Int J Adv Manuf Technol (2005) 25: 1157–1163
DOI 10.1007/s00170-003-1961-y

ORIGINAL ARTICLE

L. Wang · L. Zhang · D.-Z. Zheng

A class of hypothesis-test-based genetic algorithms for flow shop scheduling with stochastic processing time

Received: 10 June 2003 / Accepted: 15 September 2003 / Published online: 12 January 2005

Abstract As an important optimisation problem with a strong engineering background, stochastic flow shop scheduling with uncertain processing time is difficult because of inaccurate objective estimation, huge search space, and multiple local minima, especially NP-hardness. As an effective meta-heuristic, genetic algorithms (GAs) have been widely studied and applied in scheduling fields, but so far seldom for stochastic cases. In this paper, a hypothesis-test method, an effective methodology in statistics, is employed and incorporated into a GA to solve the stochastic flow shop scheduling problem and to avoid premature convergence of the GA. The proposed approach is based on statistical performance and a hypothesis test. It not only preserves the global search ability of a GA, but it can also reduce repeated searches for those solutions with similar performance in a statistical sense so as to enhance population diversity and achieve better results. Simulation results based on some benchmarks demonstrate the feasibility and effectiveness of the proposed method by comparison with traditional GAs. The effects of some parameters on the performance of the proposed algorithms are also discussed.

Keywords Genetic algorithm · Hypothesis test · Flow shop scheduling · Stochastic processing time

Nomenclature

n	Number of jobs
m	Number of machines
p_{ij}	Processing time of job i on machine j
P_{ij}	Expected processing time
$C_{\max}, C^*$	Makespan, optimal makespan value or lower bound value
P_s	Population size
$P(k)$	Population at kth generation
$P^t(k)$	Temporary population at kth generation
$\theta, \theta*$	Solution, and the best solution found so far
p_m	Mutation probability
p_c	Crossover probability
J	Performance expectation
L	Sample performance
μ	Theoretical mean
σ^2	Theoretical variance
$\bar{J}_i, s_i^2$	Estimated performance and sample variance of ith individual
$\bar{J}^*, s_*^2$	Estimated performance and sample variance of best solution
H_0, H_1	Null hypothesis, and alternative hypothesis
α	Evidence level
τ	Absolute bound of critical region
N, n_i	Simulation times
k, l, j	Index number
ξ	Noise
η	Noise magnitude
BEM	The best expected makespan calculated with expected processing time
AEM	The average expected makespan calculated with expected processing time
WEM	The worst expected makespan calculated with expected processing time

L. Wang (✉) · L. Zhang · D.-Z. Zheng
Department of Automation, CFINS,
Tsinghua University,
Beijing 100084, P.R. China
E-mail: wangling@mail.tsinghua.edu.cn
Tel.: 86-10-62783125272
Fax: 86-10-62786911

1 Introduction

Because of the complexity and Non-deterministic Polynomial (NP)-hardness of many real engineering scheduling problems and their key role in manufacturing systems, it is very important to develop efficient and effective advanced manufacturing and scheduling technologies and approaches. Among them, flow shop scheduling is one of the most well-known and well-studied production scheduling problems with a strong engineering background [1, 2]. The permutation flow shop problem with n jobs and m machines can be defined as follows. Each of the n jobs is to be sequentially processed on machine 1 through to m. At any time, no job can be processed on more than one machine, while no machine can process more than one job simultaneously. Moreover, the permutation of the n jobs to be processed on each machine is the same. The objective widely used is to minimise the completion time of the last job, i.e. makespan $C_{\max}$ [1, 2]. It has been proven that the above scheduling problem under the criterion of $C_{\max}$ is strongly NP-hard when $m \geq 3$ [1].

Up to now, various methods including constructive methods [2–4] and meta-heuristics [5–11] have been proposed for solving scheduling problems. But in many real scheduling problems, uncertainty is so prevalent that it is more important and practicable to study stochastic scheduling problems than deterministic ones. In this paper, flow shop scheduling with a stochastic processing time is considered, where the processing time $p_{i,j}$ of job i on machine j is assumed to be random. In such a case, the expected makespan is often used to evaluate the performance of the solutions [12]. Kamburowski [13] presented sufficient conditions for job processing time distributions that stochastically minimise the makespan of three-machine flow shop problems with unlimited intermediate storage. De et al. [14] presented solution algorithms for the single-machine flow-time problem with the variance as a performance measure in addition to expectation. Honkomp et al. [15] presented an approach to incorporate schedules into a simulator to validate the schedule and test rescheduling methodologies when stochastic events occur. Mathematical programming [16–18], constructive approaches [19], and fuzzy method [20] have also been studied for scheduling problems with stochastic or fuzzy processing times.

