Evolving Systems (2011) 2:119–129
DOI 10.1007/s12530-010-9022-x

ORIGINAL PAPER

Improved precedence preservation crossover for multi-objective job shop scheduling problem

Kazi Shah Nawaz Ripon · N. H. Siddique · Jim Torresen

Received: 16 December 2009 / Accepted: 12 November 2010 / Published online: 18 December 2010

Abstract Over the last three decades, a great deal of research has been focused on solving the job-shop scheduling problem (JSSP). Researchers have emerged with a wide variety of approaches to solve this stubborn problem. Recently much effort has been concentrated on evolutionary techniques to search for the near-optimal solutions optimizing multiple criteria simultaneously. The choice of crossover operator is very important in the aspect of genetic algorithms (GA), and consequently a wide range of crossover operators have been proposed for JSSP. Most of them represent a solution by a chromosome containing the sequence of all the operations and decode the chromosome to a real schedule from the first gene to the last gene. However, these methods introduce high redundancy at the tail of the chromosome. In this paper, we address this problem in case of precedence preservation crossover (PPX) which is regarded as one of the better crossover operators and propose an improved version, termed as improved precedence preservation crossover (IPPX). Experimental results reveal that our proposed approach finds the near-optimal solutions by optimizing multiple criteria simultaneously with better results and also reduces the execution time significantly.

K. S. N. Ripon (✉) · J. Torresen
Department of Informatics, University of Oslo,
P.O. Box 1080, 0316 Blindern, Oslo, Norway
e-mail: ksripon@ifi.uio.no

J. Torresen
e-mail: jimtoer@ifi.uio.no

N. H. Siddique
School of Computing and Intelligent Systems,
University of Ulster, Northland Road,
Londonderry BT48 7JL, UK
e-mail: nh.siddique@ulster.ac.uk
URL: http://www.infm.ulst.ac.uk/~siddique

Keywords Precedence preservation crossover (PPX) · Job-shop scheduling problem (JSSP) · Multi-objective evolutionary optimization · Pareto optimal front

1 Introduction

In job shop scheduling problem (JSSP), the objective is to allocate resources in a way, such that a number of tasks can be completed cost-effectively within a given set of constraints. JSSP is one of the most widely studied problems in computer science, which has great importance to manufacturing industries with the objective to minimise the production cost. Classical JSSP can be described as scheduling n different jobs on m machines. The m machines are identical, and the n nonpreemptable jobs are all independent. Job i has the size $J_i > 0, 1 \leq i \leq n$, which is to be processed on a set of m machines (M_r), $1 \leq r \leq m$. Each job has a technological sequence of machines to be processed. The jobs arrive one by one, and each job must be immediately and irrevocably scheduled without knowledge of later jobs. The size of a job is known on arrival, and the jobs are executed only after the scheduling is completed. The processing of job J_i on machine M_r is called the operation $O_{ir.}$ Operation O_{ir} requires the exclusive use of M_r for an uninterrupted duration P_{ir}, its processing time. A schedule is a set of completion time for each operation C_{ir} that satisfies these constraints. Thus, JSSP can be considered as a searching or optimization problem, where the goal is to find the best possible schedule. The classical JSSP is one of the most challenging combinatorial optimization problems, mainly because of two reasons. Firstly, even for the special case of $m = 2$, JSSP is NP-hard. Secondly, one has to deal with tricky search space as neighboring points may represent very different solutions.

Due to expansion in manufacturing industry and industrial automation, JSSP has become a practical problem in the industry and started getting a lot of attention from the research community. The difficulty of the general JSSP is that it makes rather hard for conventional search-based methods to find a set of optimal schedules within polynomial time. Deterministic scheduling methods like the branch-and-bound method (Bucker et al. 1994) or the dynamic programming (Peter et al. 1999) are always computationally expensive for an optimum solution when searching a large search space. To combat the increasing complexity, it has been suggested by many researchers for near-optimal solutions instead. Such near-optimal solutions can only be endowed by stochastic search techniques such as evolutionary algorithms (EAs) (Ripon 2007). Davis proposed the first GA-based approach to the solution of scheduling problems in 1985 (Davis 1985). This paper was instructive to the application of GA on JSSP. Since then, GA has been applied with increasing frequency to JSSP. In contrast to smart heuristics such as simulated annealing (Kirkpatrick et al. 1985) and tabu-search (Glover 1989) which are local search techniques and use a generate-and-test search manipulating one feasible solution based on physical rather than a biological analogy, the GA utilizes a population of solutions in its search, giving it more resistance to premature convergence on local minima.

Surprisingly most of the researches in JSSP have focused mainly on a single objective, and predominantly optimization of the makespan, which is defined as the time interval between start of first operation and completion of last operation. However, real world scheduling problems naturally involve multiple objectives. In general, minimization of the total makespan is often used as the optimization criterion in JSSP. However, tardiness, flow time, lateness, earliness are also the important criteria in JSSP. Inherently, real-life scheduling problems are multi-objective by nature and the final schedule must consider various objectives simultaneously. Consequently, scheduling falls into the category of multi-objective optimization problem (MOOP). In such MOOPs, there is no single optimal solution, rather there is a set of alternative solutions. These solutions, namely Pareto-optimal solutions (Deb 2001; Zitzler et al. 2000; Veldhuizen and Lamont 2000), are optimal in the wider sense that no other solutions in the search space are superior when all the objectives are considered. In a Pareto-optimal set, any two solutions of this set do not dominate each other and one or more members of this set dominate all other solutions in the search space excluding this set. Based on the principle of multi-objective optimization, obtaining an optimal solution that satisfies all objectives is almost impossible. It is mainly for the conflicting nature of objectives, where improving one objective may only be achieved when worsening another objective. Accordingly, it is desirable to obtain as many different Pareto-optimal solutions as possible, which should be converged to, and diverse along the Pareto-optimal front with respect to multiple criteria.

