Int J Adv Manuf Technol (2008) 39:965–974
DOI 10.1007/s00170-007-1354-8

ORIGINAL ARTICLE

An effective hybrid genetic algorithm for the job shop scheduling problem

Chaoyong Zhang · Yunqing Rao · Peigen Li

Received: 14 July 2005 / Accepted: 18 October 2005 / Published online: 8 July 2008

Abstract From the computational point of view, the job shop scheduling problem (JSP) is one of the most notoriously intractable NP-hard optimization problems. This paper applies an effective hybrid genetic algorithm for the JSP. We proposed three novel features for this algorithm to solve the JSP. Firstly, a new full active schedule (FAS) procedure based on the operation-based representation is presented to construct a schedule. After a schedule is obtained, a local search heuristic is applied to improve the solution. Secondly, a new crossover operator, called the precedence operation crossover (POX), is proposed for the operation-based representation, which can preserve the meaningful characteristics of the previous generation. Thirdly, in order to reduce the disruptive effects of genetic operators, the approach of an improved generation alteration model is introduced. The proposed approaches are tested on some standard instances and compared with other approaches. The superior results validate the effectiveness of the proposed algorithm.

Keywords Job shop scheduling problem · Genetic algorithm · Crossover operator · Local search

1 Introduction

The general job shop scheduling problem (JSP) with the makespan criterion can be described by a set of n jobs that must be processed on m machines. Each job composes of several operations, and the operations of a given job have to be processed in a given order. Each operation uses one of the m machines for a fixed duration. Each machine can process at most one operation at a time, and once an operation initiates processing on a given machine, it must complete processing on that machine without interruption. The objective is to find the optimal schedule of the operations on the machines, taking into account the precedence constraints, which minimizes the makespan, i.e., the finish time of the last operation completed in the schedule.

JSP is one of the most difficult NP-hard combinatorial optimization problems whose complexity grows very fast with the problem size. During the last three decades, many solution methods have been proposed to solve the JSP. Those approaches can be divided into two categories: exact methods and approximation algorithms. Exact methods, such as branch and bound, linear programming and decomposition methods, guarantee global convergence and have been successful in solving small instances, including the notorious 10×10 instance of Fisher and Thompson proposed in 1963 and only solved 20 years later. But for the big instances there is a need for approximation algorithms, which include priority dispatch, the shifting bottleneck approach, local search, and heuristic methods. In recent years, modern heuristic methods, such as genetic algorithm (GA) [1–3], simulated annealing (SA) [4], tabu search (TS) [5, 6], have captured the interest of many researchers, because they are able to attain high-quality solutions within reasonable computational times. A comprehensive survey of job shop scheduling techniques can be found in Jain [7] and Blazewicz [8].

Among the heuristic algorithms, the genetic algorithm, inspired by the process of Darwinian evolution, has been widely applied in many engineering fields, especially in the

C. Zhang (✉) · Y. Rao · P. Li
School of Mechanical Science & Engineering,
Huazhong University of Science & Technology,
Wuhan 430074, People's Republic of China
e-mail: superhust@163.com

Table 1 A sample 2×2 problem

	Operations			
Job	1	2	1	2
	Machine sequence		Processing time	
j1	2	1	2	4
j2	2	1	2	3

production scheduling field. The GA is based on the survival of the fittest and involves some selection, crossover and mutation operations. GA exhibits parallelism, contains certain redundancy and historical information of past solutions, and is suitable for implementation on massively parallel architecture [9]. However, Due to the stubborn nature of the JSP, simple GA is difficult to apply directly and successfully to the difficult problem. Much effort in the literature has focused on hybrid methods. The genetic algorithm is very effective at performing global search but often suffers from premature convergence, while local search is good at fine-tuning but often falls into local optima. So the role of local search in the context of the genetic algorithm has been taken into serious consideration. The hybrid genetic algorithm can complement the properties of the genetic algorithm and the local search method [10]. The genetic algorithm is used to perform global search to escape from local optima, while the local search is used to perform fine-turning.

In this paper, an effective hybrid genetic algorithm for the job shop scheduling problem is presented. The remainder of the paper is organized as follows. In Section 2, a new type of schedule named full active schedule is proposed. In Section 3, we present our approach to solve the job shop scheduling problem: representation scheme, schedule generation procedure, local search procedure, new genetic operators and the framework of hybrid genetic algorithm. Section 4 reports the computational results. The conclusions are made in Section 5.

2 Types of schedules

In general, schedules can be classified into three types: semi active schedule, active schedule and non-delay schedule [11]. An active schedule can be obtained by shifting the operations to the left of a semi-active schedule without delaying other jobs, such reassigning is called a permissible left shift, and a schedule with no more permissible left shifts is called an active schedule. Therefore the set of active schedules is a subset of semi active schedules. The optimal schedule is in the set of all active schedules, so it is safe and efficient to limit the search space to the set of all active schedules.