Borrowing the notion of "survival of the fittest", genetic algorithms (GAs) [21, 22] are one of the evolutionary computation algorithms with learning capability and have gained wide applications in a variety of fields, but mostly for deterministic optimisation problems. In addition, it is often found that a GA is very prone to converge prematurely. From the statistical viewpoint, a class of hypothesis-test-based genetic algorithms (HTGA) is proposed for stochastic flow shop scheduling in this paper, which applies a hypothesis test based on a mean value comparison to enhance population diversity so as to avoid premature convergence and improve the effectiveness of search space exploration via genetic operators. Simulation results based on some benchmarks demonstrate the feasibility and effectiveness of the HTGA by comparison with traditional GA. In addition, the effects of some parameters on the optimisation performances are also discussed.

The organisation of the remaining contents is as follows. In Sect. 2, the HTGA is proposed after briefly reviewing the hypothesis test and GAs. The implementation of the HTGA is described in Sect. 3 for flow shop scheduling with random processing times. Computational simulation and the effects of some parameters on optimisation performance are given in Sect. 4, and some conclusions follow in Sect. 5.

2 Hypothesis-test-based genetic algorithm

2.1 Hypothesis test

Generally speaking, a stochastic optimisation problem can be described as follows [23].

$$\min_{\theta} J(\theta) = E[L(\theta, \xi)] \,, \tag{1}$$

where θ is a feasible solution of the problem in a finite set and J is the expectation of L, the sample performance as a function of θ and ξ (noise or uncertain factors).

A hypothesis test is an important statistical method that is used to test for a predefined hypothesis based on experimental data [24]. Performing a hypothesis test for different solutions when optimising stochastic problems often needs multiple independent simulations to provide a suitable performance estimation for decision solutions. If n_i independent simulations are carried out for solution θ_i, then its unbiased estimated mean value $\bar{J}_i$ and variance s_i^2 can be calculated as follows.

$$\bar{J}_i = \bar{J}(\theta_i) = \sum_{j=1}^{n_i} \frac{L(\theta_i, \xi)}{n_i} \tag{2}$$

$$s_i^2 = \sum_{j=1}^{n_i} \frac{[L(\theta_i, \xi) - \bar{J}_i]^2}{n_i - 1} \tag{3}$$

Consider two different solutions θ_1 and θ_2, whose estimated performances $\hat{J}(\theta_1)$ and $\hat{J}(\theta_2)$ are two independent random variables. According to the law of large numbers and the central limit theorem [24], the estimation $\hat{J}(\theta_i)$ is subject to $N\left(\bar{J}_i, \frac{s_i^2}{n_i}\right)$ when n_i approaches ∞. Suppose $\hat{J}(\theta_1) \sim N\left(\mu_1, \sigma_1^2\right)$ and $\hat{J}(\theta_2) \sim N\left(\mu_2, \sigma_2^2\right)$, and the unbiased estimation values of μ_1, μ_2, and s_1^2, s_2^2 are given by Eqs. 2 and 3, respectively, then let the null hypothesis H_0 be "$\mu_1 = \mu_2$" and the alternative hypothesis H_1 be "$\mu_1 \neq \mu_2$".

If σ_1^2 and σ_2^2 are known, then the critical region of H_0 is as follows [24].

$$\left|\bar{J}_1 - \bar{J}_2\right| \geq z_{\alpha/2}\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}} = \tau \,, \tag{4}$$

where α is the evidence level with the meaning that $\phi(z_{\alpha/2}) = 1 - \alpha/2$.

If σ_1^2 and σ_2^2 are unknown and n_1, n_2 are large enough (say 50), then the critical region of H_0 can be simplified as follows [24].

$$\left|\bar{J}_1 - \bar{J}_2\right| \geq z_{\alpha/2}\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}} = \tau \,, \tag{5}$$

where $s_1^2 = \sum_{j=1}^{n_1} \frac{[L(\theta_1,\xi) - \bar{J}_1]^2}{n_1 - 1}$ and $s_2^2 = \sum_{j=1}^{n_2} \frac{[L(\theta_2,\xi) - \bar{J}_2]^2}{n_2 - 1}$ are the unbiased estimation of σ_1^2 and σ_2^2, respectively.

