Extrapolation-Directed Crossover for Job-shop Scheduling Problems : Complementary Combination with JOX

Jun Sakuma
Graduate School of Science and Engineering
Tokyo Institute of Technology
4259 Nagatsuta, Midori-ku
Yokohama 226-8502, Japan
jun@fe.dis.titech.ac.jp

Shigenobu Kobayashi
Graduate School of Science and Engineering
Tokyo Institute of Technology
4259 Nagatsuta, Midori-ku
Yokohama 226-8502, Japan
kobayashi@dis.titech.ac.jp

Abstract

In this paper, we propose a new Genetic Algorithm for JSP using two crossovers. The crossover, JOX, obtained relatively good results, however offspring generated by JOX exist around parents or within an intermediate area of them. This feature of JOX induces a convergence of the whole population. To deal with this fault of JOX, we propose a complementary combination of two crossovers. One is JOX, and the other, EDX, is our proposal. EDX is designed to have the population enlarge using a local search and explores the area where the population uncovers. Although a mutation is applied for exploration in general, we apply a framework of crossover to EDX for a more efficient exploration. The combination of two crossovers, which has a different search area, is able to compensate for each other's fault. The GA designed with these two crossovers was applied to large-size JSP benchmarks, and we show its effectiveness.

1 Introduction

The job-shop scheduling problem(JSP) is well known as one of the most difficult NP-hard ordering problems. The JSP discussed in this paper is a makespan reduction problem that can be described as follows: N jobs are to be processed on M machines; each job follows a prescribed routing; the processing times of operations on M machines are also prescribed; and all operations are non-preemptive. The objective of the JSP is to obtain a schedule having minimum makespan that is the earliest time that all jobs can be completed.

In the field of operations research, the branch and bound(BAB) method for JSP has been studied for a few decades. In recent years, however, a more efficient approaches using approximation algorithms were proposed. For example, algorithms based on local search called Shifting Bottleneck procedure [Applegate 91], simulated annealing [Aarts 94][Yamada 96], tabu search [Nowicki 93] obtained good results. The combination of simulated annealing and shifting bottleneck procedure proposed in [Yamada 96] showed the best performance, especially in large-size JSP benchmarks from the viewpoint of a computational time and a quality of the solution. On the other hand, several genetic algorithms was also applied to JSP. For example [Kobayashi 95][Ono 96][Yamada 97][Shi 96].

Although GAs are very effective optimizers and applicable to many kinds of combinatorial/continuous optimization problems, the performance of GAs mainly depends on the design of crossovers.

Crossover SXX proposed in [Kobayashi 95] was originally designed for any permutation representation, such as TSP. Crossover SXX can be applied to any combinatorial optimization problem whose solutions are represented with permutations and is superior in generality. On the other side, SXX does not consider the dependency among elements and does not exceed existent algorithms in performance. Crossover JOX+GT method [Ono 96] is designed considering the dependency among machines and got good results. Crossover SPX [Shi 96] pays attention to the almost same dependency. Crossover MSXF proposed in Genetic local Search [Yamada 97] is based on a local search. SXX, JOX and SPX generate its offspring by exchanging elements that exist in parents. On the contrary, MSXF doesn't exchange the elements between parents. But the search area of MSXF is biased toward the other parent and in terms of the feature of MSXF, it resembles traditional crossovers.

As a qualitative framework of describing a behavior of the GAs, the Functional Specialization Hypothesis (FSH) was proposed in [Kita 99]. Given the assumption that the population size is sufficiently large, FSH illustrates the GAs as a development of a probabilistic distribution function (p.d.f.) of individuals. Moreover, FSH explains three statistic operators in the GAs: se-

lection; crossover; and mutation as follows:

1. The selection operation narrows the p.d.f. by selecting and duplicating individuals having higher fitness.
2. The crossover operation, the primary search operation in the GAs, converts the p.d.f. by generating offspring by combining information of parents.
3. The mutation operation, the secondary search operation, enlarges the p.d.f. by giving perturbation to each individual.

FSH gives a clear guideline for designing crossovers for a real-coded genetic algorithm. [Kita 99] showed the crossover UNDX [Ono 97] for real-coded GA capably inherits statistics of parents such as the mean vector and the covariance matrix of the population.

We have two purposes in this paper. The first purpose is to analyze the behavior of crossovers for combinatorial optimization problems, JSP, in terms of p.d.f., given the assumption that the distance and the transition operator, which depends on the problem domain, is able to be defined. We examine in detail the feature of JOX from a viewpoint of p.d.f., and pay special attention to the fact that JOX almost generates its offspring inside the population.

