Application of Genetic Algorithm to Job Shop Scheduling Problems with Active Schedule Constructive Crossover

Lae-Jeong Park and Cheol Hoon Park
Department of Electrical Engineering
Korea Advanced Institute of Science and Technology
373-1 Kusong-Dong Yusong-Gu, Taejon 305-701, Korea
Tel: 82-42-869-3453 Fax: 82-42-869-3410 E-mail: chpark@expo.kaist.ac.kr

Keywords: genetic algorithm, representation, job shop scheduling problem

Abstract

This paper explains the application of a genetic algorithm(GA) to job shop scheduling problems. Given combinatorial problems, it is very important to develop an efficient representational scheme and effective genetic operators of the GA for better performance. Many representational schemes exist to encode ordering information of jobs and/or operations, which have their own advantages and shortcomings. For better performance, we use both job-ordered list on each machine and operation list representational schemes for a crossover and mutations, respectively, and develop a new crossover, so called, an active schedule constructive crossover(ASCX) and two mutations. This crossover can exchange meaningful ordering information of parents effectively without producing illegal solutions and the mutations can easily provide various operation orderings. Simulation results on five benchmark problems show that our genetic operators are very powerful and suitable to job shop scheduling problems and our GA outperforms the previous GA-based approaches and the GA with position-based crossover(PBX).

1. INTRODUCTION

The scheduling has been an active area of the research in applied artificial intelligence because a good scheduling algorithm can enhance productivity in various fields such as planning and operation of manufacturing systems[1], However, the generation of consistently good schedules is extremely difficult. It typically comprises several concurrent goals and several resources which must be allocated to satisfy such goals. In many cases, the combination of goals and resources has exponentially growing search space; in other words, it is called NP-complete[2]. Therefore, exact methods such as branch-and-bound[3] and integer programming techniques[4] require considerable computation time or complex mathematical formulation. Recently stochastic search techniques such as genetic algorithms(GAs)[5] have been successfully applied to job shop scheduling problems[6, 7, 8, 9, 10, 11, 12, 13]. Many GA-based approaches deal with specific problems or flow shop scheduling using appropriate representational schemes and knowledge-based genetic operators. Although the GA enables us to obtain good-quality solutions quickly and easily compared to other search techniques, there still exist performance gap between GA-based approaches and other specialized exact techniques. Our objective is to improve performance of the GA-based approach to job shop scheduling problems by developing effective genetic operators through the investigation of the relationship between representational schemes of ordering information of jobs or operations and genetic operators. In this paper, we propose new representational schemes of job shop scheduling problems and their associated genetic operators cooperating with domain-specific techniques for better performance. Table 1 depicts an example of a job shop scheduling problem composed of three jobs on three machines(3×3). Each job composes of operations with required machines and processing time along with precedence requirement. In the application to benchmark problems we use as a performance measure the makespan, that is, the total time required to complete all jobs among several criteria. This paper is organized as follows. Section 2 contains a concise survey of previous GA-based approaches to job shop scheduling problems. In section 3, we provide the two representational schemes for scheduling problems and discuss their advantages and shortcomings from the viewpoint of the search behavior by genetic operators. In section 4, we describe our genetic operators, active schedule constructive crossover(ASCX) and two mutations on each representational schemes. Two set of simulation results on five benchmark problems with our GA-based approach and others and with the ASCX and the PBX are shown in section 5. And the conclusion is made in section 6.

Job No.			
J_1	$(M_1, 5)$	$(M_2, 6)$	$(M_3, 1)$
J_2	$(M_2, 8)$	$(M_3, 2)$	$(M_1, 7)$
J_3	$(M_1, 4)$	$(M_3, 3)$	$(M_2, 5)$

Table 1: An example of job shop scheduling problem. (,) represents the machine and processing time required by each operation.