If $\sigma_1^2 = \sigma_2^2 = \sigma^2$ and σ^2 is unknown, then the critical region of H_0 is described as follows [24].

$$\left|\bar{J}_1 - \bar{J}_2\right| \geq t_{\alpha/2}(n_1 + n_2 - 2) \cdot Y_1 Y_2 = \tau \ , \tag{6}$$

where $Y_1 = \sqrt{\frac{(n_1-1)s_1^2+(n_2-1)s_2^2}{n_1+n_2-2}}$, $Y_2 = \sqrt{\frac{n_1+n_2}{n_1 n_2}}$.

Thus, if $|\bar{J}(\theta_1) - \bar{J}(\theta_2)| < \tau$, i.e. the null hypothesis holds, then it can be regarded that the performances of these two solutions are not significantly different in a statistical sense; otherwise their performances are significantly different. Furthermore, for the stochastic minimisation problem, θ_2 is better than θ_1 if $\bar{J}(\theta_1) - \bar{J}(\theta_2) \geq \tau$, and θ_1 is better than θ_2 if $\bar{J}(\theta_1) - \bar{J}(\theta_2) \leq -\tau$. In addition, for a specific problem it is often supposed that the theoretical performance variances of all solutions are the same [24], so the hypothesis test can be made according to Eq. 6. For multi-modal stochastic optimisation problems, a comparison under pure hypothesis test can often be trapped into local optima. This motivates us to combine the hypothesis test with the effective search ability of a GA to solve the problem.

2.2 Genetic algorithm

The first genetic algorithm was proposed by Holland [21]; it borrows the ideas of natural selection and survival of the fittest. GA is naturally parallel and exhibits implicit parallelism [21, 22], which does not evaluate and improve a single solution but analyses and modifies a set of solutions simultaneously. Even if random initialisation is used, a selection operator can select some "good" solutions as seed ones and the crossover operator can generate new solutions, hopefully retaining the good features from the parents, and the mutation operator can enhance the diversity and provide the opportunity to escape from local optima. A GA is an iterative learning process with a certain learning ability and thus it is regarded as a type of computational intelligence for optimisation. Although many weaknesses still exist, such as premature convergence, parameter dependence, and hardness for determining stopping criterion, GAs have been intensively studied and applied in many fields, especially in the production scheduling field. Not much use of it, however, has been made for stochastic cases.

2.3 Hypothesis-test-based genetic algorithm

In this section, we will incorporate a hypothesis test into a GA for stochastic optimisation problems.

After generating all new solutions by genetic operators, the solutions will be ordered based on their estimated mean performances, from the best to the worst, and the first solution will be put into the next population. Then, one by one from the second solution to the last one, the current solution is compared with the nearest former solution that has not been discarded. Based on the idea of a hypothesis test, if there is no significant difference between their performances (say null hypothesis holds), then the current solution will be discarded to avoid a repeated search; otherwise, the solution is retained and put into the next population. After finishing such a hypothesis-test-based comparison process, all the discarded solutions will be replaced by new solutions, randomly generated, and the new ones will be put into the next population to enhance the population diversity to some extent.

Thus, a class of hypothesis-test-based genetic algorithms (HTGA) for stochastic optimisation problems is proposed as follows:

Step 1. Randomly generate P_s solutions to form the initial population, $P(0)$. Estimate the performance $\bar{J}_i$ and variance s_i^2 of each solution with multiple independent simulations. Let $k = 0$.

Step 2. Let the best solution of $P(k)$ be θ^* with the estimated performance $\bar{J}^*$ and variance s_*^2. If the stopping criterion is satisfied, then output the best solution and its performances; otherwise, sort the individuals of $P(k)$ in ascending order, i.e. $\bar{J}_1 \leq \bar{J}_2 \leq \ldots \leq \bar{J}_{P_s}$ and continue the following steps.

Step 3. Repeat genetic operators (including selection, crossover, and mutation) $\frac{P_s}{2}$ times for $P(k)$ to form a temporary population $P^t(k)$, and estimate the performance $\bar{J}_i^t$ and variance $s_i^{t^2}$ of every new solution by multiple independent simulations.