The second purpose is to propose a more effective crossover for JSP following the result of the analysis. In this paper, to compensate for this fault of JOX, we design a new crossover, EDX, which is based on probabilistic local search and focuses on the explorative search. Moreover, we propose a GA for JSPs with complementary combination of EDX and JOX. The effectiveness of the proposed method is shown experimentally by applying 10 large-size JSP benchmark problems.

In section 2, we describe JOX+GT method [Ono 96]. In section 3, we define the method to map the distribution of the solutions in the discrete solution space to two dimensional Euclidean coordinate system and introduce the general concept of extrapolation and interpolation. Then we propose the extrapolation directed crossover, EDX and examine the behavior of EDX. In section 4 we design a GA based with two crossovers, and section 5 shows the experimental results. Section 6 contains our conclusion.

2 The Inter-machine Job-based Order Crossover and GT Method

2.1 Inter-machine Job-based Order Crossover

Inter-machine Job-based Order Crossover (JOX) uses the order of each job on all machines to represent the solution of JSP. The order of each job corresponds to sequences of operations on each machine in Gantt Chart representation. JOX considers the dependency among machines. The relevant algorithm is as follows:

1. With probability 0.5, choose the jobs whose locus is preserved.
2. Copy the jobs chosen in step 1 from donor to offspring preserving their locus.
3. Copy the jobs which are not copied in step 2 from acceptor to offspring preserving their order.

2.2 Enforcement using the GT method

Offspring generated by JOX are not always feasible. To enforce an infeasible schedule to a feasible one, the GT method is used. Originally, the GT method was proposed for exhaustively generating active schedules. An active schedule is a schedule in which no operation can be processed earlier by the permissible left shift. It was proved that optimal schedules are included in a active schedule set. The GT method generates any active schedules by repeating the following procedure.

1. Let O^*be an operation whose earliest completion time is minimum among unscheduled ones and M^*be the machine that processes O^*
2. Make a conflict set C, which contains the unscheduled operations that are processes on the M^* and whose processing overlap with O^*
3. Choose an operation from C randomly and schedule it.
4. Repeat procedures 1 to 3 until there exist no unscheduled operation.

The GT method can be used for judging the activeness of a given job sequence matrix and modifying it into an active schedule if it is not active. Let Jm^* be the first unscheduled job of machine M^* on the job sequence matrix. In step 3 of the above procedure, select the operation $O(M^*, Jm^*)$ and schedule it. Otherwise, choose an operation $O(M^*, J)$ in C randomly, shift it to the head of the job sequence of unscheduled operations and schedule it.

3 The Proposal of Extrapolation Directed Crossover

3.1 Definition of extrapolation and interpolation

In FSH, a crossover is explained as an operator that generates new sampling points depending on a distribution of the parent population and a mutation is defined as an operator that gives perturbation to each individual in order to enlarge the population. In other words, a crossover that focuses on the area covered by

the parent population plays a role of exploitation, and a mutation that focuses on the uncovered area by the parent population plays a role of exploration.

We call a search that generates sample points on the area covered by the parent distribution an **interpolation search**, and a search that generates sample points on the area uncovered by the parent distribution **extrapolation search**.

We are required to deal with the discrete space that doesn't have any concept about inside and outside. Therefore, we define **interpolation/extrapolation** using the distance between two individuals, on the assumption that the definition of the distance and the transition operator is decided in advance.

s_i represents a solution in a discrete space. Here, we consider three solution: s_a, s_b and s_c. $|s_a - s_b|$ denotes the distance between s_a and s_b. $|s_a - s_b|$ and $|s_b - s_a|$ have the same real value.

In the two dimensional Euclidean space, let the coordinates of s_a be $(0,0)$ and the coordinates of s_b be $(d,0)$, where $|s_a - s_b| = d$. Then, the coordinates of $s_c(x,h)$, where $|s_c - s_a| = d_1, |s_c - s_b| = d_2, d_1 \leq d_2$, is represented as follows:

$$\begin{aligned} x &= \frac{d^2 + d_1^2 - d_2^2}{2d} \\ h &= \sqrt{d_1^2 - x^2} \end{aligned}$$

We call this representation a **distance representation**. Moreover, we define the partial solution space about three solutions s_a, s_b, s_c as follows:

$$\begin{aligned} S_{in} &= \{s_c \in S | d_1 \leq d \text{ and } d_2 \leq d\} \\ S_{ex} &= \{s_c \in S | d_1 > d \text{ or } d_2 > d\} \end{aligned}$$

Here, S denotes the entire solution space. We refer to S_{in} as the **interpolation area** of s_a and s_b and refer to S_{ex} as the **extrapolation area** of s_a and s_b. Fig.1 illustrates an interpolation area and extrapolation area. The gray area shows the interpolation area. This definition is applicable to any discrete space if a distance is able to be defined among solutions.