Although repairing a semi-active schedule to the active one improves the makespan, the set of active schedules is usually very large and contains many schedules with relatively large delay times. The simple two-job and two-machine problem described in Table 1 is taken for example. Figure 1(a) shows an active solution of this problem which no more permissible left shifts can improve its makespan. Figure 1(b) is attained by shifting the operation (J1, 1) to the right of the operation (J2, 1) on machine2 (M2). It can be seen from Fig. 1 that there sometimes are obvious improvements that can be attained by right shifts the operations of the active schedule.

This paper proposes a new type of schedule: full active schedule (FAS), which can be defined as a schedule with no more permissible left shifts and right shifts. A FAS can be obtained by shifting the permissible operations to right of an active schedule without delaying other jobs (The example can be seen from Fig. 1), so the set of full active schedules is a subset of active schedules. Because the makespan of a FAS is less than or equal to the makespan of the corresponding active schedule, we could draw the conclusion the optimal schedule is in the set of all full active schedules. Figure 2 illustrates where the set of full active schedules is. Using the FAS sets, we can further reduce the solution space and get better solutions. The full active schedule generator procedure based the operation-based representation is proposed in Section 3.1.2.

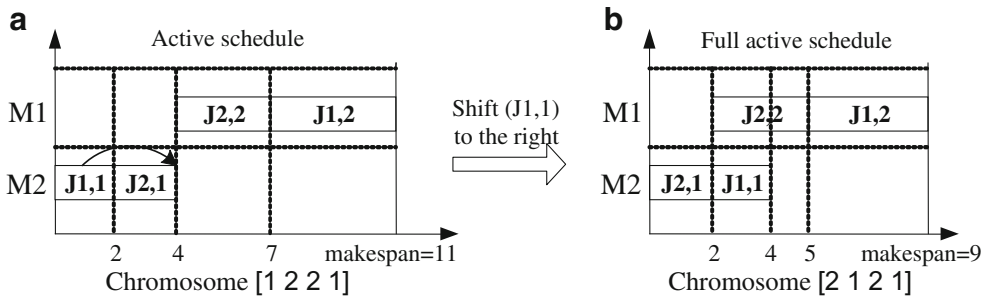

Fig. 1 Permissible right shift for an active schedule

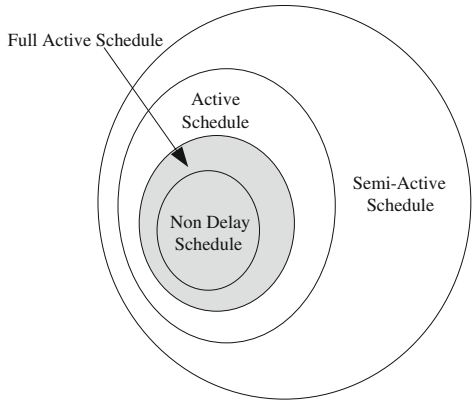

Fig. 2 Full active schedules

3 Hybrid genetic algorithm for JSP

3.1 Schedule generator procedure

3.1.1 Representation

The genetic algorithm described in this paper uses an operation-based representation that uses an unpartitioned permutation with m-repetitions of job numbers [2, 12, 13]. In this representation, each job number occurs *m* times in the chromosome. By scanning the chromosome from left to right, the *k*-th occurrence of a job number refers to the *k*-th operation in the technological sequence of this job. The important feature of the operation-based representation is that any permutation of the chromosome can be decoded to a feasible schedule.

Consider the three jobs and three machines problem given in Table 2. Suppose a chromosome is given as [2 1 1 2 2 3 3 1 3], Because each job consists of three operations, it occurs exactly three times in the chromosome. The fourth gene of the chromosome in this example is number 2. Because the number 2 has been repeated twice, the number 2 represents the second operation of job 2. A schedule is decoded from a chromosome with the following procedure: the first operation in the list is scheduled firstly, then the second operation, and so on. Each operation under treatment is allocated in the best available processing time for the corresponding machine the operation requires (also called permissible left shift). The process is repeated until all operations are scheduled. A schedule generated by the procedure can be guaranteed to be an active schedule [14, 15]. Then, using the active-decoding process, we can get the corresponding active solution shown in Table 2 and the active chromosome [2 3 1 1 2 2 3 1 3]. The active chromosome and the feasible solution Table 3 can be converted into each other; however two or more different chromosomes can be decoded to an identical solution.

3.1.2 Schedule generation procedure

Through selection, crossover operator, and mutation operator, the genetic algorithm is responsible for evolving the chromosome in each generation. The schedule generation procedure combines the FAS generation procedure and the local search procedure for feedback of the makespan and the full active chromosome to evolutionary process.