Step 4. Order all the solutions of $P^t(k)$ by $\bar{J}_1^t \leq \bar{J}_2^t \leq \ldots \leq \bar{J}_{P_s}^t$, and denote the resulted solutions $\theta_1^t, \theta_2^t, \ldots, \theta_{P_s}^t$, respectively. Let $l = 1$, $j = 2$, and put θ_1^t into $P(k+1)$ and denote it as θ_l.

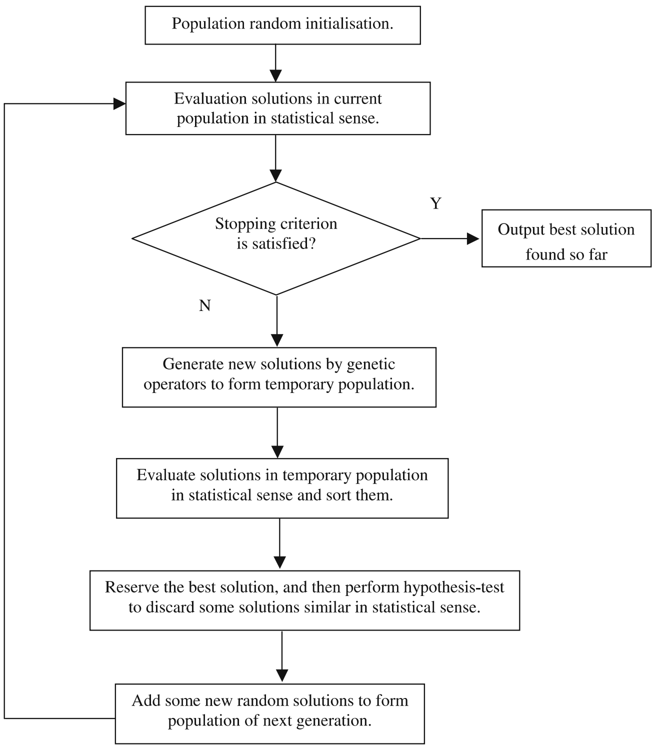
Population random initialisation.
Evaluation solutions in current population in statistical sense.
Stopping criterion is satisfied?
Y
N
Output best solution found so far
Generate new solutions by genetic operators to form temporary population.
Evaluate solutions in temporary population in statistical sense and sort them.
Reserve the best solution, and then perform hypothesis-test to discard some solutions similar in statistical sense.
Add some new random solutions to form population of next generation.

Fig. 1. Framework of the HTGA

Step 5. Perform the hypothesis test for θ_j^t with θ_l, which is in $P(k+1)$. If the null hypothesis holds, i.e. Eq. 6 does not hold, then θ_j^t is discarded from $P^t(k)$; otherwise, θ_j^t is put into $P(k+1)$ and denoted as θ_{l+1}, and let $l = l+1$.

Step 6. If $j < P_s$, then let $j = j+1$ and go to step 5; otherwise, randomly generate $p_s - l$ new solutions and put them into $P(k+1)$, and replace the worst solution of $P(k+1)$ by the best solution of $P(k)$, let $k = k+1$, then go to step 2.

It can be seen from above procedure that the algorithm firstly inherits the fundamental framework and operators of GAs to keep the generality and effective optimisation ability. Secondly, for stochastic optimisation problems a hypothesis test based on statistical performance can enhance the population diversity, avoid the repeated search, and deal reasonably with the random factor to some extent. The elitist strategy in step 6 guarantees the reservation of the best solution found so far. For the sake of clarity, the framework of the HTGA is briefly illustrated in Fig. 1.

3 Implementation of HTGA for stochastic flow shop scheduling

An encoding scheme based on job permutation is widely used in many papers for permutation flow shop scheduling [11, 22], so such an encoding scheme is also adopted in this paper. Next the implementation of the HTGA for permutation stochastic flow shop scheduling will be discussed.

3.1 Implementation of step 1

Since no prior knowledge is available, random sampling is used to generate the initial population. Of course some problem-dependent information or heuristic methods can also be applied, e.g. the NEH heuristic [3]. To guarantee a certain estimation accuracy, N independent simulation replications are performed for each individual of the initial population to estimate its performance $\bar{J}_i$ $(i = 1, 2, \ldots P_s)$ and the corresponding sample variance s_i^2.

3.2 Implementation of step 2

A maximum evolution generation $G_{\max}$ is often set as the stopping criterion of the algorithm, i.e. once $G_{\max}$ generations is reached the best solution found so far θ^* will be output. In addition, since order is easier to determine and more robust than value for stochastic optimisation problems [23], all the solutions of the current population will be ordered in ascending order and an order-based selection [25] will then be used.