Nagar et al. (1995) provided a good review on multi-objective production scheduling. This survey mostly covered researches conducted up to early 1990s based on a complete classification scheme. In those researches, the solution methodology primarily comprises of implicit enumeration techniques such as branch and bound, dynamic programming and trade-off curves. They specifically mentioned that they did not come across any paper that used techniques like simulated annealing, tabu-search and genetic algorithm. This has also been confirmed in a survey by Lei (2009) that conventional techniques are the main approaches to multi- and bi-objective scheduling problems before 1995. Lei provided extensive survey coverage on the literature of multi-objective production scheduling, where he first classified scheduling problems into deterministic and non-deterministic classes. The deterministic scheduling problems are categorized into four types. The author surveyed more than 90 papers published between 1995 and 2008, out of which about 40 papers adopted evolutionary algorithm (EA) and GA as optimization methods. It is evident from the two surveys that only in the current decade a great deal of attention has been focused on solving JSSP using EAs (Ripon 2007; Bierwirth et al. 1996; Jensen 2003; Syswerda 1991; Bierwirth 1995; Yamada and Nakano 1992; Song et al. 1999; Chan et al. 2008; Ombuki and Ventresca 2004).

The fundamental concept of JSSP can be thought of as an ordering problem. A schedule is a representational issue which resembles to a permutation problem. A permutation problem can generally be formulated in the following way. A set of n operations (tasks) with known processing times has to be scheduled on m machines (resources). A group of m operations forms a complex called a job. Altogether N jobs are defined within the set of operations, i.e. the number of operations is the Cartesian product of the number of jobs and machines defined by $O = N \times m$, where O is the number of operations. Partitioned permutation provides the bedrock for the application of EA to many combinatorial optimisation problems (Bierwirth et al. 1996) and thus serves as chromosomes in GA. The known snare of this permutation chromosome is the crossover operator, which may not safeguard the semantical properties of the underlying problem. Therefore, one of the central issues in the use of GAs for the JSSP is an efficient characterization of the strength of crossover operator. This is perhaps due to crossover operator's most exploratory power in an GA. Eshelman et al. investigated the exploratory power of crossover operator (Eshelman et al. 1989). Bierwirth developed repeating permutation representation

while Mattfeld implemented a number of crossover operations (Bierwirth et al. 1996; Bierwirth 1995; Mattfeld 1996). A large number of crossover operators have been proposed in the literature such as generalized partially mapped crossover (GPMX) (Bierwirth et al. 1996), generalized order crossover (GOX) (Bierwirth 1995), precedence preservation crossover (PPX) (Bierwirth et al. 1996; Mattfeld 1996) and such others, mainly due to the need for designing specialist crossover operations to use with permutation representations. The details of crossover operators specifically designed for ordering applications can be found in (Bierwirth et al. 1996; Jensen 2003). In all of these permutation based crossover techniques, there exist redundancies at the tail of a chromosome. A schedule, decoded from a chromosome sequence of redundant genes, has no effect on the final solution. It also imposes additional time complexity. Therefore, the objective of the current paper is to improve the crossover technique and propose the improved precedence preservation crossover (IPPX) to be used in the JSSP, which eventually resolves this issue and also reduces the execution time. In an attempt to address multiple objectives simultaneously in this work, we apply makespan and mean-flow time as the objectives and present the schedules as a set of Pareto-optimal solutions.

The remainder of the paper is organized as follows. Section 2 describes various crossover operators. Section 3 presents the proposed approach for the improvement of PPX. Experimental results on the performance of the proposed method (IPPX) and comparisons with others approaches are demonstrated in Sect. 4. Finally, some concluding remarks on the method are made in Sect. 5.

2 Crossover in permutation representation

In crossover, also known as recombination, information is exchanged among the chromosomes (or individuals) present in the mating pool to create new chromosomes. Permutation-based representations present particular difficulties for the design of crossover operators, since it is not generally possible simply to exchange strings of genes between selected parents and still maintain the permutation properties. The crossover operator has to comply with the semantic meaning (properties) of the chromosome representation meaning to combine building blocks to form larger building blocks which share the phenotypical traits of the smaller building blocks. Two genes may articulate meaningful information if they appear side by side (relative order). They may even articulate information if one gene precedes the other gene in the chromosome (absolute order) regardless of how many genes lay between. Syswerda (1991) inferred that the order as well as the position of genes in the permutation is meaningful. An excellent theoretical analysis on order-based crossover has been provided by Wroblewski (1996). To be more precise, we expect the absolute order to be of particular interest because it directly expresses precedence relation among the operations in a schedule. A number of specialized crossover operators have been designed for permutations. These aim at exchanging information as much as possible which especially held in common in both parents.

GPMX (Bierwirth et al. 1996) and GOX (Bierwirth 1995) assemble one offspring from two parent chromosomes (donator and receiver). In both techniques a substring is chosen from the donating chromosome. Then all genes of the substring are deleted with respect to their index of occurrence in the receiving chromosome. The GOX operator was first presented by Bierwirth (1995). GOX implants the substring into the receiver at the position where the first gene of the substring has occurred (before deletion) in the receiver. Unlike GOX and GPMX implants the substring at the position where it occurs in the donator. Figure 1 illustrates these two different crossover techniques.

PPX (Bierwirth et al. 1996; Jensen 2003) perfectly respects the absolute order of genes in parental chromosomes. First the offspring chromosome is initialized empty. Then a vector of length n is randomly filled with elements of the set $\{0, 1\}$. This vector defines the order in which genes are drawn from parent 1 and parent 2, respectively. After a gene is drawn from one parent and deleted from the other one, it is appended to the offspring chromosome. This step is repeated until both parent chromosomes become empty and the offspring contains all genes involved. Figure 2 depicts a typical PPX operation.

In order to apply PPX in a uniform crossover fashion the choices may alternatively change at random. However, the absolute order between any of two genes in the offspring has its origin in at least one of the parental chromosomes. In short, GOX passes on the relative order of genes, GPMX tends to pass on positions of genes by respecting the ordering to some extent, and PPX respects the absolute order of genes resulting in a perfect preservation of precedence relations among genes.