The objective of the schedule generation procedure is to improve the chromosome and obtain their makespan. The algorithm described in Section 3.1.1 can generate an active schedule. Using the same algorithm to the active schedule, we can get a full active schedule with only small modifications. By reversing the chromosome based on the operation-based representation and all of the technological sequences, a given schedule can be converted to another schedule. The new schedule is equivalent to the original one with the same makespan (the same critical path) and the reversed chromosome (i.e., reversed job processing sequences on same machine). Through left shifting the new schedule, we can obtain the makespan and chromosome of the full active schedule. For example, consider the simple 2×2 problem described in Table 1. The chromosome of the schedule shown in Fig. 1(a) is [1 2 2 1]. By reversing the chromosome (obtained the reversed chromosome [1 2 2 1]) and the technical sequence (shown in Table 4), the given schedule shown in Fig. 1(a) is converted into the new schedule shown in Fig. 3(a). The new schedule is equivalent to the original one with the same makespan and the reversed job processing sequences on same machine. Therefore we could be seen that the makespan and chromosome of a full active schedule shown in Fig. 3(b) can be obtained by using the algorithms (in Section 3.1) to the reversed chromosome and the reversed technological sequences of a given active schedule shown in Fig. 1(a). Compared Fig. 3(b) with Fig. 1(b), we must note that the sequence of the chromosome

Table 2 Example of 3×3 problem

Job	Operations routing (Processing time)		
j1	1(3)	2(1)	3(2)
j2	3(1)	1(5)	2(3)
j3	2(3)	3(2)	1(3)

Table 3 A 3×3 feasible solution

Machine	Job sequence		
m1	1	2	3
m2	3	1	2
m3	2	3	1

Table 4 Reversal of the technical sequence

	Operations			
Job	1	2	1	2
	Machine sequence		Processing time	
j1	1	2	4	2
j2	1	2	3	2

and job sequences shown in Fig. 3(b) is reversed with the given result.

Figure 4 illustrates the steps of the schedule generation procedure applied to each chromosome generated by the genetic algorithm. The schedule generation procedure firstly generates the active schedule and the corresponding full active schedule, and then applies the local search to each of them to improve them. If the local search improves the makespan, the schedule generation procedure continuously generates the full active schedule and applies the local search to improve it; otherwise, improvement of the chromosome (or schedule) ends, and the full active chromosome and the corresponding makespan is obtained for the evolutionary process. The local search procedure is introduced in the next section.

3.1.3 Local search procedure

For the JSP, a key component of a solution is the critical path, which is the longest route from source to sink in the disjunctive graph. It is possible to decompose the critical path into a number of blocks where a block is a maximal sequence of adjacent critical operations that require the same machine. Because the permutation of non-critical adjacent operation cannot improve the objective function and even may create an infeasible solution, an efficient method can be obtained by introducing a transition operator that exchanges a pair of consecutive operations only on the critical path and forms a neighborhood.

In this paper, we focus particularly on the approach of Nowicki and Smutnicki (1996), which is noted for proposing and implementing the most restrictive neighborhood in the literature. Unlike the approach that generates only a single arbitrary critical path, our approach generates all critical paths. The critical path thus gives rise to the following neighborhood of moves. Given b blocks, if $1<g<b$, then swap only the first two and the last two block operations. Otherwise, if $g=1$ (b), swap only the last (first) two block operations (see Fig. 5). In the case where the first and/or the last block contain only two operations, these operations are swapped. If a block contains only one operation no swap is made [16].

The local search used in this paper is the standard ascent method. If the swap improves the makespan, it is accepted. Otherwise, the swap is undone. Once a swap is accepted, the critical path is changed and a new critical path must be identified. If no swap of the first or the last operations in any block of critical path improves the makespan, the local search ends [17].

3.2 Crossover operation

Crossover can be regarded as the backbone of the genetic algorithm. It intends to inherit the properties of two parent solutions to two offspring solutions. To apply crossover operation successfully to the JSP, we must satisfy the following criteria: completeness, feasibility, non-redundancy and characteristics preservation [18]. We think that the characteristics preservation and the feasibility are the most important criteria to design crossover operation in JSPs. In this paper, a new crossover operator named precedence operation crossover (POX) is proposed for the operation-based representation, which can satisfy the characteristics preservation and the feasibility between parents and their children better.

The effective crossover operator proposed in this paper is described as follows. Given chromosome, parent1 and parent2, crossover applied POX generates the children, child1 and child2, by the following procedure:

1. Randomly choose the set of job numbers, {1, 2... n} into one nonempty exclusive subset J1.
2. Copy those numbers in J1 from parent1 to child1 and from parent2 to child2, preserving their locus.
3. Copy those numbers in J1, which are not copied at step 2, from parent2 to child1 and from parent1 to child2, preserving their order.

