Journal of the Operations Research
Society of Japan
Vol. 37, No. 2, June 1994

GENETIC ALGORITHMS FOR SINGLE MACHINE JOB SCHEDULING WITH COMMON DUE DATE AND SYMMETRIC PENALTIES

Chae Y. Lee
Korea Advanced Institute of Science and Technology

(Received October 26, 1992; Revised January 10, 1994)

Abstract A single machine n-job scheduling problem is examined to minimize sum of absolute deviations of completion times from a common due date. Simple and hybrid genetic Algorithms are developed by investigating basic operators for the applications of job sequencing problems. For the simple genetic algorithm two heuristic crossover schemes: Algorithm VASX and Algorithm VADX are developed based on important properties of the scheduling problem. Local Improvement techniques are considered to enhance the solution quality of the simple genetic algorithm. The power of a genetic algorithm is illustrated by comparing the performance with branch and bound procedure.

1 Introduction

Recent research on job scheduling problems has been focused on the earliness and tardiness penalties as the performance criterion. This criterion gives penalties to jobs which are completed before the due date as well as those completed after the due date. Just-In-Time (JIT) production is a typical application of the scheduling problem. In a JIT scheduling, jobs that complete early must be held in finished goods inventory until their due date, while jobs that complete after their due date may cause loss of customers' goodwill or cost of express delivery. Even the simplest model that deals with earliness and tardiness penalties is known to be NP-complete [8]. This is why most of the solution techniques for the problems rely on branch and bound procedures or heuristic methods.

One emerging heuristic search procedure for solutions to combinatorially complex problems is the genetic algorithm [5, 6, 7]. Inspired by biological evolution, genetic algorithms propagate new solutions from parent solutions via some stochastic operators. Solutions with high fitness values survive and those with low fitness values die off generation by generation. Thus, the evolutionary method has proved its effectiveness in the problems where the size and complexity of the search space preclude conventional approaches. The application of genetic algorithms to the O.R. related problems include the travelling salesman problem [5, 8, 10], bin packing [12], graph partitioning [11], job scheduling [3, 4] and many other optimization problems.

The job sequencing problem to be discussed in this paper is a typical paradigm of a wide class of problems having complexity due to the combinatorial explosion. Thus, the power of genetic algorithms combined with the inherent problem specific properties will effectively deal with the obstacle of the explosion problem.

In section 2, the single machine job sequencing problem and the problem-specific properties related to the earliness and tardiness penalties are discussed. Section 3 discusses the application of genetic algorithms for the job scheduling problem. Two different crossover schemes are proposed that are based on the properties introduced in Section 2.

The local search method is also examined to improve the solution quality. Computational results of the simple and hybrid genetic algorithms are illustrated in Section 4. The performance of of the two heuristic crossover methods is investigated. The genetic algorithm based procedure is compared to an existing solution procedure. Finally, in Section 5 we conclude this paper with some recommendation for further research.

2 Job Scheduling with Symmetric Earliness and Tardiness Penalties

Job sequencing problems with earliness and tardiness penalties are extensively discussed by Baker and Scudder [2]. The problem to be discussed in this paper is n-job scheduling on a single machine to minimize the sum of earliness and tardiness penalties.

Let p_i = processing time of job i

C_i = completion time of job i

and d = common due date.

Then, our objective is to obtain a schedule S^* that minimizes

$$f(S) = \sum_{i=1}^{n} |C_i - d| \tag{2.1}$$

We assume that all jobs are available for processing at time t=0 and are numbered as

$$p_1 \leq p_2 \leq \cdots \leq p_n \tag{2.2}$$

In the literature the above problem is classified into two cases by employing

$$\Delta = \begin{cases} p_1 + p_3 + \cdots + p_n & \text{if } n \text{ is odd} \\ p_2 + p_4 + \cdots + p_n & \text{if } n \text{ is even} \end{cases} \tag{2.3}$$

The problem is unrestricted for $d \geq \Delta$ and restricted otherwise. Bagchi, Chang and Sullivan [1] proposed a polynomial algorithm for the unrestricted case by focusing on the secondary criterion of minimization of the total processing time of jobs that complete on or before the due date.

However, the restricted version of the problem is known to be NP-complete [2]. Bagchi, Sullivan and Chang [1] examined several properties related to the optimal schedule, provided a branching procedure and illustrated the computational results with relatively small sized problems. A systematic branch and bound scheme is proposed by Szwarc [14]. In the algorithm he reduced the number of tested nodes by exploiting properties the optimal solution has. The performance of the branch and bound procedure is examined for problems with n = 15, 20 and 25.

In this paper we discuss the restricted version of job scheduling problem which has not been solved by anything other than enumerative algorithms. For the application of genetic algorithms we introduce some important properties that are due to [1, 14].