2. PREVIOUS GA APPROACHES

Since the demonstration of application of GA to scheduling problems by Davis[6], there have been many researches on this topic[7, 8, 9, 10, 11]. Some of them discussed flow shop problems or sequencing problems like the traveling salesman

problem(TSP) with crossovers designed for effective permutation of jobs or operations: edge recombination operator[7] and position-based crossover[8]. Others have been focused on the introduction of problem-specific representational schemes and thus more complicated genetic operators than traditional ones for job shop scheduling problems with alternative plans and machines[10, 11].

However, it is impossible to fairly compare these many GA-based approaches because their approaches are oriented too far to their own specific scheduling problems. Recently, Nakano *et al.* applied the conventional GA to well-known job shop scheduling benchmark problems[12]. They chose the representation of the binary genotype containing information on the job order on each machine and traditional binary crossover and mutation, but a repair mechanism is needed to correct illegal schedules after genetic operations. Fang *et al.* also used successfully the GA to solve the same problems with a variant of the ordinal representation that is developed for the validity of schedules under the conventional cut-and-mix crossover and mutation. Their performance enhancement is attributed mainly to the gene variance based operator targeting(GVOT) where the cut-point position of crossover and mutation is determined by the gene diversity in a population[13]. However, the genotype representation is redundant and thus false competition occurs.

3. REPRESENTATION

The selection of a representational scheme of solutions is a basic and essential prerequisite task for the successful application of the GA in that genetic operators work with genotypes in the adopted representational scheme. Therefore, care must to be taken to adopt both representational schemes and the associated crossover and mutation operators for an effective genetic search. From the viewpoint of a genetic search, it is worth while considering the two qualitative properties of the representational scheme: *Redundancy* and *Validity* or *Legality*.

A redundant representational scheme does not have a one-to-one correspondence between genotype and phenotype, which often causes false competition. In the case that the mapping is many-to-one, different genotypes are mapped into the same phenotype even though they are quite different. And crossover is unlikely to produce an offspring with a successful recombination of good building blocks of parents because two different but redundant genotypes may produce an offspring quite different from the original genotypes. Validity means that new solutions processed by genetic operators are still valid, and if they are invalid, illegal ones need to be repaired. Validity is decided by chosen genetic operators. It is extremely difficult to find a new representational scheme that has validity and no redundancy and still captures the inherent properties of the given problem and associated genetic operators.

For job shop scheduling problems, the genotype is usually encoded with the ordering information between jobs or operations, e.g., "J_1 on machine M_1 is scheduled before J_2 on machine M_3". And the phenotype is an actual and active schedule without unnecessary machine idle time. Our GA-based approach uses two representational schemes: *operation list* representation where a genotype consists of all operations in the scheduling order and *job-ordered list on each machine* representation. Table 2 shows one of the possible genotypes of the scheduling problem shown in table 1 with the two representational schemes. Both schemes can map all of the possible genotypes into the corresponding active schedules and are convertible to each other. Each genotype is scheduled in the following way. In the former, the first unscheduled operation belonging to the job at each position in order is scheduled on the designated machine at the earliest time when it can be started. The latter can be converted into the form of the former and then scheduled in the same way or can be scheduled by the active schedule generation algorithm[1]. Investigation of advantages and shortcomings of both representational schemes shows that the former has redundancy with respect to the latter. However, legal schedules can always be produced under any reordering(or permutation) operator, which does not hold in the case of traditional cut-and-mix crossover because the number of operations belonging to each job are changeable. On the other hand, although the latter has a one-to-one correspondence between the genotype and phenotype, both cut-and-mix crossovers and permutation operators usually generate invalid schedules. Therefore, in order not to use additional correction mechanism, genetic operators should be well designed for valid solutions as explained in the following section.

$J_1\ J_3\ J_1\ J_2\ J_3\ J_2\ J_1\ J_3\ J_2$

(a) Operation list representation

$M_1 : J_1\ J_3\ J_2$
$M_2 : J_1\ J_2\ J_3$
$M_3 : J_3\ J_2\ J_1$

(b) Job-ordered list on each machine representation

Table 2: Two different representations of one example schedule of table 1.