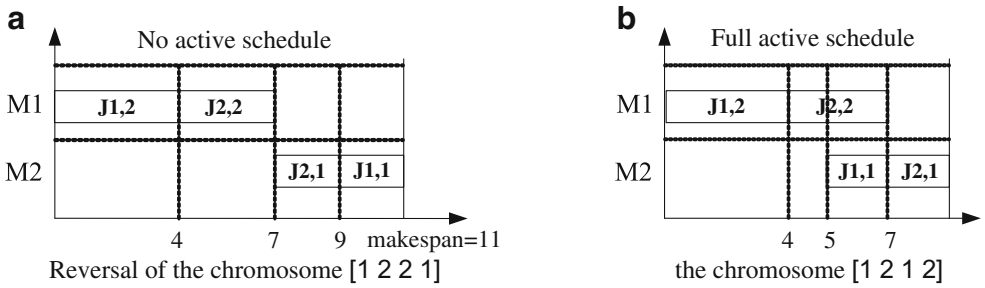

Fig. 3 The chromosome and makespan of the full active schedule generate

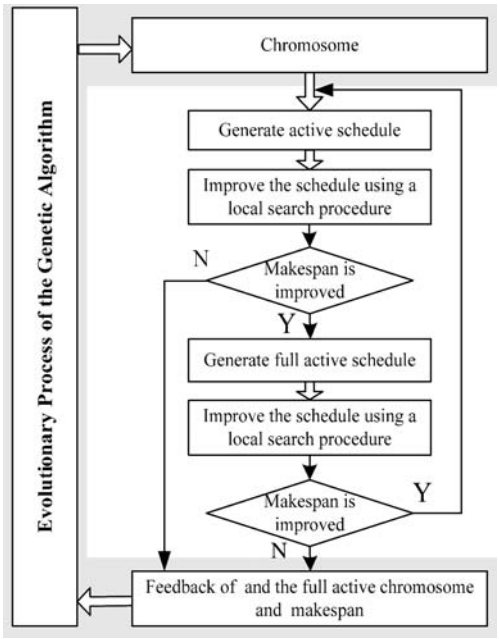

Fig. 4 The schedule generation procedure

Figure 6 shows an example of the three-job and three-machine problem; chromosome of parent1 and parent2 is {3 2 2 2 3 1 1 1 3} and {1 1 3 2 2 1 2 3 3}. The locus of job {2} is preserved. The crossover generates two children chromosomes, child1 {1 2 2 2 1 3 1 3 3} and child2 {3 3 1 2 2 1 2 1 3}. It can be seen that child1 preserves the locus and order of job {2} in parent1 and the order of job {1, 3} in parent2, respectively, and child2 preserves the locus and order of job {2} in parent2 and the order of job {1, 3} in parent1, respectively. Therefore POX is excellent in the characteristics preservation.

3.3 Mutation operation

Mutation is just used to produce perturbations on chromosomes in order to maintain the diversity of population. In this paper, two types of mutation operators named inversion mutation and insertion mutation are used. Inversion mutation serves to maintain the diversity in population. Insertion mutation is used not only to produce small perturbations but also to perform intensive search in order to find an improved offspring. Inversion mutation and insertion mutation act on half of the population, respectively. Two mutations are described as follows:

1. Inversion mutation inverts the substring between two different random positions.
2. Insertion mutation selects two elements randomly and insert the back one before the front one. Implements the insertion procedure n times and chooses the best one.

3.4 Designing a hybrid genetic algorithm for JSP

GA exhibits parallelism, contains certain redundancy and historical information of past solutions. Critical components of past good solutions can be captured, which can be combined together via crossover to form high quality solutions. Unfortunately, GA is also prone to loss of solutions and theirs substructures due to the disruptive effects of genetic operators. This is because new solutions produced by the genetic operators are always accepted, even if they are significantly inferior to older solutions. This characteristic can lead to disruption, where good solutions are lost or damaged. Therefore a simple GA often produces premature convergence and poor results in the difficult combinational problems. In this paper, we introduce an improved generation alternation model that employed in the hybrid GA for JSP, which could better preserve the meaningful characteristics of the previous generation and reduce the disruptive effects of genetic operators. The framework of the hybrid GA is shown in Fig. 7.

In contrast to a simple genetic algorithm, the hybrid GA in this paper has superior features. Firstly, an improved generation alternation model is introduced. The conception of generation alternation model is that crossover is applied to the two parent n times and $2n$ offspring are generated; the two unequal best individuals in those offspring are selected to the next generation. By comparative experiments about the optimization of traveling salesman problem, a generation alternation model gives better results than a model that does not employ it, especially in dealing with the difficult problems [19]. In this paper, we improved the generation alternation model as following: if the better individual of two children outperforms the better individual