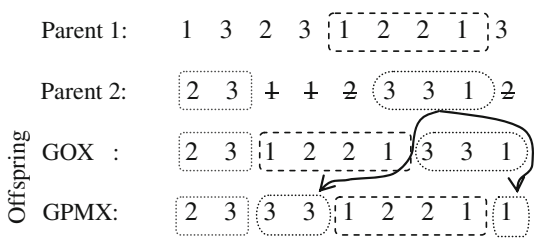

Fig. 1 GPMX in comparison to GOX for a typical 3 × 3 JSSP

Parent 1	1	3	2	2	1	3	3	2	1
Parent 2	**2**	**1**	**3**	**3**	**1**	**2**	**3**	**1**	**2**
Selection Vector	1	1	0	0	0	1	1	1	0
PPX Offspring	**2**	**1**	3	2	1	**3**	**3**	**1**	2

Fig. 2 PPX for a typical 3 × 3 JSSP

2.1 Superiority of PPX over GPMX and GOX

Bierwirth (1995) discussed about the heredity of various schedule characteristics. He emphasized on precedence relations among operations in order to explain the representational issue of absolute order for the JSSP. In this approach, they map the permutation representation into a string of 0/1-decision expressing the absolute order of any of two operations to be processed on the same machine. This mapping leads to the binary representation proposed by Yamada and Nakano (1992). The binary mapping is very useful to measure the differences and similarities of schedules on a phenotypical level by means of the hamming distance. Crossover performed by GPMX, GOX or PPX results in one offspring. Ideally the offspring inherits one half of the characteristics from each parent. Table 1 describes the phenotypical distance $d(p_1, p_2)$ between two randomly generated parents p_1 and p_2. The distances of the offspring, denoted as $d(o, p_i)$ to both parents for 1,000 calls to PPX, GPMX and GOX are also shown in the table.

The average normalized Hamming distance between two arbitrary solutions is 0.273. For all three crossover operators, we observe that $d(o, p_1) \approx d(o, p_2)$. This verifies all operators to pass on the same portion of parental information. In case of PPX, $d(o, p_1) + d(o, p_2) = d(p_1, p_2)$ holds. It means PPX passes on precedence relations perfectly in comparison to that of GOX and GPMX.

2.2 Redundancy problem of PPX

Most of the GA approaches for JSSP represent a solution by a chromosome containing the sequence of all the operations and decode the chromosome to a real schedule from the first gene to the last gene. According to Song et al. (1999), there are three common problems for these approaches. Firstly, it introduces high redundancy at the tail of the chromosome. Secondly, there exists little

Table 1 Phenotypical preservation of crossover

Operator	$d(p_1, p_2)$	$d(o, p_1)$	$d(o, p_2)$	$d(o, p_1) + d(o, p_2)$
PPX	0.273	0.137	0.136	0.273
GPMX	0.273	0.141	0.139	0.280
GOX	0.273	0.150	0.152	0.302

Parent 1	1	3	2	2	1	3	3	2	1
Parent 2	**2**	**1**	**3**	**3**	**1**	**2**	**3**	**1**	**2**
Selection Vector	1	1	0	0	0	1	1	1	0
PPX Offspring	**2**	**1**	3	2	1	**3**			

Fig. 3 Redundancies at the tail of the chromosome

significance of rear genes on the overall schedule quality. And finally, GA operators applied on the real part of the chromosome are less likely to create genetically improved good offspring, i.e., most likely a waste of evolution time.

It implies that the rear genes of a chromosome are not significant enough and consequently have less impact on overall schedule. The execution time of the last operation of a job does not reduce the waiting time of other operations of that job. As a result, it is nothing but waste of evolution time in case of applying the GA operators (crossover, mutation) on these genes. As for example, let us consider the previous 3 × 3 JSSP (Fig. 2) in the following Fig. 3.

Pointing at the last three genes (3, 2, and 1), we can find that all of these operations are the last operation of respective job. Consequently job 3, in the last four bits, we find the last operation for job 3. Now finishing the last operation of job 3 at an earlier time does not reduce the total completion time for job 3. So, it is redundant to manipulate these last genes. Most importantly, in case of large number of jobs and operations (Suppose 100 × 100 JSSP), this redundancy increases and makes unnecessary delay to deliver an optimal schedule.

3 Proposed improved PPX (IPPX) approach

3.1 Chromosome representation

Chromosome representations for JSSP using GA can be performed by two basic encoding approaches: direct and indirect (Chan et al. 2008). The direct approach encodes a schedule as a chromosome and the genetic operators are used to evolve these chromosomes into better schedules. In indirect representation, the chromosome encodes a sequence of decision preferences, like simple ordering of jobs in a machine or any heuristic rules, and a schedule builder is required to decode the chromosome into a schedule.

In this work, we applied indirect representation of chromosomes incorporated with a schedule builder. This is because, applying simple genetic operators on direct representation string often results in infeasible schedule solutions. Indirect chromosome representation is implemented with an un-partitioned operation-based representation where each job integer is repeated m times (m is the number of machines), and it is known as "permutation with

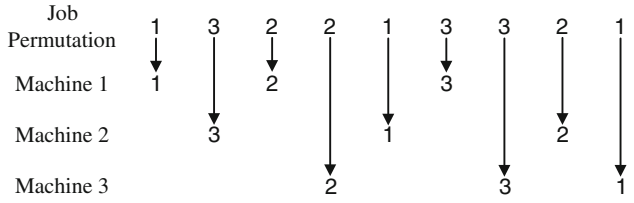

Fig. 4 Permutation with repetition approach for a 3 × 3 JSSP

repetition" (Bierwirth 1995) in mathematics. By scanning the permutation 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 as depicted in Fig. 4. In this representation, any individual can be decoded into a feasible schedule, but two or more different individuals may be translated into an identical schedule. The advantage of such a scheme is that it requires a very simple schedule builder because all the generated schedules are legal.