Property 1 (VAS): The optimal schedule is V-shaped around the job with the shortest processing time ; In the optimal schedule the jobs preceding and succeeding the shortest

job are in LPT (Longest Processing Time) and SPT (Shortest Processing Time) orders, respectively.

Property 2 (VAD): The optimal schedule is V-shaped around the due date ; In the optimal schedule the jobs that are completed on or before the due date are processed in LPT order, and the jobs that are started on or after the due date are processed in SPT sequence.

Property 3: For $d \leq (p_1 + p_2)/2$ the SPT sequence is optimal.

3 Genetic Algorithms for the Job Scheduling Problem

The framework of the genetic algorithm for the single machine job scheduling problem with earliness and tardiness penalties is discussed together with the recombination and reproduction operators. Two heuristic crossover schemes, i.e., Algorithm VASX and Algorithm VADX are developed that lead to effective combination of partial solutions on two distinct chromosomes. Local improvement is also considered for better performance in a single chromosome.

3.1 Operators in Genetic Algorithm

One of the major difficulties arising in the application of genetic algorithms to various optimization problems is to encode potential solutions into a population of chromosomes. The chromosomes might be bit strings, real number lists, permutation of elements, rules or many others. In this research since the problem is to find the best sequence of n jobs, chromosomes are represented as the permutation of n job numbers.

Essentially, genetic algorithms are processed by three basic operators; reproduction, crossover and mutation. Among them the crossover operator is the most important issue in genetic algorithms. The crossover operator causes exchange of genetic material between two parents, while mutation operator causes local alteration in a single chromosome. In Section 3.2 and 3.3 two heuristic crossover schemes are developed which greatly accelerate the search early in the evolution of population.

In Section 3.4 local search in a single chromosome is examined to improve the fitness of offspring schedules. This local search method either swaps two jobs or partially reorders jobs of a given sequence. Thus, the local search in this study can be considered as a variant form of mutation. However, the local improvement is not triggered stochastically.

Another important operator in the genetic algorithm is the reproduction by which a population of parents are selected for next generation. Theoretically, parents are selected with probability biased toward chromosomes with better evaluations. In job scheduling problems, since each chromosome is encoded as a sequence of n jobs, evaluations may have too broad spectrum with the increasing number of jobs. Thus, in this research reproduction is performed by selecting best N chromosomes after evaluation of the $3N/2$ schedules; N parent schedules in the preceding generation and $N/2$ offsprings generated by the recombination operators.

Figure 1 shows the framework of the genetic algorithm that is employed for the job scheduling problem in this research. Parent schedules with high evaluations are selected and remaining chromosomes are discarded by the reproduction operator. These parent schedules then generate new offspring chromosomes via heuristic crossover and local improvement. Finally, the evaluation process computes the fitness values of newly generated chromosomes for the reproduction in the next generation. This procedure is repeated and the population of schedules evolves generation by generation.

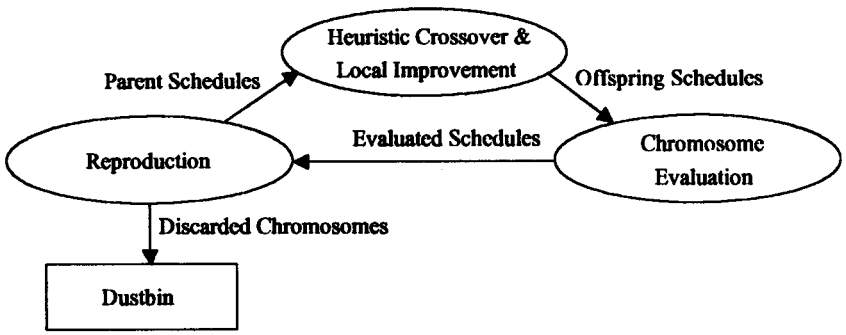

Figure 1. The Framework of Genetic Algorithm for the Job Sequencing Problem.

3.2 V-schedule Around the Shortest Job Crossover (VASX)

The VASX is based on Property 1 of Section 2. The basic idea of this crossover scheme is to produce an offspring with an approximate V-schedule around the shortest job. An offspring is constructed from two parents as follows: Compare the first job of each parent and choose the job with longer processing time as the first gene a_1 of the new chromosome. Continue to extend the schedule by comparing in each parent the job which is next to the last determined gene a_1 of the new chromosome. Choose a job the processing time of which is close to the last determined gene in preference to V-schedule. If the job would construct a cycle i.e., the job already exists in the new chromosome, then check if the other job constructs a cycle. If the latter does not construct a cycle, extend the schedule with the job. Otherwise, if both of the jobs construct cycles, extend the sequence with the closest among the remaining jobs the sequence of which are not determined in the new chromosome. Continue the process until a complete schedule $(a_1, a_2, \cdots, a_n)$ is generated. The selection of the two parents for the mating is performed by the roulette wheel method [7]. The algorithm for this heuristic crossover operation is summarized below.

