Int J Adv Manuf Technol (2007) 32: 780–786
DOI 10.1007/s00170-005-0392-3

ORIGINAL ARTICLE

K. S. Amirthagadeswaran · V. P. Arunachalam

Enhancement of performance of Genetic Algorithm for job shop scheduling problems through inversion operator

Received: 4 December 2004 / Accepted: 29 September 2005 / Published online: 30 March 2006

Abstract It is observed that the optimization technique Genetic Algorithm is gaining more importance over the past several years. With high computing power we are able to apply soft computing techniques to solve complex problems in less time. An approach through Genetic Algorithm to solve job shop scheduling problems using inversion operator has been tried, with make-span objective. Computational experiments of this attempt have shown better solutions coupled with appreciable reduction in computer processing time. A set of 20 selected benchmark problems were tried with the proposed heuristic for validation and the results are encouraging. The inversion operator is found to perform better.

Keywords Optimization · Job shop scheduling · Genetic algorithm · Inversion operator

1 Introduction

Job shop problems have a set of *n* jobs to be processed on a set of *m* machines. Job shop scheduling deals with the allocation of jobs to various machines with the objective of minimizing the total production time or make-span (the time to complete all jobs), or minimizing the tardiness (not meeting the due date) in jobs or any other required objectives. Each job is composed of a set of operations and operation order on various machines. Each operation is characterized by the machine required and the processing time required. Job shop problems occur mainly in industries where each customer has specified characters and order sizes are relatively small.

The following assumptions are made while solving the job shop scheduling problems [1]

1. Each machine is an entity
2. No pre-emption is allowed
3. Each job has m distinct operations, one on each machine
4. No cancellation of jobs
5. The processing times are independent of the schedule
6. In-process inventory is allowed
7. There is only one of each type of machine
8. Machines may be idle
9. No machine may process more than one operation at a time
10. Machines never breakdown and are available throughout the scheduling period
11. The technological constraints are known in advance and are immutable
12. There is no randomness; all the data are known and fixed

There are several constraints on jobs and machines [2]. The constraints are,

(1) a job does not visit the same machines twice
(2) there are no precedence constraints among operations of different jobs
(3) operation can not be interrupted
(4) each machine can process only one job at a time
(5) neither release times nor due dates are specified

In general, an infinite number of feasible schedules is possible for any job shop problem, as one can insert any arbitrary amount of idle time at any machine between adjacent pairs of operations [3]. Job shop scheduling problem (JSP) is well known as one of the most difficult NP-hard combinatorial problems. The JSP is harder than the traveling salesman problem (TSP).

The job shop scheduling problems have been tried with simulated annealing, neural networks, Tabu search and Genetic Algorithms (GA) for the past two decades. Applications of GA for job shop scheduling problems have been reported by researchers Yamada and Nakano [4, 5]. Specific works on the use of GA for job shop scheduling problems with Operation-based representation have been presented by Fang et al. [6] and Gen et al. [7]. Use of inversion operator for traveling salesman problems (TSP) has been discussed

K. S. Amirthagadeswaran (✉) · V. P. Arunachalam
Mechanical Engineering, Government College of Technology, Coimbatore, India
e-mail: ksagp@yahoo.co.in

by Ferreira [8] and Shubra Sankar Ray et al. [9]. TSP serves as the simple case of a variety of combinatorial problems, which are of enormous relevance to the industrial scheduling problems.

Genetic algorithms (GA), developed by John Holland, are search algorithms based on the mechanics of natural selection and natural genetics. A GA comprises a set of individual elements (known as the population) and a set of biologically inspired operators defined over the population itself. According to evolutionary theories, only the most suited elements in a population are likely to survive and generate offspring, thus transmitting their biological heredity to new generations. A GA operates through a simple cycle of stages:

(1) creation of a 'population' of strings
(2) evaluation of each string
(3) selection of 'best' strings, and
(4) Genetic manipulation (crossover and mutation) to create new population.

2 Solution methodology

2.1 Objective function

The objective of the problem is to find the job sequence of operations for which the make-span Cmax is minimum.

$$\textit{Objective function} = f(x)$$

2.2 Fitness function

Since genetic agorithms are most suitable for maximization problems, the above minimization problem is converted into an equivalent maximization problem by the following transformation.

$$\textit{Fitness function value } F(x) = C_{maxp} - f(x)$$

where C_{maxp} is taken as the largest make-span value observed in the current population. So, Cmaxp varies dynamically with varying population.