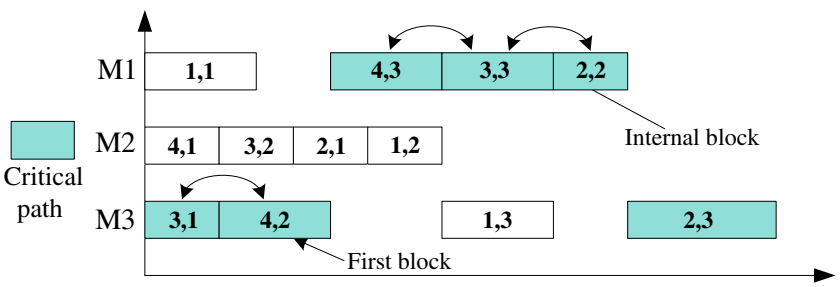

Fig. 5 Permutation of operations on a critical path

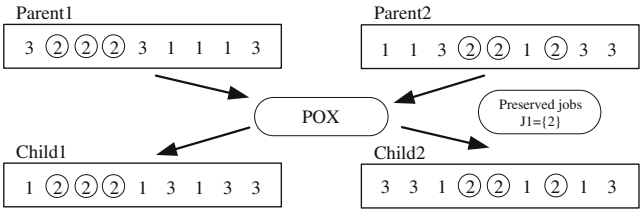

Fig. 6 POX crossover

of two parents in each crossover process, then the better individual of two parents is replaced by the better individual of two children, otherwise two parents don't change and crossover continues. Our experimental results show this improved approach could reduce the search times and obtain better results. Secondly, the crossover rate is replaced by a novel method. Consider two individuals *P1*, *P2* which select from the P_{old}. If the fitness f_{P1}, f_{P2} of two individuals *P1*, *P2* is not equal, then implements the crossover operator. Otherwise, implements the mutation operator. This new method of crossover rate can vary dynamically, adaptively in response to the state of convergence. For example, at the beginning of the evolution period, the crossover rate is big; whereas at the end of the convergence period, the crossover rate decreases and the mutation rate becomes big; this characteristic of the new crossover rate can avoid premature convergence better and relax the parameters dependence. This novel method proposed in this paper could also be used for the simple genetic algorithm. The outline of the proposed hybrid genetic algorithm procedure for JSP is illustrated as follows:

4 Computational results

To illustrate the effectiveness and performance of the algorithm described above, we consider some instances from the standard JSP test problems as follows: Fischer and Thompson (1963) instances FT06, FT10, FT20, Lawrence (1984) instances LA01 to LA40, Adams et al (1988) instances ABZ5 to ABZ9, Applegate and Cook (1991) instances ORB01 to ORB10 and Yamada and Nakano (1992) instances YN1 to YN4. All test instances were downloaded from Beasley's OR-Library, http://mscmga.ms.ic.ac.uk.

In our experiments, population size is 200. The number of crossover times is equal to the number of jobs, and the mutation rate (P_m in Fig. 8) is 0.8. The top 10% elite individuals from the previous population chromosome are copies for the next generation. The algorithm was terminated when an optimal solution was found or after 50–80 generations of the algorithm, and each instance is randomly run 20 times. The algorithm was implemented in Visual C++ and the tests were run on a computer with Pentium IV1.6G and 256MB RAM.

The hybrid genetic algorithm finds the optimal solutions for the ft10 problem almost every time in less than 20 seconds on average. Here, we regard FT06, LA01, LA06, LA11, ABZ5 and so on as easy problems, because they can be easily solved by many methods. Therefore we don't present those results in detail. Table 5 shows the makespan performance statistics of the hybrid genetic algorithm for the 15 difficult problems selected from FT,

```
Begin
  Generate randomly an initial population with P_size individual;
  Evaluate the initial population with the schedule generation procedure;
Do{
  Reproduce the 10% elite individuals from P_old to P_new;
  Select a pair of individuals P1, P2 from the P_old, and their fitness is f_P1, f_P2 respectively.
  If (f_P1 != f_P2){
    Apply the improved generation alternation model and POX operator. Implement crossover n times and
    generate 2n offspring, select two unequal best individuals in the 2n offspring to the next generation.
  }
  else {
    Mutate the chosen pair of individuals by the probability of P_m .
  }
} Until (stop criterion has been satisfied)
End
```

The outline of the hybrid genetic algorithm

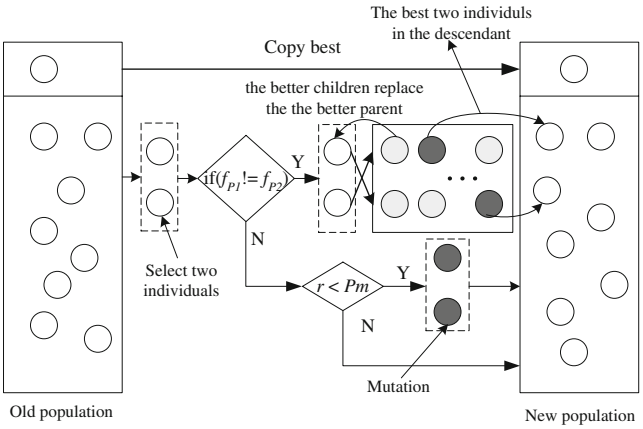

Fig. 7 Hybrid genetic algorithm procedure

LA and ABZ benchmarks. The algorithm was terminated when an optimal solution was found or after 50–80 generations of the algorithm. In the table, the column named LB lists the best known solution or best lower bound indicated in Jain [7], and the next columns named best, average, worst and S.D. list the best, average, worst and standard deviation makespan values obtained, over 20 runs, respectively, and the last two columns named n-opt and t-opt show the number of the optimal schedules obtained and the average cpu time of the optimal schedules obtained. Optimal solutions were found for nine out of the 15 problems. The small standard deviation indicates the stability of the hybrid genetic algorithm.