3.2 Reduction of redundancy

As discussed earlier in Sect. 2, the permutation based crossover techniques exhibit redundancy at the tail of chromosomes. To reduce such tail redundancy, we perform PPX for (*N*–*M*) genes from the two parent chromosome and choose the last M genes by using a simple heuristic method, where M follows the constraint $O \leq M \leq \frac{N}{3}$. Here, O is the number of operation and N is the total number of genes.

Suppose that $P1$ and $P2$ are the two chromosome (3 × 3 JSSP) using this method as shown in Fig. 5. Here, last ($M = 3$) genes are not taken under crossover operation. Instead, we choose the remaining M genes using the heuristic method. In this figure, the genes goes through crossover are shown by shadow and deleted genes in both chromosomes are represented by bold numbers. In order to adjust the total length of the chromosome, we apply a heuristic method for the rest of the genes in the offspring chromosome. In this approach, we always expect the last operations of a job is in the last M bits. From the two parent chromosomes, we choose that chromosome in which there is more number of last operations before the occurrence of a non-last operation in the remaining M bits. If this value is the same for both the parent chromosomes, then genes are taken from that parent who has a better fitness value. For example, for a 5 × 3 JSSP and $M = 4$, the last M genes of the chromosome are shown in Fig. 6.

Here, we select the last four genes of the offspring from $P1$ as it contains more number of last operations before the occurrence of a non-last operation. If we observe closely, we see that in $P0$ there are two operations before the occurrence of a non-last operation (marked by a circle) whereas in $P1$ there are three operations before the occurrence of a non-last operation. As we always keep track of the operation number of a job, this method does not require additional efforts which makes it very simple, effective and less complex. Algorithm 1 presents the pseudo code for the proposed IPPX.

Algorithm 1. Improved Precedence Preservation Crossover (IPPX)

Here we assume, V_x → Parent0
V_y → Parent1
NJ → Number of jobs and
NM→ Number of machines

1. Let, M := (NJ * NM)/10; (Initialize value of last M gene)
2. **for** the first N-M genes do //PPX for N-m gene
   a. Initialize selection vector to select a parent
   b. Change the value of selection vector after each four occurrence of 0 and 1.
   c. **If** V_x is selected then,
      Search for first available gene of V_x
      Delete this gene from V_y
      Else
      Search for first available gene of V_y
      Delete this gene from V_x
3. Calculate the values gV_x and gV_y where
   gV_x:= number of last operations in V_x before a non-last operation and
   gV_y:= number of last operations in V_y before a non-last operation
4. Choose the last M genes from the better of gV_x and gV_y.
   In case both are same, choose each of the last M genes from the fitter of the V_x and V_y

---

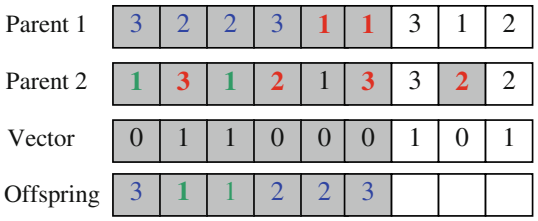

Fig. 5 Crossover for (N–M) genes

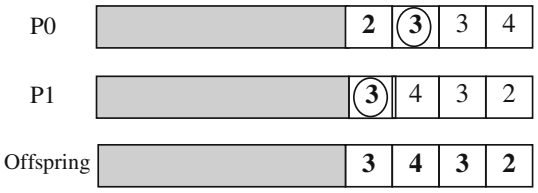

Fig. 6 Selection of the last M genes by heuristics method

3.3 Mutation

We use swapping mutation technique in our approach. Here any two genes within (N–M) genes of the offspring chromosome are swapped. And the last *M* genes are left as they are, as shown in Fig. 7.

3.4 Efficiency of IPPX method

Figure 8 represents the flowchart of the proposed GA based method for JSSP using IPPX. In our approach, we apply the GA operators on less number of genes in a chromosome. As the number of operation required for these operators are less than the usual ones, it saves execution time. In a generation, if the crossover rate is 0.8 and population size is 100, the number of crossover will be 80. So, for 1,000 generations there will be 80,000 crossovers. As the time required to complete a crossover is reduced, the total execution time will be reduced significantly. Thus, this approach is very effective when the population, generations, jobs and operations are huge in number. Real world scheduling problems are generally too large to find their optimal schedule in a reasonable amount of time. The experimental results justify that our approach is able to achieve satisfactory and sometimes better than the average results in comparison with the existing methods.

