thank Professor Ceyda Oguz for the helpful discussion with them and for providing the benchmarks. This work was funded under a Grant from the National Natural Science Foundation of China (nos. 71071156 and 61305084).

References

- [1] S. A. Slotnick, "Order acceptance and scheduling: a taxonomy and review," *European Journal of Operational Research*, vol. 212, no. 1, pp. 1–11, 2011.
- [2] N. G. Hall and M. J. Magazine, "Maximizing the value of a space mission," *European Journal of Operational Research*, vol. 78, no. 2, pp. 224–241, 1994.
- [3] V. S. Gordon and V. A. Strusevich, "Single machine scheduling and due date assignment with positionally dependent processing times," *European Journal of Operational Research*, vol. 198, no. 1, pp. 57–62, 2009.
- [4] K. Charnsirisakul, P. M. Griffin, and P. Keskinocak, "Order selection and scheduling with leadtime flexibility," *IIE Transactions*, vol. 36, no. 7, pp. 697–707, 2004.
- [5] K. Charnsirisakul, P. M. Griffin, and P. Keskinocak, "Pricing and scheduling decisions with leadtime flexibility," *European Journal of Operational Research*, vol. 171, no. 1, pp. 153–169, 2006.
- [6] C. Oğuz, F. Sibel Salman, and Z. B. Yalçın, "Order acceptance and scheduling decisions in make-to-order systems," *International Journal of Production Economics*, vol. 125, no. 1, pp. 200–211, 2010.

- [7] F. T. Nobibon and R. Leus, "Exact algorithms for a generalization of the order acceptance and scheduling problem in a single-machine environment," *Computers and Operations Research*, vol. 38, no. 1, pp. 367–378, 2011.
- [8] S. A. Slotnick and T. E. Morton, "Order acceptance with weighted tardiness," *Computers and Operations Research*, vol. 34, no. 10, pp. 3029–3042, 2007.
- [9] B. Cesaret, C. Oğuz, and F. S. Salman, "A tabu search algorithm for order acceptance and scheduling," *Computers and Operations Research*, vol. 39, no. 6, pp. 1197–1205, 2012.
- [10] S.-W. Lin and K.-C. Ying, "Increasing the total net revenue for single machine order acceptance and scheduling problems using an artificial bee colony algorithm," *Journal of the Operational Research Society*, vol. 64, no. 2, pp. 293–311, 2012.
- [11] W. O. Rom and S. A. Slotnick, "Order acceptance using genetic algorithms," *Computers and Operations Research*, vol. 36, no. 6, pp. 1758–1767, 2009.
- [12] Y.-Y. Xiao, R.-Q. Zhang, Q.-H. Zhao, and I. Kaku, "Permutation flow shop scheduling with order acceptance and weighted tardiness," *Applied Mathematics and Computation*, vol. 218, no. 15, pp. 7911–7926, 2012.
- [13] B. Yang and J. Geunes, "A single resource scheduling problem with job-selection flexibility, tardiness costs and controllable processing times," *Computers and Industrial Engineering*, vol. 53, no. 3, pp. 420–432, 2007.
- [14] C. Cheng, Z. Yang, and L. Xing, "An improved genetic algorithm with local search for order acceptance and scheduling problems," in *Proceedings of the IEEE Symposium on Computational Intelligence in Production and Logistics System*, Singapore.
- [15] J.-M. Renders and S. P. Flasse, "Hybrid methods using genetic algorithms for global optimization," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 26, no. 2, pp. 243–258, 1996.
- [16] P. C. Chu and J. E. Beasley, "A genetic algorithm for the generalised assignment problem," *Computers and Operations Research*, vol. 24, no. 1, pp. 17–23, 1997.
- [17] B. Mc Ginley, J. Maher, C. O'Riordan, and F. Morgan, "Maintaining healthy population diversity using adaptive crossover, mutation, and selection," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 5, pp. 692–714, 2011.
- [18] M. Crepinsek, S. H. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: a survey," *ACM Computing Surveys*, vol. 45, no. 3, article 35, pp. 31–35, 2013.
- [19] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins, "A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows," *Computers and Operations Research*, vol. 40, pp. 475–489, 2013.
- [20] T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi, and W. Rei, "A hybrid genetic algorithm for multidepot and periodic vehicle routing problems," *Operations Research*, vol. 60, no. 3, pp. 611–624, 2012.
- [21] L. Masisi, V. Nelwamondo, and T. Marwala, "The use of entropy to measure structural diversity," in *Proceedings of the IEEE 6th International Conference on Computational Cybernetics (ICCC '08)*, pp. 41–45, November 2008.
- [22] A. Misevičius, "Generation of grey patterns using an improved genetic-evolutionary algorithm: some new results," *Information Technology and Control*, vol. 40, no. 4, pp. 330–343, 2011.
- [23] E. Burke, S. Gustafson, and G. Kendall, "Diversity in genetic programming: an analysis of measures and correlation with fitness," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 1, pp. 47–62, 2004.
- [24] T. Friedrich, P. S. Oliveto, D. Sudholt, and C. Witt, "Analysis of diversity-preserving mechanisms for global exploration," *Evolutionary Computation*, vol. 17, no. 4, pp. 455–476, 2009.
- [25] J. C. Bean, "Genetic algorithms and random keys for sequencing and optimization," *ORSA Journal on Computing*, vol. 6, pp. 154–160, 1994.
- [26] J. C. Potts, T. D. Giddens, and S. B. Yadav, "Development and evaluation of an improved genetic algorithm based on migration and artificial selection," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 24, no. 1, pp. 73–86, 1994.
- [27] P. W. Poon and J. N. Carter, "Genetic algorithm crossover operators for ordering applications," *Computers and Operations Research*, vol. 22, no. 1, pp. 135–147, 1995.
- [28] H.-F. Wang and K.-Y. Wu, "Hybrid genetic algorithm for optimization problems with permutation property," *Computers and Operations Research*, vol. 31, no. 14, pp. 2453–2471, 2004.
- [29] T. Murata, H. Ishibuchi, and H. Tanaka, "Genetic algorithms for flowshop scheduling problems," *Computers and Industrial Engineering*, vol. 30, no. 4, pp. 1061–1071, 1996.
- [30] R. Ruiz and T. Stützle, "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem," *European Journal of Operational Research*, vol. 177, no. 3, pp. 2033–2049, 2007.
- [31] S. F. Galan and O. J. Mengshoel, "A novel mating approach for genetic algorithm," *Evolutionary Computation*, vol. 21, no. 2, pp. 197–229, 2013.