Table 6 summarizes the computational results of the fifteen tough problems by comparing with the following algorithms:

Hybrid genetic algorithm (HGA)	José Fernando Gonçalves (2002)[17]
GRASP (GRASP)	Aiex et al. (2003)[20]
Tabu search (TSAB′)	Nowicki and Smutnicki (1996)[6]
Tabu search method guided by SB (TSSB)	Ferdinando Pezzella et al. (2000)[21]
Guided local search with SB (SB-GLS)	Balas, E. and Vazacopoulos, A. (1998) [22]

It lists problem name, problem dimension, the best-known lower bound **(LB)**, and the solution obtained by our hybrid genetic algorithm and by each of the other algorithms. The last line shows the mean relative error (MRE), which is used to analyze the effectiveness of the proposed algorithm. The MRE is calculated from the best known lower bound (LB), and the upper bound (UB) that is given by the makespan of the best solution found by our algorithm. The "relative deviation" formula is 100×(UB – LB) / LB and was calculated for each instance of problems. Overall, the MRE our algorithm obtained is 0.28% and outperforms almost all others algorithms except for the famous SB-GLS5. Compared with those of a similar HGA which use a procedure that generates parameterized active schedule and is presented by J. F. Gonçalves, our computational results showed a great improvement both in terms of solution quality and computing time, validating the effectiveness of the full active schedule procedure. Table 7 presents the computational results of the ORB benchmarks, and the MRE our algorithm obtained is less than all others algorithms. Moreover, the tabu search algorithm had better performance in the stability. Table 8 presents the computational results of the YN benchmarks that are problems of dimension 20×20. Figure 8 shows the optimal solution of the LA38 15×15 problem that is one of the hardest problems.

5 Conclusions

This paper presents a hybrid algorithm combining genetic algorithm with local search for the JSP, in which GA performs global exploration among the population and the local search performs local exploitation around chromosomes. The chromosome representation of the problem is based on the operation-based representation. A new full active schedule procedure based on the operation-based representation is proposed to construct schedule, which can further reduce the search space. The superior results compared with other HGA prove the effectiveness of the full active schedule procedure. To preserve the meaningful characteristics of the previous generation and reduce the disruptive effects of genetic operators, a new crossover operator named precedence operation crossover (POX) is proposes for the operation-based representation and an improved generation alteration model is introduced. The approach is tested on some standard instances taken from the literature and compared with the other approaches. The MRE our algorithm obtained is less than almost all others algorithms, which validates the effectiveness of this approaches.

Further research is necessary to reduce the computing time. The TS algorithm showed better performance in computing times and stability, providing a further improvement procedure for our algorithm. If we apply the TS to instead the local search in the hybrid GA algorithm, better results might be obtained in reasonable times.