Algorithm VASX

Step 0 (Selection of the first gene)
Let i=1 and compare the first genes of the two parent chromosomes. Choose the one with the longer processing time, i.e. higher job number, as the first gene a_1 of the child.

Step 1 (Stopping, Order of Ascending/Descending)
If $i = n+1$, then stop. The crossover is complete. If $i \geq k$ for which $a_k = 1$, then go to Step 3. Otherwise, go to Step 2.

Step 2 (Descending Order)
Let $i = i+1$. From each parent select the very next gene to a_{i-1} of the child chromosome. If a_{i-1} is the last gene of a parent, then select the first gene of the parent. Let a_i be the gene determined as follows and go to Step 1.

2.1) If both of the genes already exist in the new chromosome, then choose the highest gene which is lower than a_{i-1}.

2.2) If one constructs a cycle and the other does not, then choose the latter as a_i.

2.3) If neither of the genes constructs a cycle and there exist genes lower than a_{i-1}, then choose the higher one among them as a_i.

2.4) If neither of the genes constructs a cycle and neither of them is lower than a_{i-1}, then choose the lower one as a_i.

Step 3 (Ascending order)

Let $i = i+1$. From each parent select the very next gene to a_{i-1} of the child chromosome. If a_{i-1} is the last gene of a parent, then select the first gene of the parent. Let a_i be the gene determined as follows and go to Step 1.

3.1) If both of the genes construct cycles, then among the remaining genes, choose the lower gene which is higher than a_{i-1}. If there are no such genes, then select the lowest as a_i.

3.2) If one constructs a cycle and the other does not, then choose the latter as a_i.

3.3) If neither of the genes constructs a cycle and there exist genes higher than a_{i-1}, then choose the lower one among them as a_i.

3.4) If neither of the genes constructs a cycle and neither of them is higher than a_{i-1}, then choose the lower one as a_i.

Example 1 The following example illustrates the algorithm VASX in generating a child chromosome.

Chromosome 1: 9 8 4 5 6 7 1 3 2 10
Chromosome 2: 8 6 1 2 3 10 5 9 4 7
Child : 9 8 6 1 2 3 10 5 7 4

The first gene of a child is determined by the first job with longer processing time of the two parents. Thus, job 9 is selected as the first gene. Then job 8 of chromosome 1 and job 4 of chromosome 2 are compared and the job with longer processing time is selected to follow job 9 by Step 2.3. Thus, job 8 follows job 9. The procedure then continues by Step 2.3 and job 6 and job 1 of chromosome 2 follow. Since job 1 with the shortest processing time is selected, the algorithm now proceeds in ascending order. Jobs 2, 3 and 10 of chromosome 2 follow job 1 in the child chromosome by Step 3.3. Then by Step 3.2 Job 10 is followed by job 5 of chromosome 2, since job 9 of chromosome 1 constructs a cycle. Job 7 is then selected to follow job 5 by Step 3.1 since both job 6 and 9 of the two parents construct cycles. Finally, job 4 constitutes the last gene of the child chromosome.

The VASX has the tendency to combine good partial schedules from both parents. Step 2.3 in the algorithm keeps good LPT orders, while Step 3.3 maintains SPT orders inherent in the two parents. The heuristic crossover always generates a feasible schedule even though it may not be a perfect V-schedule. Note that the V-schedule may be broken down by Step 2.2, 2.4, 3.2 and 3.4. However, these steps maintain a feasible schedule by avoiding cycles in offspring chromosomes. The performance of this heuristic crossover is discussed in Section 4.

3.3 V-schedule Around the Due date Crossover (VADX)

This heuristic crossover is based on the Property 2 of Section 2. In the VADX, jobs are sequenced as in the VASX up to due date d. One of the differences in the VADX is that whenever a job is selected in the new chromosome, it applies Property 3 to determine whether to sequence the remaining jobs in SPT or not. If Property 3 does not hold, the process is continued up to due date. Otherwise, the remaining jobs are sequenced in SPT and the heuristic crossover is complete. The procedure is justified with the following Lemma.

Lemma 1 Let $S = \sigma r \pi$ be a n-job schedule, where σ and π are two arbitrary sequences. Also, let jobs u, v $\in \pi$ be such that $p_u \leq p_v \leq p_i$ for any $i \in \pi$. If $0 \leq d - C_r \leq (p_u + p_v)/2$, then for σr fixed $f(S)$ is minimized by sequencing jobs in π in SPT.
Proof See Property 3.