2.3 Input

The input data are: Job numbers, operation sequences, and operation time.

2.4 Initialization

This module deals with the creation of all possible chromosomes, within the population size. The operation of GA begins with a population of random strings representing decision variables. Since job numbers are involved in this problem, and operation based representation is followed, coding is done by integer coding method. As it is observed from literature that minimum make-span is found in operation based representation, this representation has been tried. Operation based representation has been adopted by Fang et al. [6] and Gen et al. [7] in solving Job shop problems using GA. In this method, strings (chromosomes) are coded as a sequence of numbers (genes) with each gene representing one of the operations of the jobs involved. The specific operations represented by the genes are interpreted according to the order of presence of the genes in the chromosome. Each of the *n* (jobs) differently named (coded) genes will appear *m* (machines) times spread over the entire chromosome.

Let us consider a three jobs four machines problem.

Processing time matrix of the problem:

		Operations			
		1	2	3	4
Jobs	1)	10	8	4	3
	2)	8	3	5	6
	3)	4	7	3	3

Machine sequence matrix of the problem:

		Operations			
		1	2	3	4
Jobs	1)	1	2	3	4
	2)	2	1	4	3
	3)	1	2	4	3

For the above problem a coded string comprising 3*4 numbers is generated randomly. The string say, 3 2 3 1 1 3 2 2 3 1 1 2 is a typical chromosome, where 1 stands for job-1, 2 for job-2 and 3 for job-3. Because each job has four operations, each job occurs exactly four times in the chromosome. There are four 3 s in the chromosome, representing the four operations of job-3. The first 3 corresponds to the first operation of job-3, which will be processed on machine 1, the second 3 corresponds to the second operation of job-3 which will be processed on machine 2, the third 3 corresponds to the third operation of job-3 which will be processed on machine 4, and the fourth 3 corresponds to the fourth operation of job which will be processed on machine 3. Ten random strings are created to form the initial population.

2.5 Evaluation

The objective criterion for the problems has been chosen as the make-span. The population of chromosomes is evaluated for this objective. Population size (p-size) is kept constant throughout the trial and the fitness value for each string generated is calculated. This value is used for reproduction operation. For the pattern of chromosome, 3 2 3 1 1 3 2 2 3 1 1 2, the make-span is found by placing the

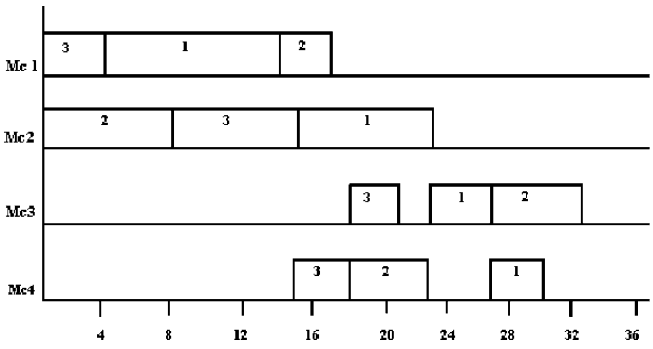

Fig. 1 Gantt chart for the chromosome (Make-span=33)

jobs in various machines as per the operation sequence and as dictated by the chromosome. The placement of jobs for the chromosome discussed is shown in Fig. 1.

2.6 New population creation

Creation of new population deals with selection of chromosomes and application of crossover and mutation operators.

2.6.1 Reproduction

At this step, we select good strings in the current population to form the mating pool. Roulette-wheel selection operator is used for reproduction. The action of reproduction operator is to clear the inferior points from further consideration by probabilistic elimination.

$x = p\_size$

$p(x) = F(x) \Big/ \sum F(x)$

$x = 1$

p(x)=Probability of selection of chromosome x.

Through the generation of random numbers the chromosomes are probabilistically picked in proportion to their weight and put into the mating pool.

One of the important and distinctive features of all Genetic Algorithms is the population handling technique. The original GA adopted a generational replacement policy, according to which the whole population is replaced in each generation. Conversely, the steady-state policies used by much subsequent Genetic Algorithms selectively replace the population.

It is possible, for example, to keep one or more population members for several generations, while those individuals sustain a better fitness than the rest of the population [10]. Steady state replacement policy has been adopted in this attempt to retain the current best of the strings in the population always. As the best individual is copied, probably, more than once, the second and other copies of the best individual are also disturbed, keeping open the chance of the present best string becoming still better. The evaluation and reproduction of chromosomes is shown in Table 1.