4. GENETIC OPERATORS

Based on merits and weaknesses of the two representational schemes explained in section 3, we develop new genetic operators more suitable and specific to job shop scheduling problems. Crossover and mutation effectively co-operates with each other in such a way that crossover mixes building blocks of both parents in the job-ordered list on each machine representation and mutation provide various building blocks in the operation list representation.

Crossover

The role of crossover is to generate better solutions by exchanging information contained in the current good solutions. However, in the case that redundancies exist in a representation, in other word, when the one-to-one correspondence does not exist between genotype and phenotype, simple exchange or mixing of both parents in genotypes may not fulfill its role

because it may generate unexpected offsprings even with two different genotypes of one phenotype. Therefore, it is very important to design a crossover that generates offsprings not by simply mixing but by effectively exchanging information contained in the two parents and to choose a representational scheme where a solution can be represented nonredundantly and explicitly so that a crossover can be developed as a useful information exchanger.

For job shop scheduling problems, we develop a new crossover, so called, active schedule constructive crossover (ASCX) that enables us to always generate new legal genotypes based on the active schedule generation algorithm. It is a variant of uniform crossover where a choice is made by inheriting an operation from either parent represented by a job-ordered list on each machine based on calculation of lower-bounds of partial schedules during construction of a new schedule, which will be explained in the below. Because job-ordered list on each machine representation provides more appropriate and structural representational form for information exchanger (or crossover) owing to no redundancy, the ASCX is more *structural* than permutation operators designed on the operation list representational scheme where job order information is implicitly encoded with high redundancy, but it is rather less disruptive. Table 3 shows the procedure of the ASCX. Let

$PS(i)$	be a partial schedule containing i scheduled operations,
$S(i)$	be the set of schedulable operations associated to given $PS(i)$ at stage i,
$\sigma(i,k)$	be the earliest time at which operation $k \in S(i)$ can be started,
$\phi(i,k)$	be the earliest time at which operation $k \in S(i)$ can be completed ,
$O(j,m)$, $j = 1,2$,	be the job-ordered list on machine m of two parents, respectively.

In STEP 3 the function $f()$ returns an operation among two selected operations k_1 and k_2 with probability p_s. That is, k_1 is selected with probability p_s, and k_2 with $1 - p_s$. According to this probability, the ASCX decides different recombination. To utilize the advantage that it constructs a schedule until completion of a schedule, we calculate p_s as shown in equation 1, through evaluation of partial schedules by calculation of their lower-bounds which is usually used in branch-and-bound search techniques[3]

$$p_s = \frac{1}{1 + e^{-\frac{L(k_2)-L(k_1)}{T}}} \tag{1}$$

where $L(k)$ is a lower-bound when operation k is added to current partial schedule, $PS(i)$, and T is a normalization factor. A large value of T makes p_s almost equal to 0.5. And on the other hand, in small values of T an operation with smaller lower-bound is more preferably selected. We use a simple lower-bound calculation method of Brooks-White[1] considering additional calculation time. It consists of the calculation of a job-based bound and a machine-based bound, based on the assumption that there is no machine conflict

STEP 1		Initialization
	i	$:= 0$
	$PS(i)$	:= { the null partial schedule }
	$S(i)$	:= { all operations with no precedent operations }
STEP 2		
	$\phi(i)^*$	$:= \min_{k \in S(i)}\{\phi(i,k)\}$
	m^*	:= the machine on which ϕ^* can be realized
STEP 3		
	$SS(i)$	:= { operations k that requires m^* and $\sigma(i,k) < \phi^*$ among $S(i)$ }
	k_1	:= the first appearing operation on $O(1,m^*)$ and belonging to $SS(i)$
	k_2	:= the first appearing operation on $O(2,m^*)$ and belonging to $SS(i)$
	k^*	$:= f(k_1,k_2)$; calculate the lower bounds with k_1 and k_2 and choose one of them according to $f(k_1,k_2)$
	$PS(i)$	$:= PS(i)$ + { operation k^* selected starting at time $\sigma(i,k^*)$ }
STEP 4		
	$S(i)$	$:= S(i)$ - { operation k^* } + { direct successor of operation k^* }
	i	$:= i + 1$
STEP 5		
		If all operations have been scheduled, exit. Otherwise, return to STEP 2