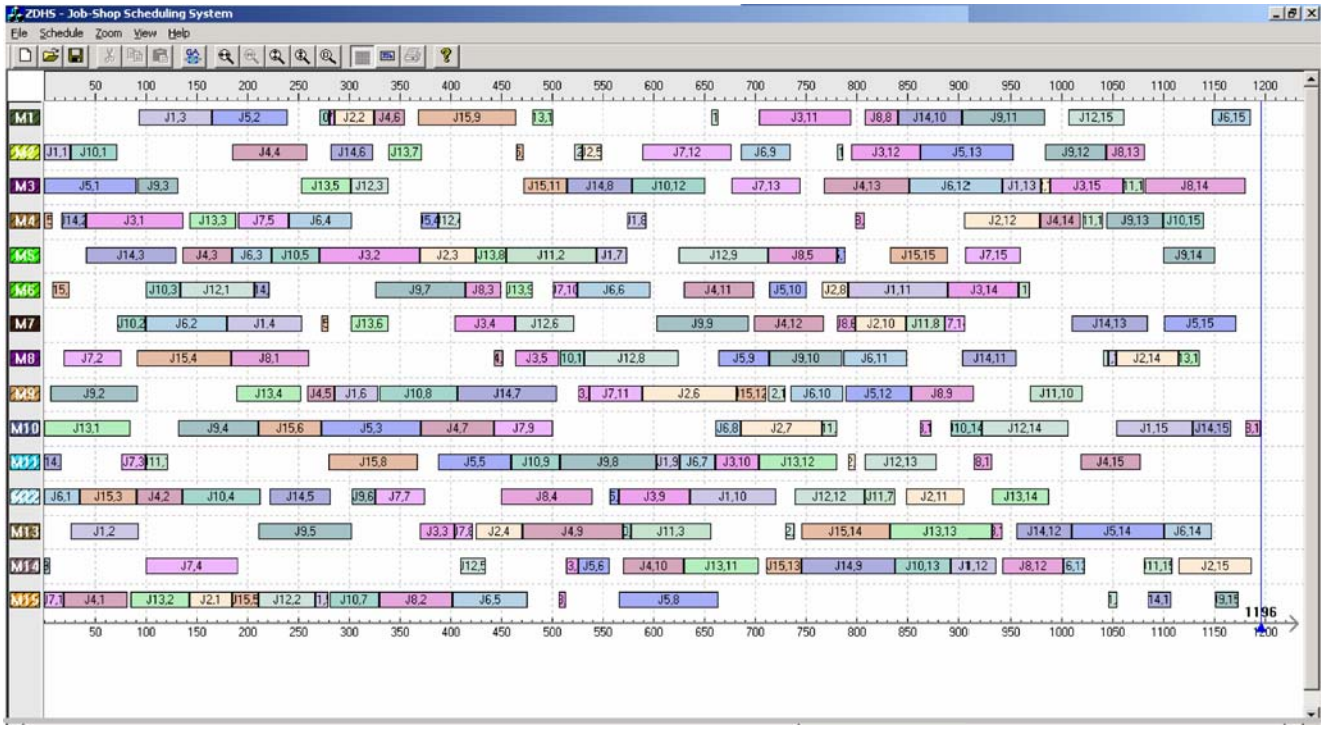

Fig. 8 The optimal solution of the LA38 15×15 problem

Table 5 Results of the 15 tough FT, LA and ABZ problems

Problem	Size	LB	Best	Average	Worst	S.D.	n-opt	t-opt
FT10	10×10	930	930*	930	930	0	20/20	12.6
FT20	20×5	1165	1165*	1166.05	1178	3.2	18/20	37.5
LA21	15×10	1046	1046*	1052.4	1059	3.6	1/20	109
LA24	15×10	935	935*	941.6	946	3.5	1/20	186
LA25	20×10	977	977*	981.95	984	2.5	3/20	156
LA27	20×10	1235	1236	1252.35	1264	8.3	–	–
LA29	20×10	1152	1162	1177	1191	8.4	–	–
LA36	15×15	1268	1268*	1279.6	1291	5.5	1/20	394
LA37	15×15	1397	1397*	1399.1	1411	4.1	12/20	228
LA38	15×15	1196	1196*	1203.5	1215	4.8	3/20	222.7
LA39	15×15	1233	1233*	1240.15	1249	2.6	1/20	252
LA40	15×15	1222	1224	1233.25	1242	5.6	–	–
ABZ7	20×15	656	666	671.4	678	3.1	–	–
ABZ8	20×15	665(645)	672	680.8	690	4.9	–	–
ABZ9	20×15	679(661)	682	688.9	695	3.9	–	–

*The optimal solutions were found by our algorithm

Table 6 Computational results of the 15 tough FT, LA and ABZ problems

							Balas, E.(1998)		
Prob.	Size	LB	Our HGA best	GRASP	HGA	TSSB	SB-GLS2	SB-GLS5	TSAB′
FT10	10×10	930	930	930	930	930	930	930	930
FT20	20×5	1165	1165	1165	1165	1165	1165	1165	1165
LA21	15×10	1046	1046	1057	1046	1046	1048	1046	1047
LA24	15×10	935	935	954	953	938	937	935	939
LA25	15×10	977	977	984	986	979	977	977	977
LA27	20×10	1235	1236	1269	1256	1235	1240	1235	1236
LA29	20×10	1152	1162	1203	1196	1168	1164	1164	1160
LA36	15×15	1268	1268	1287	1279	1268	1268	1268	1268
LA37	15×15	1397	1397	1410	1408	1411	1397	1397	1407
LA38	15×15	1196	1196	1218	1219	1201	1198	1196	1196
LA39	15×15	1233	1233	1248	1246	1240	1233	1233	1233
LA40	15×15	1222	1224	1244	1241	1233	1234	1224	1229
ABZ7	20×15	656	666	692	–	666	671	664	670
ABZ8	20×15	665(645)	672	705	–	678	676	671	682
ABZ9	20×15	679(661)	682	740	–	693	694	679	695
MRE			0.28	2.58	1.21	0.69	0.61	0.22	0.64

Table 7 Computational results of the ORB benchmarks

			Our HGA					Balas, E(1998)		
Prob.	Size	LB	Best	Average	t-opt[a]	GRASP	TSSB	SB-GLS2	SB-GLS5	TSAB′
ORB01	10×10	1059	1059*	1062.2	19.8	1059	1064	1059	1059	1059
ORB02	10×10	888	889	889.2	–	888	890	888	888	890
ORB03	10×10	1005	1005*	1005	22.2	1005	1013	1005	1005	1005
ORB04	10×10	1005	1005*	1008.6	15.5	1011	1013	1019	1013	1011
ORB05	10×10	887	887*	888.7	20.3	889	887	889	889	889
ORB06	10×10	1010	1010*	1014.1	22.5	1012	–	1010	1010	1013
ORB07	10×10	397	397*	399.4	8.6	397	–	397	397	397
ORB08	10×10	899	899*	899.5	15.8	899	–	899	899	913
ORB09	10×10	934	934*	934	6.9	934	–	934	934	941
ORB10	10×10	944	944*	944	15.3	944	–	944	944	946
MRE			0.01			0.10	0.10	0.87	0.10	0.38