2.6.2 Crossover

Gen and Cheng [11] presented that the performance of Genetic Algorithm depends, to a great extent, on the performance of the crossover operator used. A number of crossover operators are available for use with operation based representation of jobs. Among them are the GOX (generalized order crossover) proposed by Bierwirth [12] and GPX (generalized position crossover) proposed by Mattfeld [13]. The GPX operator is reported to be better in performance compared to GOX operator [14]. The inversion operator (INV) has been reported to be surpassing all the modified forms of crossover especially tailored to deal with combinatorial problems [8]. It is also stated in the earlier reports that order based mutation when used at small rates and in combination with inversion may be useful for finer adjustments.

Table 1 Evaluation and reproduction of chromosomes

Ch no.	Chromosomes in Initial population	Objective function value f(x)	Fitness function value F(x)	Probability of selection p(x)	Cumulative probability of selection	Rand no.	Chrom. no.	Chromosomes for mating pool
1	221133332121	36	8	0.1194	0.1194	0.28	3	133131321222
2	312332213211	36	8	0.1194	0.2388	0.78	9	211131332322
3	133131321222	39	5	0.0746	0.3134	0.78	9	211131332322
4	213233232111	42	2	0.0299	0.3433	0.09	1	221133332121
5	331311213222	44	0	0.0000	0.3433	0.38	6	332312312121
6	332312312121	37	7	0.1045	0.4478	0.42	6	332312312121
7	311311332222	44	0	0.0000	0.4478	0.93	10	211233332112
8	323312113212	33	11	0.1642	0.6120	0.32	4	213233232111
9	211131332322	31	13	0.1940	0.8060			211131332322*
10	211233332112	31	13	0.1940	1.0000	0.13	2	312332213211

*Best string is taken to the mating pool

The working of GPX operator is explained below. Let the parent strings are as follows.

– parent1=3 2 3 1 1 3 2 2 3 1 1 2
– parent2=1 3 3 1 3 1 3 2 1 2 2 2

First a substring is chosen from the donating chromosome. The underlined sub-string is taken from parent 1. It consists of two operations of job 1 (index 1 and 2), one operation of job 3 (index 3) and one operation of job 2 (index 2).

– parent1=3 2 3 ***1 1 3 2*** 2 3 1 1 2
– parent2=***1*** 3 3 ***1 3*** 1 3 2 1 ***2*** 2 2

After choosing the substring all the operations of the sub-strings are deleted with respect to their index in the receiving chromosome (parent 2).

– parent1=3 2 3 ***1 1 3 2*** 2 3 1 1 2
– parent2=***1*** 3 3 ***1 3*** 1 3 2 1 ***2*** 2 2
– Offspring=3 3 1 1 1 3 2 3 2 1 2 2

Finally the donor's substring is implanted into the receiver at the position where it occurs in the donor to create the offspring.

The inversion operator (INV), in its mechanism, corresponds basically to the inversion operator firstly described by Holland [15]. In operation it randomly selects two points (genes), known as the points of inversion, in the chromosome selected for inversion, and inverts the sequence between these points. In this study only one inversion point is selected and the other point is taken to be the last gene. A typical feature of the operator is that it operates on individual strings and not on pairs. Each chromosome transforms itself into a new chromosome, without combining with any other chromosome. Multiple inversions have been adopted in the algorithm SWAP-GATSP for solving TSP [9].

If a chromosome qualifies for crossover, a random number from 1 to m*n is generated to select the site for crossover (inversion) operation. When the inversion point is, say 3, inversion happens as indicated below.

Before inversion: 3 2 3 ***1 1 3 2 2 3 1 1 2***
After inversion: 3 2 3 ***2 1 1 3 2 2 3 1 1***

Crossover is the operation mainly responsible for the search of new chromosomes/strings. The probability of crossover, p_cross, is taken as 0.80 to cover 80% of the chromosome for inversion to produce children. A random number between 0 and 1 is generated for each chromosome and crossover (inversion) is effected if the number is less than or equal to p-cross (0.80). The process is repeated for 9 (p_size-1) chromosomes, keeping the best of the chromosomes in the mating pool undisturbed. As the operator selected for crossover is of inversion type, it operates on individual chromosomes.