Table 3: The procedure of the ASCX.

when the remaining processing for all jobs is scheduled and that there is no precedence conflict when the remaining processing for all machines is scheduled, respectively. Let

$R(i,k)$	be the total of the unscheduled processing time for the job containing the operation $k \in S(i)$,
$\mu(i,m)$	be the latest completion time of operations in $PS(i)$ on machine m at stage i,
$N(i,m)$	be the total of the unscheduled processing time that will require machine m at stage i,
M	be the number of machine.

Equations (2) and (3) show the job-based lower-bound and the machine-based lower-bound, respectively. The final lower-bound is $\max\{b_1, b_2\}$.

$$b_1 = \max_{k \in S(i)} \{\sigma(i,k) + R(i,k)\} \tag{2}$$

$$b_2 = \max_{1 \le m \le M} \{\mu(i,m) + N(i,m)\} \tag{3}$$

With equations (1) to (3), we can perform a probabilistic recombination during construction of a new schedule. Incorporation of knowledge-based techniques like calculation of lower-bounds is useful in genetic search because it can accelerate convergence speed through the probabilistic guidance of promising search regions, which has been also exploited for TSP[14]. The usefulness of this probabilistic guidance is shown in simulation results compared to performance of random guidance with $p_s = 0.5$.

Mutation

Because of low disruptiveness of the ASCX, disruptive mutation operators are needed in order for the GA to continue to search without premature loss of the genetic diversity. In this sense, the operation list representation is more adequate for mutation operators in scheduling problems than the job-ordered list on each machine representation. The reason is as follows. The latter scheme needs complex and carefully-designed mutations preventing illegality and thus may not generate various ordering information due to legality restriction of schedules. On the other hand, the former enables us to always generate various legal schedules with any reordering operators.

Our two mutations, *scramble mutation*(SM) and *self-replication mutation*(SRM), are very simple. Given a schedule represented by the operation list representation, we select the positions to be reordered with probability α linearly decreasing as the GA evolve, and then change the order of operations in the selected positions randomly(SM) with probability γ. Or they change in accordance with the scheduling order, that is, the position order of the jobs containing the selected operations(SRM) with probability $1-\gamma$. Table 4 shows new schedules by scramble mutation and self-replication mutation in an example of table 1, respectively. The SM changes the order $\{J_1, J_2, J_3, J_2\}$ into $\{J_3, J_2, J_2, J_1\}$ randomly, and the SRM changes the order $\{J_1, J_2, J_3, J_2\}$ into $\{J_1, J_3, J_2, J_2\}$ because the position order is $\{J_1, J_3, .., J_2, .., J_2\}$. The SRM is expected to be helpful for the job shop scheduling problems whose machine order of jobs is similar.

	$J_1\ J_3\ J_1\ J_2\ J_3\ J_2\ J_1\ J_3\ J_2$
selection	$\bar{J}_1\ J_3\ J_1\ \bar{J}_2\ \bar{J}_3\ \bar{J}_2\ J_1\ J_3\ J_2$
SM	$\bar{J}_3\ J_3\ J_1\ \bar{J}_2\ \bar{J}_2\ \bar{J}_1\ J_1\ J_3\ J_2$
SRM	$\bar{J}_1\ J_3\ J_1\ \bar{J}_3\ \bar{J}_2\ \bar{J}_2\ J_1\ J_3\ J_2$

Table 4: Examples of two mutations: the scramble mutation and the self-replication mutation.