*The optimal solutions were found by our algorithm
[a] **t-opt** indicates the average cpu time which the optimal schedule are obtained

Table 8 Computational results of the YN benchmarks

			Our HGA		Balas, E.(1998)		
Prob.	Size	LB	Best	Average	SB-GLS2	SB-GLS5	TSAB′
YN1	20×20	888 (826)	889	906.4	896	893	897
YN2	20×20	909 (861)	917	923.9	918	911	924
YN3	20×20	893 (827)	898	905.8	901	897	901
YN4	20×20	968 (918)	974	984.7	990	977	988

Acknowledgements This paper is supported by the National Basic Research Program of China (under the Grant No. 2005CB724107) and the National Natural Science Foundation, China (under the Grants No. 70271053, 50305008). The authors would like to thank the referees for their helpful comments and suggestions.

References

1. Davis L (1985) Job shop scheduling with genetic algorithms. In: Proceedings of the International Conference on Genetic Algorithms and their Applications, Hillsdale, pp 136–149
2. Bierwirth C (1995) A generalized permutation approach to job shop scheduling with genetic algorithms. OR Spektrum 17:87–92
3. Croce FD, Tadei R, Volta G (1995) A genetic algorithm for the job shop problem. Comput Oper Res 22(1):15–24
4. Laarhoven PV, Aarts E, Lenstra JK (1992) Job shop scheduling by simulated annealing. Oper Res 40(1):113–125
5. Taillard ED (1994) Parallel taboo search techniques for the job-shop scheduling problem. ORSA J Comput 6(2):108–117
6. Nowicki E, Smutnicki C (1996) A fast taboo search algorithm for the job shop problem. Manag Sci 42(6):797–813
7. Jain AS, Meeran S (1999) Deterministic job-shop scheduling: Past, present and future. Eur J Oper Res 113:390–434
8. Blazewicz J, Domschke W, Pesch E (1996) The job shop scheduling problem: Conventional and new solution techniques. Eur J Oper Res 93:1–33
9. Leung Y, Gao Y, Xu ZB (1997) Degree of population diversity -a perspective on premature convergence in genetic algorithms and its Markov-chain analysis. IEEE Trans Neural Netw 8(5):1165–1176
10. Cheng R, Gen M, Tsujimura Y (1999) A tutorial survey of job shop scheduling problems using genetic algorithms, part II: hybrid genetic search strategies. Comput Ind Eng 36:343–364
11. Pinedo M (2002) Scheduling Theory, Algorithms, and System, 2nd edn. Prentice Hall, Upper Saddle River, New Jersey
12. Gen M, Tsujimura Y, Kubota E (1995) Solving job-shop scheduling problems by genetic algorithm. In: Proceedings of the 1995 IEEE International Conference on Systems, Man, and Cybernetics, Vancouver, pp 1577–1582
13. Shi GY, IIMA H, Sannomiya N (1996) A new encoding scheme for job shop problems by genetic algorithm. In: Proceedings of the 35th Conference on Decision and Control, Kobe, Japan, pp 4395–4400
14. Baker KR (1974) Introduction to sequencing and scheduling. Wisely, NewYork
15. Cheng R, Gen M, Tsujimura Y (1996) A tutorial survey of job-shop scheduling problems using genetic algorithms -I. representation. Comput Ind Eng 30(9):83–97
16. Jain AS, Rangaswamy B, Meeran S (2000) New and "stronger" job-shop neighborhoods: A focus on the method of Nowicki and Smutnicki (1996). Journal of Heuristics 6:457–480
17. Gonçalves JF and et al (2002) A hybrid genetic algorithm for the job sShop scheduling problem. AT&T Labs Research Technical Report, doi:optimization-online.org/DB_FILE/2002/09/538.pdf
18. Kobayashi S, Ono I, Yamamura M (1995) An efficient genetic algorithm for job shop scheduling problems. In: Proceedings of the 6th International Conference on Genetic Algorithms, pp 506–511
19. Ono I, Kobayashi S (1998) A genetic algorithm taking account of characteristics preservation for job shop scheduling problems. In: Proceedings of the 5th Conference on Intelligent Autonomous Systems, pp 711–718
20. Aiex RM, Binato S, Resende MGC (2003) Parallel GRASP with path-relinking for job shop scheduling. Parallel Comput 29:393–430
21. Pezzella F, Merelli E (2000) A tabu search method guided by shifting bottleneck for the job shop scheduling problem. Eur J Oper Res 120:297–310
22. Balas E, Vazacopoulos A (1998) Guided local search with shifting bottleneck for job-shop scheduling. Manag Sci 44(2):262–275