2.6.3 Mutation

The next step is to perform mutation on chromosomes in the intermediate population. Mutation is to maintain diversity in the population. This operator in Genetic Algorithm enables to explore the search space properly before converging to a region prematurely reaching only a local minimum. The probability of mutation is taken as 0.05 to cover 5% of the chromosomes for mutation. A random number between 0 and 1 is generated and mutation is effected, if the number is less than or equal to p-mute (0.05). Order-based mutation (OBM) has been applied, wherein two genes at random positions in the chromosome are swapped. Two random numbers between 1 to m*n (Machines*Jobs) are generated for the selection of genes and the selected genes are interchanged for position.

When the genes selected for mutation are say, 6 and 9, mutation takes place as follows.

– Before mutation: 3 2 3 2 1 ***1*** 3 2 ***2*** 3 1 1
– After mutation: 3 2 3 2 1 ***2*** 3 2 ***1*** 3 1 1

The above steps are repeated with the new population for the required number of generations. The application of inversion and mutation operators on the chromosomes is shown in Table 2.

Table 2 Application of crossover and mutation operators

Ch. no.	Population in the mating pool	Rand no.	Crossover point, if selected	Population after crossover	Rand no.	Genes for mutation, if selected	Population after mutation
1	133131321222	0.86	N.S.	133131321222	0.44	N.S.	133131321222
2	211131332322	0.82	N.S.	211131332322	0.51	N.S.	211131332322
3	211131332322	0.59	7	211131322323	0.94	N.S.	211131322323
4	221133332121	0.76	4	221112123333	0.12	N.S.	221112123333
5	332312312121	0.88	N.S.	332312312121	0.51	N.S.	332312312121
6	332312312121	0.78	7	332312312121	0.04	4, 6	332213312121
7	211233332112	0.95	N.S.	211233332112	0.25	N.S.	211233332112
8	213233232111	0.47	8	213233231112	0.41	N.S.	213233231112
9	211131332322**	Not genetically operated					
10	312332213211	0.37	3	312112312233	0.66	N.S.	312112312233

p_cross=0.80; p_mute=0.05; N.S. – Not selected
**One copy of the best string in the mating pool is not genetically operated

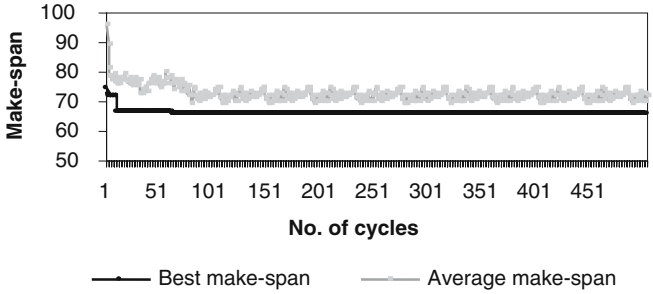

Fig. 2 A typical online performance of the proposed method with INV operator on the mt06 benchmark problem

2.7 Output

The solution, which gets improved during each iteration/ generation, is available as the result.

3 Validation

To validate the proposed heuristic, two instances from the suite of Fisher and Thompson (mt06 and mt10) [16], two instances set by Adams et al. (abz5 and abz6) [17], four instances set by Yamada and Nakano (yn1, yn2, yn3 and yn4) [5], six instances set by Applegate and Cook (orb01, orb02,orb07,orb08,orb09 and orb10) [18] and six instances set by Lawrence (la16,la36,la37,la38,la39 and la40) [19] have been selected (problems are available in operations research (OR) library [20]). Roulette wheel selection, inversion reordering (INV) operator and order based mutation (OBM) are selected as GA operators. The inversion and mutation probabilities are assigned to be 0.80 and 0.05 respectively. It has been decided to have the population size as 10 and number of generations as 1,000.

A typical benchmark problem (Fisher and Thompson 6×6 alternate name - mt06) [16] has been taken for illustration. The first line contains the number of jobs and the number of machines, followed by one line for each job listing the machine number and processing time for each step of the job. The machines are numbered starting with 0.

```
6  6
2  1  0  3  1  6  3  7  5  3  4  6
1  8  2  5  4  10 5  10 0  10 3  4
2  5  3  4  5  8  0  9  1  1  4  7
1  5  0  5  2  5  3  3  4  8  5  9
2  9  1  3  4  5  5  4  0  3  3  1
1  3  3  3  5  9  0  10 4  4  2  1
```

Best schedule found in 1,000 generations:

```
5 2 2 0 5 3 0 4 3 3 5 1 1 4 4 0 0 1
2 5 0 3 5 5 4 4 2 1 2 3 1 3 2 4 1 0
```

Best Make-span=65; CPU time=0.60 secs.