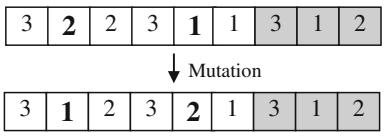

Fig. 7 Mutation

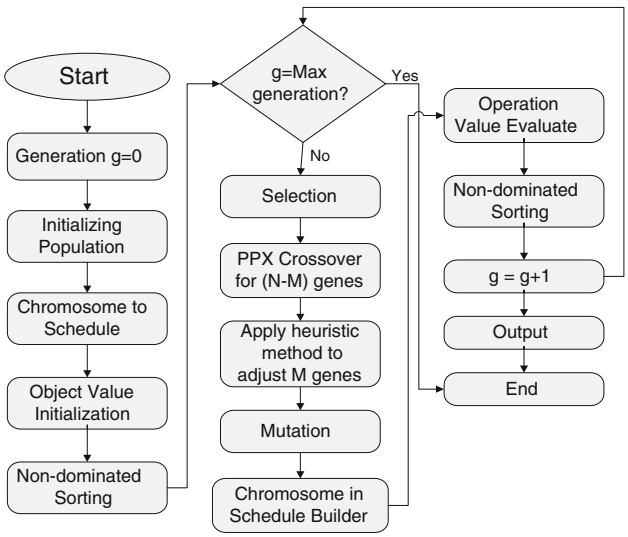

Fig. 8 Flowchart of the proposed method

4 Experimental result and analysis

4.1 Experimental setup

The experiments are conducted using 100 chromosomes and 150 generations. The probabilities of crossover and mutation are 0.9 and 0.3, respectively. Using the same setting, each problem is tested for 30 times with different seeds, and the best and average solutions are recorded. Then all the final generations were combined and a non-dominated sorting is performed to constitute the final non-dominated solutions. To justify the efficiency of the proposed crossover method (IPPX) (as shown in Algorithm 1) the results are compared to existing well-known crossover methods GPMX, GOX, and PPX in the framework for a multi-objective genetic algorithm. In this work, we use the non-dominated sorting genetic algorithm (NSGAII) (Deb et al. 2002) as the multi-objective GA framework. Additionally, to justify the efficiency of the proposed IPPX-based JSSP approach, we present a comparison with other GA-based JSSP approaches in single objective context. In addition, we compare the performance of the IPPX-based approach with the performance of other existing GA-based JSSP approaches in multi-objective context.

4.2 Benchmark problems

To evaluate how IPPX performs with respect to solution quality, we run the algorithm on various benchmark data.

Table 2 Benchmark problems with their lower bounds

Instance data	Number of jobs	Number of machines	Lower bound (makespan)
mt06	6	6	55
mt10	10	10	930
mt20	20	5	1,165
la21	15	10	1,040
la24	15	10	935
la25	15	10	977
la27	20	10	1,235

The first three well-known benchmark problems, known as mt06, mt10 and mt20, formulated by Muth and Thompson (1963) are commonly used as test beds to measure the effectiveness of a certain method. The mt10 and mt20 problems have been a good computational challenge for a long time. However, it is no longer a computational challenge now. Indeed, the mt10 problem has been referred as "notorious", because it remained unsolved for over 20 years. The mt20 problem has also been considered as quite difficult. Applegate and Cook (1991) propose a set of benchmark problems called the "ten tough problems" as a more difficult computational challenge than the mt10 problem, by collecting difficult problems from literature, some of which still remain unsolved (Applegate and Cook 1991). Among these, the la problems are parts of 40 problems la01–la40 originated from (Lawrence 1984). The problem data and the lower bounds information are taken from the OR-library. Table 2 presents the problem size and the best known lower bounds for makespan for the problems that have been used in this study.

4.3 Objective functions

The quality criteria for a good schedule are: maximum tardiness, average tardiness, weighted flow time, weighted lateness, weighted tardiness, weighted number of tardy jobs and weighted earliness plus weighted tardiness (Fang et al. 1996). In the present work, the makespan has been considered as the first objective. The mean flow time, as the second objective, continues to be very important since it assists to select the appropriate one when many algorithms proposed in the literature have reached the same makespan for many instances. The two objective functions makespan and mean flow-time are defined by Eqs. 1 and 2, respectively. Note that, both of these objectives are subject to minimization.

$$\text{Makespan} = \min\{\max\lfloor C \rfloor\} \tag{1}$$

$$\text{Mean flow-time} = \frac{1}{n}\sum_{i=1}^{n} C_i \tag{2}$$

where C_i is the completion time of job i and n is the number of jobs.

4.4 Analysis of results

4.4.1 Single objective context

Until now, almost all JSSP algorithms try to optimize single criteria only (mainly minimization of the makespan). Therefore, to evaluate our proposed IPPX based algorithm as an evolutionary approach, we first compared the makespan obtained by our approach with the existing GA-based approaches to justify its capability to optimize makespan. We then demonstrated its performance as a multi-objective evolutionary JSSP algorithm by optimizing makespan and mean flow-time simultaneously. Note that, for both cases we have used the same results achieved by our approach.

The values provided in Table 3 show the makespan of the best schedules obtained in case of mt06, mt10 and mt20 problems by some GA-based scheduling algorithms. The column labeled sGA is based on the GA using the simple mutation that swaps the positions of two random jobs (Ombuki and Ventresca 2004), whereas SGA is based on simple GA proposed by Yamada and Nakano (1992, 1997). LSGA and GTGA indicate the GA-based job-shop scheduling algorithm incorporating local search and the GA based on GT crossover (Yamada and Nakano 1992), respectively. From this Table, it is ample clear that the proposed method is capable of producing near-optimal schedules in most of the cases. For mt06 and mt10 problems, it achieved the lower bound. In the case of mt20, JGGA and the proposed IPPX method cannot obtain the lower bound, however it outperforms the other contestant algorithms. Although some other heuristic scheduling algorithms may achieve the same result, it should be noted that the main objective of the proposed method is the use of new crossover technique for multi-objective JSSP. The comparisons with single objective JSSP is only to justify the capability of the proposed method in the case of minimizing makespan.

4.4.2 Multi-objective context

Multi-objective optimization differs from single objective optimization in many ways. For two or more objectives, each objective corresponds to a different optimal solution; however none of the trade-off solutions is optimal with

Table 3 Comparison of different operators in single objective context

Instance data	Lower bound	sGA	GTGA	LSGA	SGA	JGGA	IPPX
mt06	55	55	55	55	55	55	**55**
mt10	930	994	930	976	965	930	**930**
mt20	1,165	1,247	1,184	1,209	1,215	1,180	**1,180**

The bold values indicate the best found results

respect to all objectives. Table 4 shows the comparison of our proposed crossover-based method with some existing crossover-based methods in multi-objective context. The results shown in the Table indicate that, the IPPX based approach clearly outperforms the other crossover-based methods in multi-objective context in terms of both makespan and mean flow-time. From the results, it is also interesting to note that the performance of IPPX-based approach is superior to other in achieving better makespan values (which is considered the primary objective, if we consider single objective optimization) than the competing methods.