Fig. 2 shows a typical distribution of offspring generated by JOX with distance representation. We adopted I2 distance (see Appendix) as a distance of the solution space of JSP. The parents are located at $(0,0)$ and $(104,0)$. All offspring are located in the interpolation area. We experimentally confirmed that more than 99% of the offspring generated by JOX exist in an interpolation area in any stage of the search of the GA using JOX.

When only an interpolation crossover is applied to GAs, it may still possible to obtain a optimum if large-

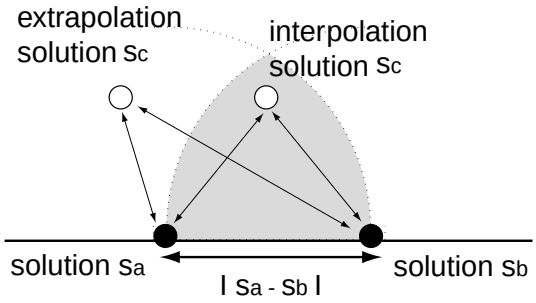

Figure 1: Definition of extrapolation and interpolation

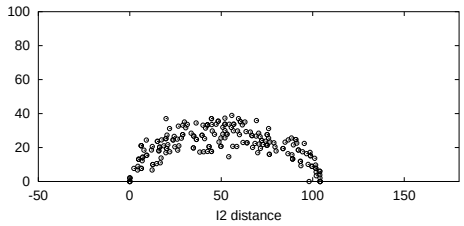

Figure 2: The distribution of offspring generated by JOX

size population is provided and an ideal generation alternation model, which can preserve a variety of individual sufficiently, is used. Under the limited computational cost, however, the population size can not help but be limited. Therefore, we conclude from the above that the combination of mutation operator that gives perturbation to each individuals is desirable.

3.2 Previous Work

3.2.1 Inter-machine Job-order based Shift Change

In combinatorial optimization problems where a solution is represented as a permutation, Swap or Shift Change is generally used to give perturbation to the individuals. Fig. 3 shows the distribution of the offspring that were applied a mutation, Job-order based Shift Change (JBSC)[Ono 96], which considers a dependency among machines. The parents are the same as used in Fig. 2. First, the offspring was generated by JOX, which are represented with gray circles in Fig. 3, and then JBSC was applied, which are represented with black circles.

The search area of mutations, which include JBSC, is not affected by any other individuals and is maintained uniformly throughout the whole process of the search. Therefore, it is efficient for the purpose of recovering the elements that were lost from the population in the early stage of the search. But it does not work as well especially in the end stage of the search because the

distribution of random mutations is too broad as observed in Fig. 3 and the improvement rate of random mutations becomes extremely low.

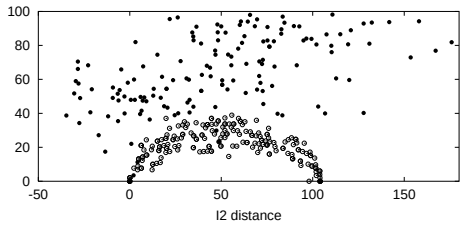

Figure 3: The distribution of offspring generated by JOX and JBSC

3.2.2 Multi Step Crossover Fusion

MSXF [Yamada 97] is a crossover fused with a stochastic local search using two solutions.

In MSXF, a solution, initially set to be one of the parents, selects a candidate among the CB neighborhoods of the parent. This selection is biased by the distance between the candidate and another parent. The smaller the distance to the other parent, the bigger the probability that the selected solution becomes a candidate. If the fitness value becomes better than the solution before a transition, it is replaced. Otherwise, the replacement is taken place stochastically (the metropolis method).

Fig. 4 represents the search trajectory of MSXF with the distance representation. By biasing the search direction using the other parent, MSXF narrows its search area efficiently.