3.3 Implementation of step 3

In this step, selection based on the order of each individual instead of the exact value of performance is used, so that the transition from objective value to fitness value in the traditional value-based proportional selection operator can be avoided and it is not necessary to estimate the exact performance [23, 25]. Here, the ith ordered solution of the current population is assigned a probability $\frac{2(P_s-i+1)}{P_s(P_s+1)}$ of being selected. Then a crossover operation is performed between the selected solutions.

The widely used linear order crossover (LOX) [26] is applied here, which is reported to have a good capability for solving permutation flow shop problems. It was considered that LOX could preserve as much as possible the relative positions between the genes and the absolute positions relative to the extremities of the chromosome [26]. The LOX step can be explained briefly with an example as follows. Two cutting sites of the parents, e.g. (2 6 4 7 3 5 8 9 1) and (4 5 2 1 8 7 6 9 3), are chosen randomly, e.g. 2 and 5. Secondly, the symbols that appear in the cross section of the first parent (the area situated between the two cutting sites) are removed from the second parent leaving some “holes”, i.e. (H 5| 2 1 8| H 6 9 H) and (H 6| 4 7 3 |5 H 9 H). Then the holes are slid from the extremities towards the centre until they reach the cross section, i.e. (5 2| H H H| 1 8 6 9) and (6 4| H H H|7 3 5 9). Finally the cross section is substituted with that of the corresponding parent to obtain the children, i.e. (5 2| 4 7 3| 1 8 6 9) and (6 4| 2 1 8| 7 3 5 9). After performing such an order-based selection and crossover $P_s/2$ times, P_s new solutions will be generated, which will perform the next mutation operation.

As we know, crossover can explore the search space by combining the individuals in the current population, and mutation can maintain population diversity to some extent. With a probability p_m, the SWAP mutation [26, 27] is applied for all solutions generated by crossover, i.e. two distinct elements are randomly selected and swapped. Then, the resulted P_s individuals form the temporary population $P^t(k)$.

3.4 Implementation of step 4

To preserve good solutions and to perform an hypothesis-test-based comparison, all individuals of $P^t(k)$ will firstly be sorted in ascending order according to their estimated performance through N independent simulation replications for each solution.

3.5 Implementation of step 5

In this step, from the second individual of $P^t(k)$ to the last one, a hypothesis-test-based comparison will be performed between each individual and the nearest former one that has not been discarded. If the null hypothesis holds, then the two individuals will be considered similar in a statistical sense and the current individual will be discarded from $P^t(k)$ to maintain population diversity; otherwise, the current individual will be put into $P(k+1)$, the two individuals are considered statistically different.

3.6 Implementation of step 6

After the hypothesis-test-based comparison, all discarded solutions will be replaced by new solutions generated randomly to form the entire population $P(k+1)$. An elitism strategy is applied to reserve the best solution found so far, i.e. the worst solution of $P(k+1)$ will be replaced by the best solution of $P(k)$ if necessary. Then the genetic search with exploration and exploitation abilities will continue by going back to step 2 till the stopping criterion is satisfied.

4 Numerical test

Computational simulation is often carried out with some benchmarks. In this paper, eight benchmarks named car1–car8 by Carlier [28] and 21 benchmarks named rec01, rec03, ... , rec41 by Reeves [7] are used, which have been used widely and most of them have been found to be particularly difficult to solve [11]. Meanwhile, $p_{i,j}$ is supposed to be subjected to a uniform distribution $U((1-\eta)P_{i,j}, (1+\eta)P_{i,j})$, where $P_{i,j}$ is the expected processing time provided by benchmarks, and η denotes noise magnitude.

4.1 Simulation results of HTGA and SGA

Setting $P_s = 40$, $p_c = 1.0$, $p_m = 0.1$, and $\eta = 5\%$, we respectively carry out 20 independent simulations for the HTGA and the simple genetic algorithm [22, 26] (namely SGA-20) both with 20 simulation replications for estimation (i.e. $N = 20$). The statistical results are shown in Table 1. For those solutions obtained for stochastic scheduling problems, it is more meaningful to show their expected makespan values than sample or estimated performances [12]; therefore, only the best, average, and worst expected makespan (denoted by BEM, AEM, and WEM, respectively), calculated with the expected processing time for those solutions obtained by the algorithm with estimated performances, are illustrated.