Figure 9 graphically displays the convergence behavior of solutions obtained for la24 problem using different crossover operators PPX, GPMX, GOX and our proposed IPPX, averaging 30 different starting random seeds. In each case, all methods start with identical randomly picked initial solutions. Also it is able to populate the Pareto front

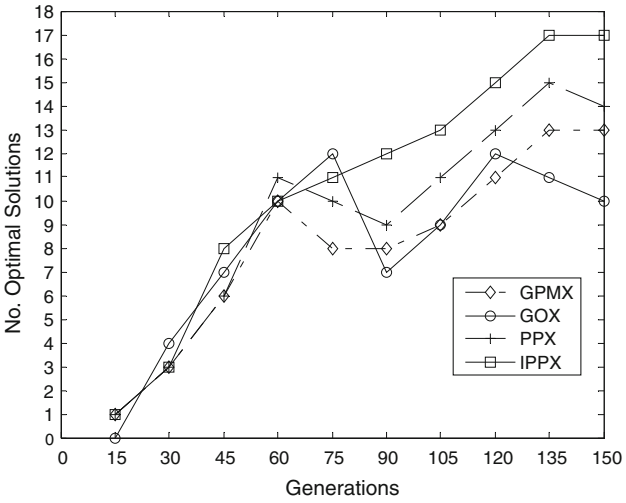

Fig. 9 Comparison of different operators in finding Pareto optimal solutions for la24 problem

Table 4 Comparison with other operators in multi-objective context

Instances	Crossover operators	Makespan		Mean flow time	
		Best	Average	Best	Average
Mt06	GPMX	55	59.73	53	53.046
	GOX	55	59.81	53	52.51
	PPX	55	59.71	53	53.918
	IPPX	**55**	**59.35**	**50**	**50.086**
Mt10	GPMX	945	1,035.13	814	875.89
	GOX	943	1,030.01	812	880.13
	PPX	942	1,016.14	830	871.815
	IPPX	**930**	**1,013.16**	**822**	**830**
Mt20	GPMX	1,188	1,266.60	818	901.089
	GOX	1,196	1,325.80	820	822.043
	PPX	1,188	1,256.90	810	915
	IPPX	**1,180**	**1,216.39**	**767**	**768.90**
La21	GPMX	1,062	1,112.43	898	903.173
	GOX	1,058	1,116.543	905	908.231
	PPX	1,058	1,123.56	908	914.184
	IPPX	**1,046**	**1,103.68**	**913**	**919.66**
La24	GPMX	968	972.054	817	817.34
	GOX	968	986.61	819	829.59
	PPX	966	998.30	829	833.006
	IPPX	**935**	**975.88**	**829**	**867.73**
La25	GPMX	998	1,065.18	809	821.30
	GOX	1,002	1,071.09	811	856.13
	PPX	993	1,045.701	803	811.861
	IPPX	**982**	**1,033.314**	**773**	**823.27**
La27	GPMX	1,257	1,395.31	1,088	1,099.91
	GOX	1,258	1,401.03	1,091	1,116.81
	PPX	1,255	1,398.10	1,123	1,133.7
	IPPX	**1,243**	**1,384.66**	**1,111**	**1,111.25**

The bold values indicate the best found results

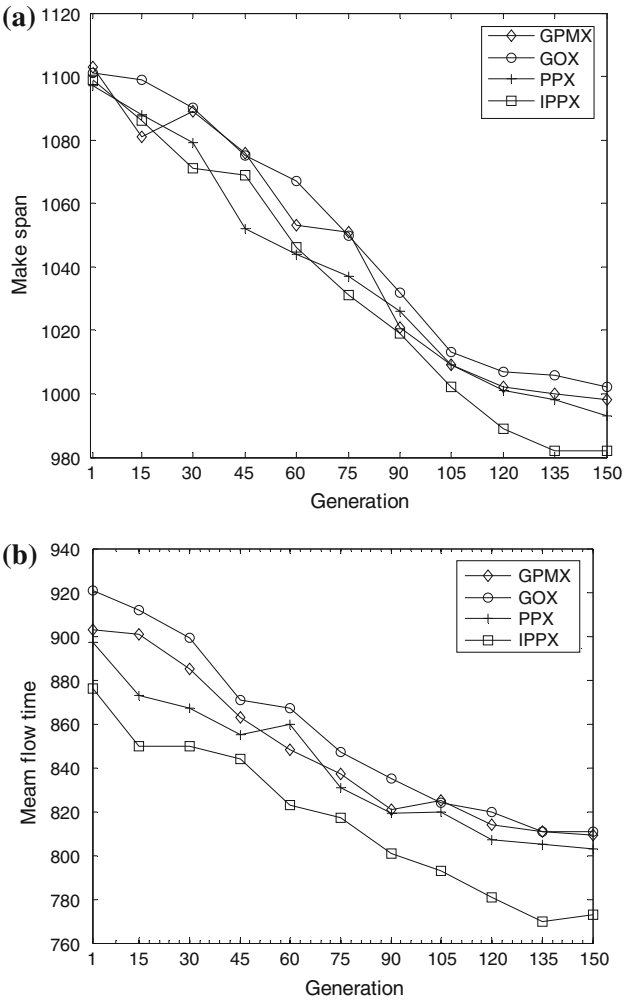

Fig. 10 **a** Makespan and **b** mean-flow time over generations for la25 problem

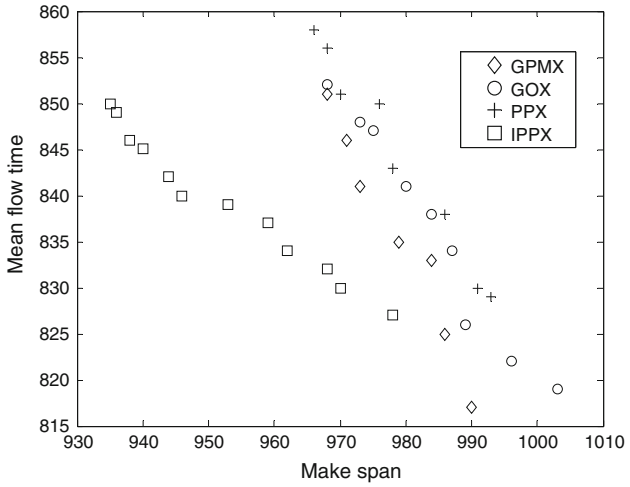

Fig. 11 Pareto optimal front obtained by IPPX, PPX, GOX and GPMX operators

faster than the existing approaches. As compared to other crossover-based methods, the IPPX-based method produced more non-dominated solutions with generations.