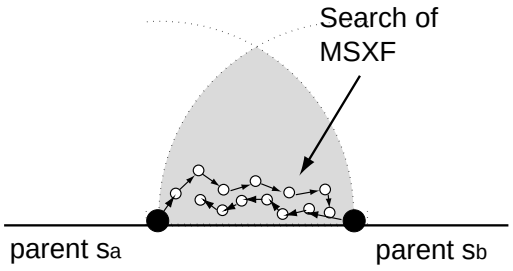

Figure 4: The search trajectory of MSXF

3.3 Proposal of Extrapolation Directed Crossover for JSP

On the basis of this previous work, we propose a Extrapolation Directed Crossover, EDX, based on a stochastic local search. EDX is designed for the purpose of giving individuals perturbation, and assumed to be applied with JOX together in the GA. The search area of EDX is controlled by the landscape of fitness around the individual and therefore the given perturbation becomes more efficient than a random mutation.

Several kinds of neighborhood for JSP have been proposed previously. We adopt the CB neighborhood proposed in [Yamada 96] whose size is relatively small and the improvement rate is high. The CB neighborhood is generated by shifting a operation that exists inside the critical block to the head position of the critical block or to the end thereof.

EDX uses two individuals as parents similar to JOX or MSXF. The search of EDX starts from one of the parents. The parent solution steps away from the other parent by selecting a solution from one of the CB neighborhoods and replacing it with the parent. Although a solution transits only to its neighbor, the direction of the search is biased by the other parents and is dynamically changed by the combinations of two parents. This feature is similar to a traditional crossover.

EDX is executed by repeating the following procedure: s_i denotes a solution of JSP; b^i_j denotes a $I2$ coordinates (see Appendix) of solution s_i; $f(s)$ denotes a fitness of solution s; p_{ex}, p_{temp} are real value parameters. p_{ex} is bounded from 0 to 1 and p_{temp} is a positive parameter. The EDX algorithm, which starts from a parent s_a, is as follows:

1. Select a pair of parent solutions s_a, s_b. $f_{best} := f(s_a), s_{best} := s_a$.
2. Generate CB neighborhoods of s_a, $CB(s_a)$.
3. Divide the $CB(s_a)$ into two sets $CB_{ex}(s_a)$ and $CB_{in}(s_a)$ so that $s_{ex} \in CB_{ex}$ satisfies $|b^{ex}_{j^*} - b^b_{j^*}| > |b^a_{j^*} - b^b_{j^*}|$ and $s_{in} \in CB_{in}$ satisfies $|b^{in}_{j^*} - b^b_{j^*}| \leq |b^a_{j^*} - b^b_{j^*}|$. Here, j^* denotes a job name of the operation which was transferred when generating the CB neighborhoods.
4. Select s_{new} randomly from CB_{ex} with probability p_{ex} (extrapolation search), otherwise select s_{new} randomly from CB_{in}(interpolation search).
5. If $f(s_{new}) < f_a$, then $s_a := s_{new}$ with probability 1. Otherwise, $s_a := s_{new}$ with probability $p_{ac} = exp(-\frac{f(s_{new}) - f(s_a)}{p_{temp}})$.
6. If $f(s_a) < f_{best}$, then $s_{best} := s_a, f_{best} := f(s_a)$.
7. If a termination condition is satisfied, output s_{best}, f_{best} and terminate. Otherwise, repeat the above steps from 2 through 6.

3.4 The Behavior of EDX

There exists three parameters that control the behavior of EDX.

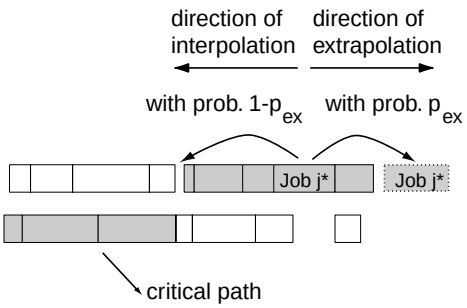

Figure 5: The algorithm of EDX

1. p_{ex} is a probability that the solution transfers to the direction of extrapolation area. As this parameter becomes larger, the search area of EDX becomes enlarged to the direction of extrapolation.
2. p_{LS} is a probability of applying EDX to a selected pair. Then a probability of applying JOX becomes $1 - p_{LS}$. As the parameter becomes larger, the chance of executing extrapolation search becomes more frequent.
3. p_{temp} is a temperature parameter.