5. SIMULATION RESULTS AND DISCUSSION

Our GA-based approach is tested on five benchmark problems such as ABZ6, CAR4, LA22, MT10, and MT20[15]. Their problem size are 10×10, 14×4, 15×10, 10×10, and 20×5, respectively. And the makespans of the optimal solutions are 943, 8003, 927, 930, and 1165, respectively. In the case of MT10 and MT20, the machine order of operations belonging to each job is assigned so that the lower-numbered machines tend to be used for the earlier operations and the higher-numbered machines for later operations. ABZ6 and LA22 does not have any tendency, and CAR4 has the same machine order on each job.

All simulations are performed with population size of 500, linear rank-based selection for reproduction, deterministic selection for replacement, and 300 generations so that for comparison the total numbers of trials becomes 150,000 that is equal to those of other GA-based approaches[12, 13]. Initial populations are generated by active schedule generation algorithm. Parameters of our genetic operators is chosen by rough trial-and-errors: normalization factor T of the ASCX and the reordering portion factor α of two mutations are 20 and 0.1, respectively. And the selection ratio of two mutations, γ, is 0.7. Table 5 shows the results of the previous

	MT10		MT20	
	Average	Best	Average	Best
Optimum	–	930	–	1165
Nakano *et al.*	–	965	–	1215
Fang *et al.*	977	949	1215	1189
Our GA	949	936	1185	1178

Table 5: The best results of GA-based approaches on MT10 and MT20.

GA-based approaches and our GA with MT10 and MT20, respectively[1]. Although GA-based approaches adopted by Nakano *et al.* can apply the conventional GA to job shop scheduling problems directly because of binary representation of the precedence relation of jobs on each machine, the conventional crossover with their highly redundant representational scheme produces illegal solutions and thus was inefficient. With a variant ordinal representation Fang *et al.* could always generates legal schedules under crossover and mutation without any repair mechanism and improved the results. But the results are not satisfiable, yet, because of the representation and the conventional crossover such as one-point crossover and uniform crossover improper to scheduling problems. Better performance of our GA-based approach is presumably to be attributed to the synergy effect between crossover and mutation designed adequately on different representational schemes, respectively. In other words, with operation list representation, two kinds of reordering mutations continuously generates various new schedules that otherwise may easily not be generated. On the other hand, the ASCX based on the job-ordered list on each machine representation generates new schedules by structurally exchanging ordering information explicitly represented in the two parents.

In order to examine that the ASCX really helps to improve performance of the GA, we also run the GA with position-based crossover(PBX)[8] and the operation list representation because it was demonstrated that the PBX is the fittest in some scheduling problems[16]. It generates a new schedule in the following way. A set of positions to be reordered is selected with probability β, and then the order of operations in the selected positions in one parent changes in accordance with the order of the corresponding operations in the other parent. Parameter β decides the portion of all positions to be reordered and 0.5 is chosen among 0.2, 0.3, 0.4, 0.5 by rough trial-and-errors. In some sense, the PBX has less meaningful information exchange than the ASCX since the order of operations of the other parent is not preserved. We run GAs with three sets of crossover rate p_c and mutation rate p_m: (0.5, 1.0), (1.0, 1.0), and (1.0, 0.5). Table 6

[1] Adapted from [13] where 10 runs are averaged.

		(0.5, 1.0)		(1.0, 1.0)		(1.0, 0.5)	
		Aver.	Best	Aver.	Best	Aver.	Best
ABZ6	PBX	957	947	956	947	953	943
	ASCX	948	947	953	947	964	947
	SM	955	948	–	–	–	–
CAR4	PBX	8052	8003	8012	8003	8010	8003
	ASCX	8003	8003	8003	8003	8004	8003
	SM	8097	8003	–	–	–	–
LA22	PBX	957	941	950	940	952	942
	ASCX	949	935	955	943	970	950
	SM	967	955	–	–	–	–
MT10	PBX	961	937	958	937	966	958
	ASCX	949	936	953	940	955	940
	SM	963	937	–	–	–	–
MT20	PBX	1193	1178	1190	1180	1191	1180
	ASCX	1185	1178	1186	1180	1197	1180
	SM	1204	1191	–	–	–	–