From Table 1, it can be concluded that the HTGA can achieve much better solutions than the SGA for stochastic flow shop scheduling problems. Secondly, the results obtained by the HTGA are very close to the true optimum of the problems (in expectation sense). Thirdly, the average performances of the HTGA are also better than that of the SGA, especially for the large-scale problems. It can also be concluded that the HTGA is more robust for initial solutions than the SGA because the BEM, AEM, and WEM values are much closer than those of the SGA. The reason is that in the process of the SGA, it is very easy to become trapped in local minima, resulting in premature convergence due to the loss of population diversity and the uncertainty of performance estimation. The HTGA can maintain the diversity by discarding the individuals similar in a statistical sense, so as to prolong order-based genetic evolution.

4.2 Effect of noise magnitude on HTGA

For stochastic optimisation problems, as the magnitude of the uncertainty increases, a larger noise will be added in performance

Table 1. Comparisons between SGA and HTGA

Problem	n, m	C^*	HTGA			SGA-20		
			BEM	AEM	WEM	BEM	AEM	WEM
Car1	11, 5	7038	7038	7038.0	7038	7038	7038.0	7038
Car2	13, 4	7166	7166	7176.5	7376	7166	7208.0	7376
Car3	12, 5	7312	7312	7331.6	7400	7312	7378.7	7422
Car4	14, 4	8003	8003	8003.0	8003	8003	8009.3	8129
Car5	10, 6	7720	7720	7728.8	7821	7720	7773.2	7821
Car6	8, 9	8505	8505	8511.5	8570	8505	8544.0	8570
Car7	7, 7	6590	6590	6590.0	6590	6590	6611.4	6753
Car8	8, 8	8366	8366	8366.0	8366	8366	8377.1	8530
Rec01	20, 5	1247	1247	1256.5	1308	1249	1269.5	1326
Rec03	20, 5	1109	1109	1113.8	1122	1111	1116.4	1130
Rec05	20, 5	1242	1242	1251.9	1269	1245	1253.0	1273
Rec07	20,10	1566	1566	1585.9	1600	1584	1594.4	1619
Rec09	20,10	1537	1537	1564.6	1588	1538	1574.7	1608
Rec11	20,10	1431	1431	1462.9	1498	1445	1488.6	1536
Rec13	20,15	1930	1930	1974.4	2011	1962	1993.5	2022
Rec15	20,15	1950	1950	1988.6	2033	1979	2011.7	2049
Rec17	20,15	1902	1919	1974.0	2014	1924	1977.6	2019
Rec19	30,10	2093	2131	2169.3	2205	2137	2172.1	2218
Rec21	30,10	2017	2050	2077.6	2126	2050	2089.4	2141
Rec23	30,10	2011	2058	2082.1	2133	2069	2096.9	2140
Rec25	30,15	2513	2568	2620.8	2659	2595	2639.7	2671
Rec27	30,15	2373	2423	2465.1	2494	2423	2483.7	2531
Rec29	30,15	2287	2357	2413.9	2453	2380	2428.3	2507
Rec31	50,10	3045	3129	3176.4	3215	3131	3184.7	3245
Rec33	50,10	3114	3140	3173.8	3225	3140	3179.9	3233
Rec35	50,10	3277	3277	3302.1	3347	3284	3319.4	3370
Rec37	75,20	4951	5211	5281.3	5352	5211	5290.8	5368
Rec39	75,20	5087	5252	5373.8	5440	5298	5387.3	5486
Rec41	75,20	4960	5227	5308.1	5378	5227	5320.0	5414