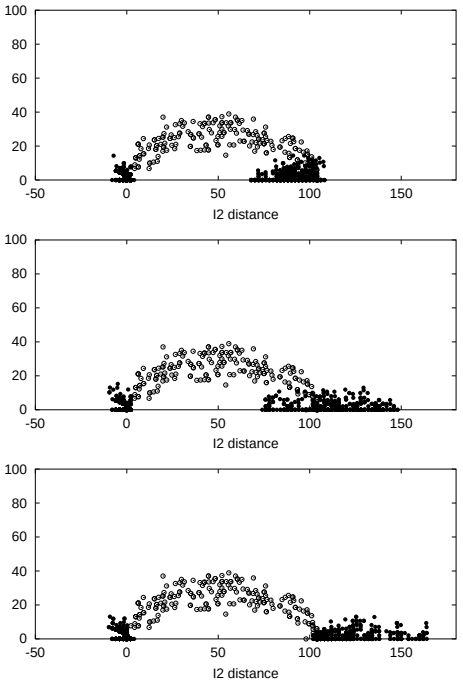

Figure 6: The search area of EDX and JOX : the top is $p_{ex} = 0.5$/the middle is $p_{ex} = 0.75$/ the bottom is $p_{ex} = 1.0$

Fig. 6 shows offspring generated by EDX when the parameter p_{ex} is 0.5, 0.75 and 1.0. Gray points represent offspring generated by JOX and black points represent the offspring generated by EDX.

The EDX with $p_{ex} = 0.5$ corresponds to a no-biased stochastic local search. Therefore, the solution is allowed to transfer both directions, extrapolation and interpolation. In Fig. 6, a right-hand side solution transfers to the interpolation area. However, the EDX with $p_{ex} = 1.0$ allows a solution only to transfer to the direction of extrapolation. Offspring in the interpolation area at $p_{ex} = 1.0$ are shifted ones by the GT method.

4 Designing A GA for JSP

In this section, we design a GA for JSPs. Fig. 7 shows the concept of generation alternation model, CCM [Ono 98b] employed in this paper.

We should note that CCM localizes its selection pressure not to the whole population as Simple GA or Steady-State does, but only to the descendant of the parent. By comparative experiments about the optimization of real-value function and Traveling Salesman Problem(TSP), a generation alternation model which localizes selection pressure to its family gives better results than a model that does not localizes it [Satoh 96]. Localizing a selection pressure to the descendant, which is more rigid localization than to the family, suppress the convergence of the population capably, especially in the difficult problems[Ono 98b][Nagata 99].

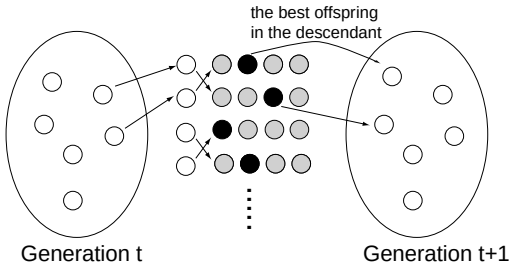

Figure 7: Concept of CCM

To apply two crossovers to our GA, we modify the CCM. The algorithm of the modified CCM for the combination of EDX and JOX is as follows:

1. Make an initial population that is composed of n_{pop} random operation sequences representing active job schedules.
2. Prepare $n_{pop}/2$ pairs of two individuals s_a, s_b randomly from current population.
3. Generate offspring by applying EDX n_{EDX} times to each pair with probability p_{ex}, otherwise applying JOX n_{JOX} times.
4. Enforce all offspring to active schedules and evaluate them.

5. Choose the offspring that has the highest fitness in each *Descendant* [1] and create next population with them.
6. Repeat the above steps 2 through 5 until a stop condition is satisfied.

In case of applying EDX at step 5, the enforcement and evaluation is executed sequentially while processing EDX.

5 Experiments and Results

The proposed procedure was implemented by C++ on the Pentium3-550MHz.

5.1 Preliminary Experiments

The performance of the proposed method was tested by benchmark problems. In the first experiment, the well known and relatively small-size (10 jobs-10 machines) benchmark, ft10 and a more difficult benchmark abz5 was used. The optimal makespan of ft10 is 930 and that of abz5 is 1234.

The purpose of the first experiment is to determine the fine parameters for EDX, p_{ex} and p_{LS}. At first, we set $p_{LS} = 0.1$ temporarily, and examine several different value of p_{ex}. We set the following parameters in advance. $p_{temp} = 15, n_{pop} = 50, n_{JOX} = 100, n_{EDX} = 500$. We stop the whole algorithm, if the GA find the optimal schedule or the 5.0×10^5 schedules are produced.

The results of these experiments are summarized in Table 1 and Table 2, where each column denotes the value of p_{ex}, an averaged best fitness, an averaged cpu time for a single run and the number of the trial that succeeded to obtain the optimum solution. Each result was averaged over 30 trials. The unit of time is one second and a cpu time for each run means the final updated time of the best fitness of the population.