Table 6: Simulation results of five benchmarks.

shows the makespans of the best schedules and their averages obtained after 20 runs of 300 generations by the PBX with mutation, the ASCX with mutation, and the SM only, respectively. As shown in table 6, the SM only find relatively good schedules quickly without crossover compared to other GA-based approaches in table 5 because reordering operators are more appropriate to job shop scheduling problems than conventional cut-and-mix crossovers. GAs with crossovers give better solutions at the late iterations as shown in figure 1. The ASCX with mutation gives the best performance when (p_c, p_m) is (0.5, 1.0) and shows rapid convergence rate at the early iterations but finally premature convergence with higher crossover rate $(p_c, p_m) = (1.0, 1.0)$ and (1.0, 0.5). The SRM was powerful for CAR4, MT10, and MT20 as expected. On the other hand, in the case of the PBX with mutation (1.0, 1.0) and (1.0, 0.5) give good performance. Through these observations, the ASCX is less disruptive than position-based crossover but gives better performance with mutations. It seems that for job shop scheduling problems, when highly disruptive reordering mutation is used, crossover having more structural information exchange property and less crossover rate may be preferable than more disruptive crossover.

Figure 2 shows convergence rates with and without probabilistic guidance by partial schedule evaluation. Probabilistic guidance during construction of new schedules results in somewhat faster convergence and better performance. In the case of CAR4 and MT10, it improves performance significantly.

6. CONCLUSION

This paper explains the application of a GA to job shop scheduling problems and presents that the GA is an effective stochastic search algorithm in the sense of reasonable computation time(about 15 minutes on SPARC 2) and quality of solutions. With consideration of representation properties such as redundancy and validity we developed a new crossover, active schedule constructive crossover, on the job-ordered list on each machine representation and two mutations on the operation list representation. Simulations on five benchmark problems shows that our GA outperforms previous GA-based approaches and the GA with other genetic operators like position-based crossover. These results show that when scheduling information is represented as genotype, synergy effect between crossover and mutation may be maximized through a structural crossover and non-structural and highly disruptive reordering mutation.

Although the GA is unlikely to outperform highly specialized and knowledge-incorporated algorithm in job shop scheduling problems up to now, it offer high-quality solutions quickly and simply. And it is also easier to parallelize than other traditional methods. We are trying to implement parallel version of our GA on a MIMD parallel machine so as to achieve better solutions in shorter computation time. Studies on job shop problems with due dates are under investigation.

References

[1] K. R. Baker, *Introduction to sequencing and scheduling*, Wisely, New-York, 1974.

[2] M. R. Garey and D. S. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*, Freeman, 1979.

[3] J. Carlier and E. Pinson, "An algorithm for solving the job shop problem," *Management Science*, vol. 35, pp. 164-176, 1989.

[4] M. Fisher, "Lagrangian relaxation method for solving integer programming problems," *Management Science*, vol. 27, pp. 1-8, 1981.

[5] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*, Addison-Wesley, 1989.

[6] L. Davis, "Job shop scheduling with genetic algorithm," *Proceedings of the First Conference on Genetic Algorithms and their Applications*, pp. 136-140, 1985.

[7] D. Whitely, T. Starkweather, and D'Ann Fuquay, "Scheduling problems and traveling salesman: The genetic edge recombination operator," *Proceedings of the Third Conference on Genetic Algorithms*, pp. 133-140, 1989.

[8] G. Syswerda, Schedule optimization using genetic algorithm, In *Handbook of genetic algorithm*, Van Nostrand Reinhold, New York, 1990.

[9] G. A. Cleveland and S. F. Smith, "Using genetic algorithms to schedule flow shop releases," *Proceedings of the Third Conference on Genetic Algorithms*, pp. 160-169, 1989.