Figure 10 demonstrates the optimization behavior of the methods using the crossover operators over generations for la25 problem. The Fig. 10a presents the comparative observation on the optimization of the first objective makespan over generation. Though all PPX, GPMX, GOX and IPPX can optimize over generations successfully, IPPX converges faster than others after generation 90. Figure 10b presents the comparative observation for the second objective mean flow-time over generation. Overall PPX, GPMX, GOX and IPPX have steady convergence over the generations but IPPX shows much better convergence than others from the very first generation. In addition, the proposed approach clearly optimizes both of the objectives over the generations. From the figures, it can be found that from generation 1 to generation 150, the proposed method is able to optimize both of the makespan and mean flow-time.

The aim of multi-objective optimization problems is to find all possible trade-offs among multiple objectives. To illustrate the convergence and diversity of the solutions, a comparative Pareto optimal solution to la25 problem obtained by PPX, GPMX, GOX and IPPX is shown in Fig. 11. From the figure, it can be observed that the IPPX-based approach can find a set of well diverse solutions along the Pareto front compared to its peers. For this reason, it is capable of finding extreme solutions and produce more suitable non-dominated solutions for the decision makers to choose the appropriate one based on different situations. In general, the proposed IPPX-based multi-objective JSSP algorithm is able to achieve consistent near-optimal solutions which are both well spread and converged.

It is clear that there are two distinct goals in MOEA: converge as close as possible to the Pareto-optimal front, and maintain as diverse a set of solutions over the Pareto front as possible. An MOEA will be considered good only if both of these two goals are satisfied simultaneously. Hypervolume (Zitzler et al. 2000; Veldhuizen and Lamont 2000) provides a qualitative measure of convergence as well as diversity. This

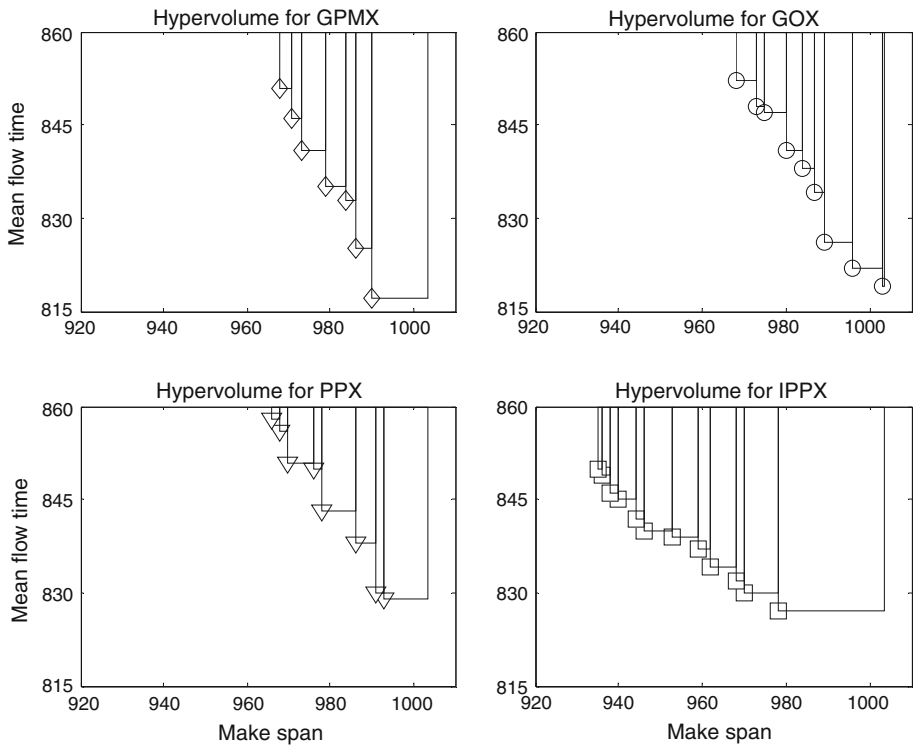

Fig. 12 Hypervolume enclosed by the non-dominated solutions by different operators

metric calculates the volume in the objective space covered by members of the Pareto-optimal set with a reference point. The reference point can simply be found by constructing a vector of worst objective function values. Obviously, maximization of the volume means the minimization of all the objectives. Figure 12 shows the hypervolume enclosed by the non-dominated solutions by the operators GPMX, GOX, PPX and IPPX for la25 problem. The chosen reference point was $w^* = (1003.5, 860)$.

The values of hypervolumes for this problem are 1068.5 for GPMX, 847.5 for GOX, 717.5 for PPX and 1784.5 for IPPX operators. Zitzler (1999) defined a metric, called maximum spread, measuring the length of the diagonal of a hyperbox formed by the extreme function values observed in the non-dominated set. The maximum spreads of the Pareto-front solutions of the corresponding operators were 40.4969 for GPMX, 48.1041 for GOX, 39.6232 for PPX, and 48.7647 for IPPX. Though the maximum spread does not reveal true distribution of the solutions, but inspecting the Fig. 12 we can see all non-dominated solutions are nearly uniformly spaced. It is evident from the above performance metrics that IPPX outperforms its peers.

4.5 Computational time

As we remove redundancy at the tail of a chromosome while applying IPPX, it finds near optimal solutions in less time duration than that of IPPX. Table 5 depicts this scenario. If we observe this table closely, we can find that the time difference increases with problem domain. In la27 (20 × 10) time difference between PPX and our approach is 4 s. It can also be noted that the time difference increases with generations in the same problem domain. In short, the larger the generation and the problem domain, the more the time difference among the proposed and the existing methods.