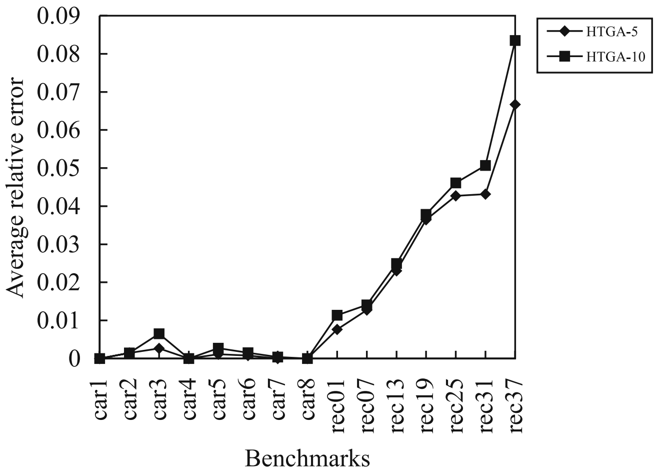

Fig. 2. Effect of uncertainty magnitude on HTGA

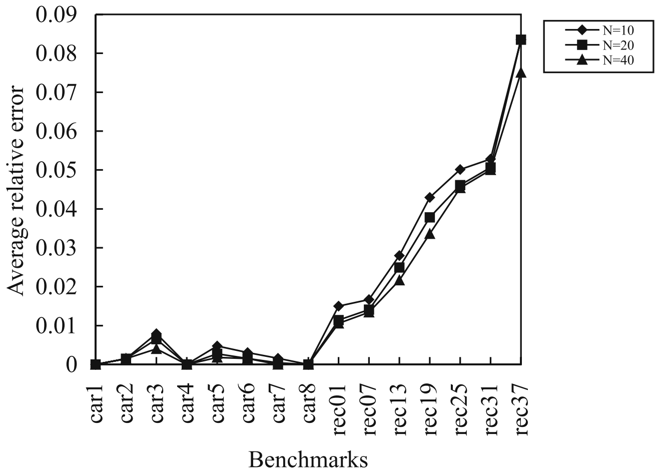

Fig. 3. Effect of estimation replication on HTGA

estimation so that it is harder to estimate the performance accurately, and consequently it would be more difficult to solve the problems. With the same parameters as the above except η, we carry out 20 independent simulations for the HTGA with $\eta = 5\%$ and $\eta = 10\%$. The effect of noise magnitude (i.e. η) on the relative error of AEM with respect to the theoretically optimal value, i.e. $\frac{\text{AEM}-C^*}{C^*}$, is shown in Fig. 2, where HTGA-n denotes the HTGA with $\eta = n\%$.

From Fig. 2, it is shown that as the magnitude of the uncertain noise increases, the optimisation problems become more difficult to solve. That is, the larger the uncertainty magnitude is, the poorer the resulting performance (AEM values or the relative error with respect to the theoretically optimal value is shown in Fig. 2). The reason is that the increase in the uncertainty magnitude may cause a larger variance and a more inaccurate estimation so that more solutions will be discarded by applying the hypothesis test during the evolutionary search of the HTGA. Consequently, worse optimisation quality may result when the maximum evolve generation is fixed. Next it will be shown that the optimisation quality, when the noise magnitude increases, will be improved by using more simulation replications.

4.3 Effect of simulation replication on HTGA

For stochastic optimisation problems, the objective value can usually be estimated only through certain numbers of independent random experiments. The accuracy of estimation of objective value plays an important role in optimisation. A widely used method to improve estimation accuracy is to increase the number of simulation replications. Setting $\eta = 10\%$ and other parameters the same as the above, we carry out 20 independent simulations for the HTGA with 10, 20, and 40 replications for estimation (denoted by $N = 10$, $N = 20$, and $N = 40$). The effect of simulation replication (i.e. N) on the relative error of AEM with respect to the theoretically optimal value is shown in Fig. 3.

From Fig. 3, it is shown that with multiple-replication estimation the HTGA can achieve better results (AEM or its relative error with respect to the theoretically optimal value is decreased). This is because through multiple replications the performance of each individual can be estimated more precisely so that more genetic searching will be made for “real” good solutions and the global optima can be obtained more accurately. Together with the results of Sect. 4.2, although noise magnitude increases, a better performance can be achieved by using more simulation replications for estimation. However, as we know, more simulation replications means more computing effort. It is found that the quality of improvement from $N = 20$ to $N = 40$ is not larger than that from $N = 10$ to $N = 20$, but the increased simulation replications from $N = 20$ to $N = 40$ are much larger than that from $N = 10$ to $N = 20$. So, considering both the optimisation quality and the optimisation time performance, we recommend performing 20–30 replications for the performance estimation of each solution.

5 Conclusion

In this paper, a class of hypothesis-test-based genetic algorithms has been proposed for flow shop scheduling problems with stochastic processing times, which not only applies the evolutionary searching mechanism of a GA to perform exploration and exploitation efficiently, but is also combined with a hypothesis test to perform a statistical comparison to maintain population diversity. Using statistical comparisons and using the hypothesis test to maintain the population diversity, much repeated searching can be avoided and performance estimation with multiple replications can lead to a better performance of the algorithm for stochastic optimisation problems. This has been demonstrated by benchmark-based computational simulation. The effects of some parameters on the optimisation quality have been discussed. Future work will theoretically study the convergence behaviour of the HTGA, as well as develop some adaptive mechanisms. Due to the generality and easy implementation of the HTGA, other applications will be attempted.