[10] S. Bagchi, S. Uckun, Y. Miyabe, and K. Kawamura, "Exploring problem-specific recombination operators for job shop scheduling," *Proceedings of the Fourth Conference on Genetic Algorithms*, pp. 10-17, 1991.

[11] R. Bruns, "Direct chromosome representation and advanced genetic operators for production scheduling," *Proceedings of the Fifth Conference on Genetic Algorithms*, pp. 352-359, 1993.

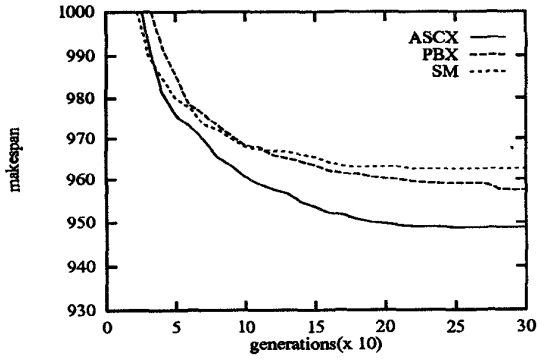
[12] R. Nakano and T. Yamada, "Conventional genetic algorithm for job shop problems," *Proceedings of the Fourth Conference on Genetic Algorithms*, pp. 474-479, 1991.

[13] H.-L. Fang, P. Ross, and D. Corne, "A promising genetic algorithm approach to job-shop scheduling, rescheduling, and open-shop scheduling problems," *Proceedings of the Fifth Conference on Genetic Algorithms*, pp. 375-382, 1993.

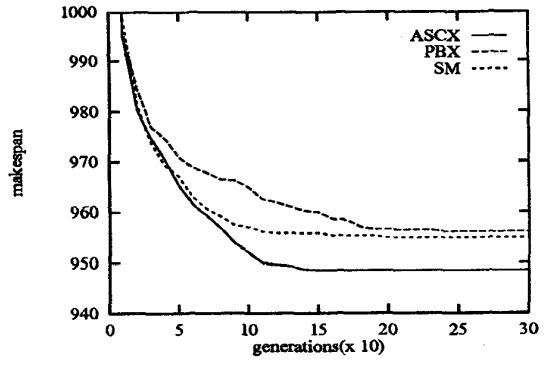
[14] J. J. Grefenstette, Incorporating problem-specific knowledge into genetic algorithm, In *Genetic algorithms and simulated annealing*, pp. 43-60, Morgan Kaufmann, 1987.

[15] D. C. Mattfeld, OR-Library, taken from the ftp site(o.rlibrary@ic.ac.uk).

[16] T. Starkweather, D. McDaniel, K. Mathias, D. Whitely, and C. Whitely "A comparison of genetic sequencing operators," *Proceedings of the Fourth Conference on Genetic Algorithms*, pp. 69-76, 1991.



(a) MT10



(b) ABZ6

Figure 1: Convergence of the averages of the best solutions of three operators: ASCX with mutations, PBX with mutations, and SM on two benchmark problems, MT10 and ABZ6.

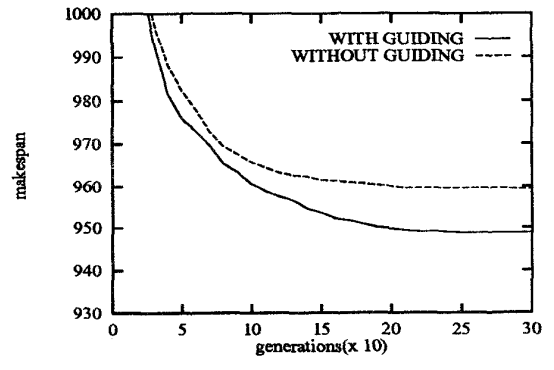


Figure 2: Convergence of the averages of the best solutions with and without probabilistic guidance by evaluation of partial schedules on MT10.