The performance in ft10 isn't affected by the value of p_{ex}. Thus, ft10 is an easy instance and it is assumed that the extrapolation search doesn't promote the search speed of the GA. On the contrary, in abz5, when p_{ex} is 0.66 or 0.75, the search speed is shortened by about 40% while preserving the same quality of the solutions compared with $p_{ex} = 0.5$. This shows the effectiveness of the extrapolative search.

Second, we determine the parameter p_{LS}. We set $p_{ex} = 0.66$ following the result of the first experiment, and examine several different values of p_{LS}. Any other condition of the experiment is the same as the first experiment. The results of these experiments are summarized in Table 3 and Table 4.

Table 1: Performance in each p_{ex} : ft10

p_{ex}	ave. of best	ave. of time	opt/trial
0.5	930.5	102.19	28/30
0.66	930.67	99.66	27/30
0.75	930.4	99.0	28/30
1.0	932.90	124.26	21/30

Table 2: performance in each p_{ex} : abz5

p_{ex}	ave. of best	ave. of time	opt/trial
0.5	1235.17	243.4	22/30
0.66	1234.76	137.5	24/30
0.75	1234.73	135.6	22/30
1.0	1237.0	230.8	8/30

Table 3: Performance in each p_{LS} : ft10

p_{LS}	ave. of best	ave. of time	opt/trial
0.0	932.43	62.18	22/30
0.1	931.26	100.02	26/30
0.25	930.53	126.10	28/30
0.50	931.03	171.09	26/30
0.75	931.40	199.89	25/30
1.0	933.36	180.28	18/30

Table 4: performance in each p_{LS} : abz5

p_{LS}	ave. of best	ave. of time	opt/trial
0.0	1242.40	100.72	1/30
0.1	1235.20	177.56	20/30
0.25	1234.46	202.49	24/30
0.50	1234.13	209.56	28/30
0.75	1234.33	239.61	26/30
1.0	1235.03	203.96	21/30

The GA without EDX (i.e. $p_{LS} = 0.0$) can hardly obtain the optimal solution of $abz5$. But with EDX, the GA can find the optimal solution in the ratio of more than eight to ten. As p_{LS} becomes larger from 0.0 to 0.5, the quality of solution improves and the running time becomes longer. But at more than 0.5, the quality of solution becomes worse considering the running time.

From the above, we confirmed that too much extrapolative search has a bad influence upon the search itself. Therefore, we may reasonably conclude that p_{ex} should be from 0.66 to 0.75, and p_{LS} should be 0.1 to 0.25, especially in the difficult instances.

Third, in order to confirm the effectiveness of EDX, we compare EDX with a mutation JBSC. We set $p_{ex} = 0.75$ and $p_{LS} = 0.25$. JBSC was applied with probability 0.1 after JOX was applied. Any other condition of the experiment is the same as the first experiment. The results of these experiments are summarized in Table 5 and Table 6.

[1] *Descendant A* of EDX means s_a and offspring generated from s_a. *Descendant A* of JOX means s_a and offspring that are generated from parents s_a, s_b and preserves the loci of s_a.

Table 5: Comparison results with several JOXs : ft10

crossover	ave. of best	ave. of time	opt/trial
JOX	932.43	62.18	22/30
JOX+JBSC	932.03	154.81	19/30
JOX+EDX	930.53	126.10	28/30

Table 6: Comparison results with several JOXs : abz5

crossover	ave. of best	ave. of time	opt/trial
JOX	1242.40	100.72	1/30
JOX+JBSC	1239.00	201.89	4/30
JOX+EDX	1234.46	202.49	24/30

The both results show that EDX with JOX is preferable to JBSC in both terms of computation time and the quality of the best obtained solution.

5.2 Experiments and Results

We applied our proposal method to the well-known large-size problems of JSP, ten tough problems. We set the following parameters in advance. $p_{temp} = 15, n_{pop} = 50, n_{JOX} = 200, n_{EDX} = 500, p_{LS} = 0.25, p_{ex} = 0.66$. We stopped the whole algorithms if the GA finds the optimal schedule or the 5.0×10^7 schedules are produced. The results of these experiments are summarized in Table 7. The symbol * indicates that the solution is optimum. The first column shows the name of instances and optimum(*)/upper bound(no marked). The second column shows the best fitness obtained thorough 10 iterations. The third column shows the cpu time(sec) to obtain the best solution.