Table 5 Computation time by different operators

Problem domain	Generation	GPMX (s)	GOX (s)	PPX (s)	IPPX (s)
Mt10 (10 × 10)	1,000	18	18	18	**17**
	2,000	**34**	35	35	**34**
	3,000	53	53	53	**51**
	4,000	70	71	70	**68**
Mt20 (20 × 5)	1,000	**18**	**18**	**18**	**18**
	2,000	35	36	36	**34**
	3,000	54	54	54	**51**
	4,000	71	72	71	**68**
La27 (20 × 10)	1,000	**34**	**34**	**34**	**34**
	2,000	69	69	68	**67**
	3,000	101	101	100	**99**
	4,000	134	135	134	**130**
La21 (15 × 10)	1,000	**26**	**26**	**26**	**26**
	2,000	52	52	52	**51**
	3,000	77	78	77	**75**
	4,000	102	103	102	**99**
La24 (15 × 10)	1,000	**26**	27	27	**26**
	2,000	52	52	52	**50**
	3,000	77	77	77	**75**
	4,000	101	102	101	**98**

The bold values indicate the best found results

5 Conclusion

This paper presents an improved PPX-based crossover for multi-objective JSSP that reduces the redundancy at the tail of a chromosome. Experimental results exhibit that removing redundancy at the tail of a chromosome is helpful for getting better result in a reasonable computational time. Considering the real world demand for scheduling with reasonable time limit, experimental results justify that the proposed method will be very effective for large problems like 100 × 200, 150 × 300, etc. which is very usual in practice. We also believe that removing redundancy at the tail of a chromosome will be beneficial for all operation based chromosome representation for multiple-objective JSSP.

References

Applegate D, Cook W (1991) A computational study of the job-shop scheduling problem. ORSA J Comput 3(2):149–156

Bierwirth C (1995) A generalized permutation approach to job shop scheduling with genetic algorithms. OR Spektrum 17:87–92

Bierwirth C, Mattfield DC, Kopfer H (1996) On permutation representation for scheduling problems. Parallel Problem Solv Nat 4:310–318

Bucker P, Jurish B, Sievers B (1994) A branch and bound algorithm for the job shop scheduling problem. Discrete Appl Math 49:105–127

Chan FTS, Chung SH, Chan Y (2008) An introduction of dominant genes in genetic algorithms for FMS. Int J Prod Res 46(16):4369–4389

Davis L (1985) Job-shop scheduling with genetic algorithms. In: Proceedings of international conference on genetic algorithms and their applications, Lawrence Erlbaum, Hillsdale, NJ, pp 136–149

Deb K (2001) Multi-objective optimization using evolutionary algorithms. Wiley, Chichester

Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197

Eshelman L, Caruana R, Schaffer D (1989) Biases in the crossover landscape. In: Proceedings of the third international conference on genetic algorithms, June 1989, Virginia, USA, pp 10–19

Fang HL, Corne D, Ross P (1996) A genetic algorithm for job-shop problems with various schedule criteria. In: Evolutionary computing, AISB workshop, selected papers, Springer, Berlin, Germany, pp 39–49

Glover F (1989) Tabu search: part I. ORSA J Comput 3:190–206

Jensen MT (2003) Generating robust and flexible job shop schedules using genetic algorithms. IEEE Trans Evol Comput 7(3):275–288

Kirkpatrick S, Gelatt CD, Vecchi MP (1985) Optimization by simulated annealing. Sci New Ser 220:671–680

Lawrence S (1984) Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (supplement). Graduate School of Industrial Administration, Carnegie Mellon University, Technical Report

Lei D (2009) Multi-objective production scheduling: a survey. Int J Adv Manuf Technol 43:926–938

Mattfeld DC (1996) Evolutionary search and the job shop. Production and logistics. Physica-Verlag, Heidelberg

Muth JF, Thompson GL (1963) Industrial scheduling. Prentice-Hall, Englewood Cliffs

Nagar A, Haddock J, Heragu S (1995) Multiple and bicriteria scheduling: a literature survey. Eur J Oper Res 81:88–104

Ombuki BM, Ventresca M (2004) Local search genetic algorithms for the job shop scheduling problem. Appl Intell 21:99–109

OR Library. http://mscmga.ms.ic.ac.uk. Accessed 12 July 2009

Peter B, Chen LD, Thakur LS (1999) An effective approach for job-shop scheduling with uncertain processing requirements. IEEE Trans Rob Autom 15(2):328–339

Ripon KSN (2007) Hybrid evolutionary approach for multi-objective job-shop scheduling problem. Malays J Comp Sci 20(2):183–198

Song Y, Hughes JG, Azarmi N, Voudouris C (1999) A genetic algorithm with an incomplete representation for job shop scheduling problems. In: Proceedings of 18th work shop of the UK planning and scheduling special interest group, University of Salford, UK, 1999

Syswerda G (1991) Schedule optimization using genetic algorithm. In: Davis L (ed) Handbook of genetic algorithms. Van Nostrand Reinhold, NY, pp 332–349

Veldhuizen DV, Lamont GB (2000) Multi-objective evolutionary algorithms: analyzing the state-of-the-art. Evol Comput J 8(2): 125–148

Wroblewski J (1996) Theoretical foundations of order-based genetic algorithms. Fundam Inform 29:423–430

Yamada T, Nakano R (1992) A genetic algorithm applicable to large-scale job-shop problems. In: Proceedings of the second international conference on parallel problem solving from nature (PPSN '92), Elsevier Science Publishers, North Holland, 1992, pp 281–290

Yamada T, Nakano R (1997) Genetic algorithms for job-shop scheduling problems. In: Proceedings of modern heuristic for decision support, UNICOM seminar, 18–19 March, London, pp 67–81

Zitzler E (1999) Evolutionary algorithms for multi-objective optimisation: methods and applications. Ph.D. thesis, Dissertation ETH No. 13398, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland

Zitzler E, Deb K, Thiele L (2000) Comparison of multi-objective evolutionary algorithms: empirical results. Evol Comput J 8(2): 173–195