The variation in average make-span and best make-span recorded in first 500 cycles are shown in Fig. 2.

Table 3 Result showing performance of INV and GPX operators

Problem no.	Problem instance	Size	GPX Operator		INV Operator	
			Make-span	CPU Time(sec)	Make-span	CPU Time(sec)
1	abz5	10×10	1,887	7	1,775	1.54
2	abz6	10×10	1,392	6	1,356	1.53
3	mt06	6×6	69	5	65	0.60
4	mt10	10×10	1,636	7	1,395	1.65
5	yn1	20×20	1,857	22	1,541	6.15
6	yn2	20×20	1,822	21	1,564	6.15
7	yn3	20×20	1,856	18	1,699	6.10
8	yn4	20×20	1,862	19	1,852	6.15
9	la36	15×15	2,168	13	1,840	3.52
10	la37	15×15	2,431	13	2,232	3.51
11	la38	15×15	2,153	15	2,006	3.46
12	la39	15×15	2,310	12	2,165	3.46
13	la40	15×15	2,146	13	2,007	3.52
14	orb07	10×10	616	7	561	1.56
15	orb08	10×10	1,533	7	1,377	1.64
16	orb09	10×10	1,555	7	1,411	1.59
17	la16	10×10	1,523	7	1,432	1.59
18	orb10	10×10	1,499	9	1,392	1.60
19	orb01	10×10	1,696	6	1,428	1.59
20	orb02	10×10	1,370	6	1,289	1.59

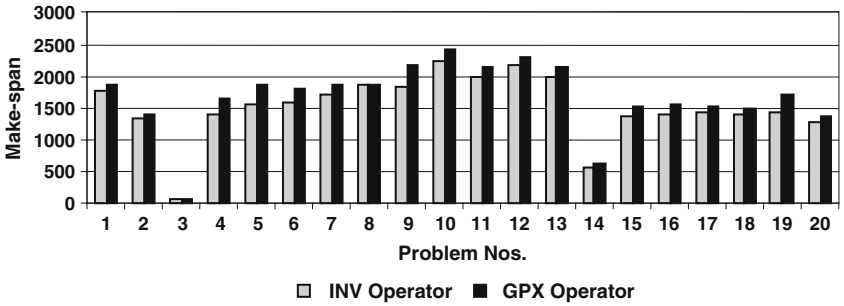

Fig. 3 Bar chart between make-span and problem instances

4 Results and discussions

The details about the name of the problem, size of the problem, the make-span found and the CPU time using inversion reordering (INV) operator and generalized position crossover (GPX) operator are presented in Table 3. The results are also presented in the form of charts in Figs. 3 and 4.

The same parameter settings used in the technical paper with GPX operator [14] have been adopted in this study for ease of comparison. *The values of make-span presented under INV column are the values observed in a typical run and not the best. Better values have been observed in many runs*. It may be inferred that inversion (INV) reordering operator clearly performs better than generalized position crossover (GPX) operator.

The experiments with GPX operator were reported to have been conducted with Intel Celeron - MMX CPU at 266 MHz, 32 MB RAM computing system. Experiments with inversion (INV) operator, reported in this paper, have been conducted using an Intel Pentium MMX CPU at 200 MHz, 32 MB RAM system. The algorithm has been implemented in C.

5 Conclusions

In this paper, an attempt has been made to evaluate the performance of Genetic Algorithm in solving job shop scheduling problems. The performance of inversion (INV) reordering operator has been compared with Generalized Position Crossover (GPX) operator. The make-span and CPU time observed are presented in the form of tables and charts. The results show that inversion reordering operator performs better than the Generalized Position Crossover operator. The heuristic is also found to give better solutions in shorter computational time.

6 Statement of contribution

This paper deals with the job shop scheduling problems. *The objective is to find the sequence of operations of jobs in various machines for minimum make-span* - the time required to complete all jobs. *Operation based representation scheme has been considered* in solving the problems. One of the soft computing techniques, namely, Genetic algorithm has been applied in this study. *The performance of the algorithm for two different operators viz Generalized position crossover (GPX) and inversion reordering (INV) operator have been compared*. The results for a set of 20 benchmark problems available in OR library have been presented. The performance of generalized position crossover operator for operation based representation reported in Production Planning and control (An International Journal) on the 20 problem instances have been taken. *The finding is that by using the inversion reordering operator better solutions in make-span are found with appreciable reduction in computational time*.