Table 7: Results of experiments in 10 tough problem using JOX+EDX

instance	JOX+EDX	time	opt/trial
*abz7(*656)*	670	3.58×10^4	0/10
abz8(669)	683	2.04×10^4	0/10
abz9(679)	686	1.94×10^5	0/10
*la21(*1046)*	*1046	5.90×10^4	1/10
*la24(*935)*	*935	1.75×10^4	4/10
*la25(*977)*	*977	5.65×10^3	4/10
*la27(*1235)*	1236	1.13×10^5	0/10
la29(1153)	1167	2.00×10^5	0/10
*la38(*1196)*	*1196	2.79×10^4	1/10
*la40(*1222)*	1224	2.72×10^4	0/10

We compare our results to the other GA/GLS approaches (Table 8). Ono98 was implemented with a population size 3000 and JBSC was used as mutation [Ono 98a]. The proposed method JOX+EDX loses to other GA/GLS approaches in two instances at most. In comparison with Ono98 (JOX+JBSC), the population size of Ono98 (JOX+JBSC) is 3000 and our method is only 50. However, our method has better performance in all instances.

Table 8: Comparison results with GA/GLS approaches

instance	JOX+EDX	Ono98	YN97	Matt
abz7	670	680	678	672
abz8	683	685	686	683
abz9	686	702	697	703
la21	*1046	1050	*1046	1053
la24	*935	944	*935	938
la25	*977	984	*977	*977
la27	1236	1258	*1235	1236
la29	1167	1189	1166	1184
la38	*1196	1202	*1196	1201
la40	1224	1235	1224	1228

Table 9: Comparison results with other famous approximation methods

	best	Nowi	Aarts	Appl	YN96
abz7	670	—	668	668	665
abz8	683	—	670	687	675
abz9	686	—	691	707	686
la21	*1046	1047	1053	1053	*1046
la24	*935	939	*935	*935	*935
la25	*977	*977	983	*977	*977
la27	1236	1236	1249	1269	*1235
la29	1167	1160	1185	1195	1154
la38	*1196	*1196	1208	1209	1198
la40	1224	1229	1225	*1222	1228

Moreover, we compare our results to the famous approximation algorithms (Table 9). Nowi is the taboo search algorithm proposed in [Nowicki 93]; Aarts is the simulated annealing method proposed in [Aarts 94]; Appl is a shifting bottleneck procedure proposed in [Applegate 91]; YN96 is a combination of simulated annealing and shifting bottleneck procedure proposed in [Yamada 96]. In comparison with the famous approximation methods except YN96, our method also loses in two instances at most. Our method finds the optimum solution which YN96 could not find, however solutions our method found are behind YN96 in four instances.

From the view point of the computation time, the computation time for obtaining the optimum of $la24, la25$ in [Yamada 96] is $6.24 \times 10^3 sec, 4.14 \times 10^3 sec$ and our method is $1.75 \times 10^4 sec, 5.65 \times 10^3 sec$. We can not conclude that out method is superior to [Yamada 96] in consideration of the difference of the computer used for calculation.

Moreover, from the viewpoint of the quality of the obtained solution, our method finds good solution especially in middle size instances, such as *la* instances. However, our method is inferior to YN96 in large size instances such as *abz* instances.

6 Conclusion

In this paper, we proposed a new genetic algorithm using the combination of two crossovers, JOX and EDX. We proposed a method for analyzing the behavior of crossovers using a distance in a general discrete solution space and the definition of interpolation and extrapolation . Then we analyze JOX using proposed method and we confirmed that offspring generated by JOX exists mainly in the extrapolation area. To compensate for this fault of JOX, we designed a new crossover, EDX, that searches the extrapolation area and analyzed the behavior of EDX. We designed a GA for these two crossovers and applied the GA to the large-size benchmark problems. From this experiment, we were able to demonstrate its effectiveness.

There still exists several future works to be completed. Although precise comparison is difficult because the computers used for calculation is different, on the running time of the whole procedure, our method is relatively long compared with other approximation methods. Furthermore techniques for recovering this fault are desirable especially for the large-size instances. Finally, we would like to extend the proposed definition about extrapolation/interpolation to various problem domains, design EDX, and show the generality thereof.

Appendix

Let s_a, s_b be a schedule, where $|M|$ is a number of the machine and $|J|$ is the number of the job of the schedule s. Let $o(p,q)$ be an operation whose job name is p and whose machine name is q. Let j_i be a set of an operation that has the same job name, where i is a job name, that is, $j_i = \{o(i,k)|k = 1, ..., M\}, s = \{j_k|k = 1, ..., J\}$. Let l be a function that returns a number of the locus of an operation. For example, a operation $o(1,1)$ is located at the fifth locus of the job sequence of the job 1, then $l(o(1,1)) = 5$.