Acknowledgement This research was partially supported by the National Science Foundation of China under the Grant 60204008 and 60374060, the National 973 Program under the Grant 2002CB312200, and the Basic Research Foundation from Tsinghua University.

References

1. Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. Freeman, San Francisco
2. Baker KR (1974) Introduction to sequencing and scheduling. Wiley, New York
3. Nawaz M, Enscore E Jr, Ham I (1983) A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega 11(1):91–95
4. Koulamas C (1998) A new constructive heuristic for the flowshop scheduling problem. Eur J Oper Res 105:66–71
5. Ogbu FA, Smith DK (1990) The application of the simulated annealing algorithm to the solution of the $n/m/C_{\max}$ flowshop problem. Comput Oper Res 17(3):243–253
6. Osman IH, Potts CN (1989) Simulated annealing for permutation flowshop scheduling. Omega 17(6):551–557
7. Reeves CR (1995) A genetic algorithm for flowshop sequencing. Comput Oper Res 22(1):5–13
8. Wang L, Zheng DZ (2003) A modified evolutionary programming for flow shop scheduling. Int J Adv Manuf Technol 22(7–8):522–527
9. Nowicki E, Smutnicki C (1996) A fast tabu search algorithm for the permutation flow-shop problem. Eur J Oper Res 91:160–175
10. Grabowski J, Pempera J (2001) New block properties for the permutation flowshop problem with application in tabu search. J Oper Res Soc 52:210–220
11. Dimopoulos C, Zalzala AMS (2000) Recent development in evolutionary computation for manufacturing optimization: problems, solutions, and comparisons. IEEE Trans Evol Comput 4(2):93–113
12. Pinedo M (1995) Scheduling theory, algorithms, and systems. Prentice-Hall, Englewood Cliffs, NJ
13. Kamburowski J (2000) On three-machine flow shops with random job processing times. Eur J Oper Res 125:440–449
14. De P, Ghosh JB, Wells CE (1992) Expectation-variance analysis of job sequences under processing time uncertainty. Int J Prod Econ 28: 289–297
15. Honkomp SJ, Mockus L, Reklaitis GV (1999) A framework for schedule evaluation with processing uncertainty. Comput Chem Eng 23(4–5):595–609
16. Balasubramanian J, Grossmann IE (2002) A novel branch and bound algorithm for scheduling flowshop plants with uncertain processing times. Comput Chem Eng 26(1):41–57
17. Gutjahr WJ, Hellmayr A, Pflug GC (1999) Optimal stochastic single-machine-tardiness scheduling by stochastic branch-and-bound. Eur J Oper Res 117(2):396–413
18. Luh PB, Chen D, Thakur LS (1999) An effective approach for job-shop scheduling with uncertain processing requirements. IEEE Trans Robot Autom 15(2):328–339
19. Kouvelis P, Daniels RL, Vairaktarakis G (2000) Robust scheduling of a two-machine flow shop with uncertain processing times. IIE Trans 32(5):421–432
20. Tsujimura Y, Park SH, Chang IS, Gen M (1993) An effective method for solving flow shop scheduling problems with fuzzy processing times. Comput Ind Eng 25:239–242
21. Holland JH (1975) Adaptive in natural and artificial systems. University of Michigan Press, Ann Harbor
22. Goldberg DE (1989) Genetic algorithms in search, optimization and machine learning. Addison-Wesley, Reading, MA
23. Ho YC, Cassandras CC, Chen CH, Dai L Ordinal optimization and simulation. J Oper Res Soc 51(4):490–500
24. Liu X (1998) Introduction to statistics. Tsinghua University Press, Beijing
25. Wang L, Zhang L, Zheng DZ (2003) A class of order-based genetic algorithm for flow shop scheduling. Int J Adv Manuf Technol 22(11–12):828–835
26. Wang L, Zheng DZ (2003) An effective hybrid heuristic for flow shop scheduling. Int J Adv Manuf Technol 21(1):38–44
27. Wang L, Zheng DZ (2001) An effective hybrid optimization strategy for job shop scheduling problems. Comput Oper Res 28(6):585–596
28. Carlier J (1978) Ordonnancements a contraintes disjonctives. R.A.I.R.O. Oper Res 12:333–351