Acknowledgements Thanks are due to Dr. C. Rajendran, Professor, Department of Humanities and Social sciences, Indian Institute of Technology, Chennai, India for useful suggestions that helped improve the paper.

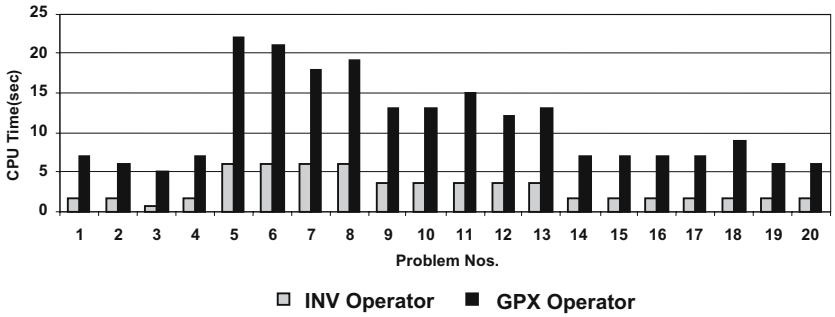

Fig. 4 Bar chart between CPU time and problem instances

References

1. Franch S (1981) Sequencing and scheduling – an introduction to the mathematics of the job-shop. Wiley, New York
2. Baker KR (1974) Introduction to sequencing and scheduling. Wiley, New York
3. Pinedo M, Chao X (1999) Operation scheduling with applications in manufacturing and services. McGraw-Hill, Boston
4. Nakano R, Yamada T (1991) Conventional genetic algorithms for job-shop problems. In: Belew RK, Booker LB (eds) Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, pp 477–479
5. Yamada T, Nakano R (1992) A genetic algorithm applicable to large-scale job shop problems. In: Manner R, Manderick B (eds) Parallel problem solving from nature: PPSN II, Elsevier Science, North-Holland, pp 281–290
6. Fang H, Ross P, Corne D (1993) A promising genetic algorithm approach to job shop scheduling, rescheduling and open shop scheduling. In: Forrest S (ed) Proceedings of the Fifth International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, pp 375–382
7. Gen M, Tsujimura J, Kubota E (1994) Solving job-shop scheduling problem using genetic algorithms. In: Gen M, Kobayashi S (eds) Proceedings of the 16th International Conference on Computers and Industrial Engineering, Ashikaga, Japan, pp 576–579
8. Ferreira C (2002) Combinatorial Optimization by Gene expression programming: Inversion revisited. Argentine Symposium on Artificial Intelligence, pp 160–174
9. Shubhra Sankar Ray, Sanghamitra Bandyopadhyay, Sankar K. Pal (2004) New operators of Genetic Algorithms for Traveling Salesman Problem. IEEE Int. Conf. on Pattern Recognition (ICPR 04) pp 497–500
10. Filho JLR, Treleaven PC (1994) Genetic algorithm programming environments. IEEE Comput 27(6):28–43
11. Gen M, Cheng R (1997) Genetic algorithms and engineering design. Wiley, New York
12. Bierwirth C (1995) A generalized permutation approach to job shop scheduling with genetic algorithm. OR Spectrum 17: 87–92
13. Mattfeld DC (1996) Evolutionary search and the job shop: investigations on genetic algorithms for production scheduling. Physica-Verlag, Heidelberg, Germany
14. Ponnambalam SG, Aravindan P, Sreenivasa Rao P (2001) Comparative evaluation of genetic algorithms for job-hop scheduling. Prod Plan Control 12(6):560–574
15. Holland JH (1975) Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor, MI, USA
16. Fisher H, Thompson GL (1963) Probabilistic learning combinations of local job-shop scheduling rules. In: Muth JF, Thompson GL (eds) Industrial scheduling. Prentice-Hall, Englewood Cliffs, NJ, USA, pp 225–251
17. Adams J, Balas E, Zawak D (1988) The shifting bottle-neck procedure for job-shop scheduling. Manage Sci 34(3):391–401
18. Applegate D, Cook W (1991) A computational study of the job-shop scheduling problems. ORSA-J Comput 3:149–156
19. Lawrence S (1984) Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques. Technical report, Graduate School of Industrial Administration, Carnegie Melon University, USA
20. Beasley JE (1990) OR-Library. http://mscmga.ms.ic.ac.uk/info.html. Cited 20 November 2004