Here, we define a distance between j_i^a and $j_i^b, I2_i(s^a, s^b)$ such as the following, where j_i^a is a j_i of s_a. $I2_i(s^a, s^b) = \sum_{k=1}^{M} |l(o^a(i,k)) - l(o^b(i,k))|$

Moreover, we define I2 distance between two schedules s_a, s_b as follows: $I2(s^a, s^b) = \sum_{k=1}^{J} I2_k(s_a, s_b)$

When we regard $l(o(p,q))$ as a coordinate of a operation, we can define a coordinate of j_p, b_p as: $b_p = \frac{\sum_{k=1}^{M} l(o(p,k))}{M}$

References

[Aarts 94] Aarts, E., van Laarhoven, P., Lenstra, J and Ulder, N. : A Computational Study of Local Search Algorithms for Job-shop Scheduling, ORSA J. on Comput, Vol.6, No.2, pp.118-125 (1994).

[Applegate 91] Applegate, D. and Cook, W. : A Computational Study of the Job-shop Scheduling Problem, ORSA J. on Comput., Vol.3, No.2, pp.149-156 (1991).

[Giffler 60] Giffler, B. and Thompson, G. : Algorithms for Solving Production Scheduling Problems, Oper. Res., Vol.8, pp.487-503 (1960).

[Mattfeld 94] Mattfeld, D.C., Kopfer, H. and Bierwirth, C. : Control of Parallel Population Dynamics by Social-like Behavior of GA-individuals, 3rd PPSN (1994).

[Nowicki 93] Nowicki, E. and Smutnicki, C. : A Fast Taboo Search Algorithm for the Job Shop Problem, Institute of Engineering Cybernetics, Technical University of Wroclaw, Poland., Vol. Preprinty nr 8/93 (1993).

[Kita 99] Kita, H. and Yamamura, M. : A Functional Specialization Hypothesis for Designing Genetic Algorithms IEEE International Conference on Systems, Man, and Cybernetics (1999).

[Kobayashi 95] Kobayashi, S., Ono, I. and Yamamura, M. : An Efficient Genetic Algorithm for Job-shop Scheduling Problems, Proceedings of the 6th International Conference on Genetic Algorithms, pp.506-511 (1995).

[Nagata 99] Nagata, Y., Kobayashi, S. : An analysis of Edge Assembly Crossover for the Traveling Salesman Problem, IEEE International Conference on Systems, Man and Cybernetics (1999).

[Ono 96] Ono, I., Yamamura, M. and Kobayashi, S. : A Genetic Algorithm for Job-shop Scheduling Problems Using Job-based Order Crossover, Proceedings of 1996 IEEE International Conference on Evolutionary Computation, pp.547-552 (1996).

[Ono 97] Ono, I. and Kobayashi, S. : A Real-Coded Genetic Algorithm for Function Optimization Using Unimordal Normal Distribution Crossover, Proceedings of the 7th International Conference on Genetic Algorithms, pp.246-253 (1997).

[Ono 98a] Ono, I. and Kobayashi, S. : An Evolutionary Algorithm for Job-Shop Scheduling Problems Using the Inter-Machine Job-Based Order Crossover, Journal of Japanese Society for Artificial Intelligence, Vol.13, No.5, pp.780-790 (1998).

[Ono 98b] Ono, I. and Kobayashi, S. : A Genetic Algorithm Taking Account of Characteristics Preservation for Job-shop Scheduling Problems, Proceedings of the 5th conference on Intelligent Autonomous Systems, pp.711-718 (1998) .

[Shi 96] Shi, G. and Sannomiya, N. : A New Encoding Scheme for Solving Job-Shop Problems by Genetic Algorithms, Proceedings of 35th IEEE Conf. on Decision and Control, pp.4395-4400 (1996).

[Satoh 96] Satoh, H., Yamamura, M., and Kobayashi, S. : Minimal Generation Gap Model for GAs Considering Both Exploration and Exploitation, Proc. IIZUKA '96, pp.494-497 (1996).

[Yamada 96] Yamada, T. and Ryohei, N. : Job-shop Scheduling by Simulated Annealing Combined with Deterministic Local Search, Kluwer Academic Publishers, MA, USA (1996).

[Yamada 97] Yamada, T. and Ryohei, N. : Genetic Algorithms for Job-Shop Scheduling Problems, Proc. of Modern Heuristic for Decision Support,pp.67-81 (1997).