However, in the crossover procedure suppose that a job r is spread over the due date, i.e., $C_\sigma < d$ and $C_r > d$. In this case Lemma 1 does not hold. Also, suppose that jobs in σ and r are sequenced by the algorithm VASX and the sequence of remaining jobs in π are not determined. If the tardiness penalty of job r is less than the earliness penalty of the previous job of job r, then the sum of total penalties of the sequence $\sigma s \pi'$, where $s \in \pi$ is replaced with r and π' is generated by the replacement, may be decreased by the selection of an appropriate job s. However, if the tardiness penalty of job r is greater than or equal to the earliness penalty, then the sum of total penalties can be decreased by replacing job r with any job s with $p_s < p_r$. These properties are formalized as follows.

Consider two n-job schedules $S = \sigma r \pi$ and $S' = \sigma s \pi'$, where σ and π are arbitrary sequences and π' is generated by replacing job s in π with job r. Let C_i and C'_i be respectively the completion times of job i in schedule S and S'.

Lemma 2 Let S be a sequence such that $C_r - p_r < d < C_r$. If $d - (C_r - p_r) > C_r - d$, then $f(S') < f(S)$ for

$$p_r - 2(C_r - d) < p_s < p_r. \tag{3.1}$$

Proof If $C'_s \geq d$, then $C'_s - C_r = p_s - p_r$ and $C_k - C_r = p_s - p_r$ for job k which is sequenced after s and before r in S'. Thus $f(S') < f(S)$ for $p_s < p_r$.
If $C'_s < d$ then the difference of two penalties of job s in S' and job r in S is

$$|C'_s - d| - |C_r - d| = d - C_r + p_r - p_s - (C_r - d) = p_r - p_s - 2(C_r - d) \tag{3.2}$$

Thus, $|C'_s - d| - |C_r - d| < 0$ for $p_r - 2(C_r - d) < p_s$. Also, for job k which is sequenced after s and before r in S', $|C'_k - d| - |C_k - d| < 0$. Therefore, $f(S') < f(S)$ for $p_r - 2(C_r - d) < p_s$.

Lemma 3 Let S be a sequence such that $C_r - p_r < d < C_r$. If $d - (C_r - p_r) \leq C_r - d$, then $f(S') < f(S)$ for

$$p_s < p_r. \tag{3.3}$$

Proof Clear from the fact that for $p_s < p_r$, $|C'_s - d| - |C_r - d| < 0$, if $d - (C_r - p_r) \leq C_r - d$.

Theorem 1 Let $S = \sigma r \pi$ be a sequence such that $d - (C_r - p_r) > C_r - d > 0$. Suppose that the sequence of jobs in σ is fixed, and that there exist jobs that satisfy Equation (3.1). If a job s with $C'_s - d \geq 0$ is considered for $S' = \sigma s \pi'$, then $f(S')$ is minimized by the following schedule:

a) Select a job $s \in \pi$ which minimizes $C'_s - d$
b) Sequence the job in π' in SPT order

Proof Clear from Lemma 2.

Clearly, if there are no jobs which satisfy Equation (3.1), then $S = \sigma r \pi$ is minimized by sequencing jobs in π in SPT order. However, when $C'_s - d < 0$ for the job s, then no

sequencing rule is guaranteed for the minimization of $f(S')$ except when $d - C'_s$ and jobs u and v in π' satisfy the condition of Lemma 1.

Theorem 2 Let $S = \sigma r \pi$ be a sequence such that $0 < d - (C_r - p_r) \leq C_r - d$. Suppose that the sequence of jobs in σ is fixed and that there exist jobs which satisfy Equation (3.3). If a job s with $C'_s - d \geq 0$ is considered for $S' = \sigma r \pi'$, then $f(S')$ is minimized by the following schedule:

a) Select a job $s \in \pi$ that minimizes $C'_s - d$.

b) Sequence the jobs in π' in SPT order.

Proof Clear from Lemma 3.

Now, suppose that a subsequence is constructed by VASX and that a job r is spread over the due date d. Then the schedule which minimizes total penalties of the remaining jobs can be constructed by Theorem 1 and Theorem 2 under the assumption that a job s with $C'_s - d \geq 0$ is swapped for the job r. For the case in which this assumption is relaxed, an approximate scheduling rule can be adopted where a job s with minimum $|C'_s - d|$ is substituted for job r. Based on Lemma 1, Theorem 1 and Theorem 2 the following heuristic crossover algorithm is proposed.

Algorithm VADX

Step 0 (Selection of the first gene)

Let i=1 and compare the first genes of the two parent chromosomes. Choose the one with longer processing time as the first gene a_1 of the child. If the processing time of a_1 is greater than the due date d, then let $r = a_1$ and go to Step 2. Otherwise, go to Step 1.

Step 1 (Up to due date)

Let $i = i+1$. From each parent selects the very next gene to a_{i-1} of the child chromosome. If a_{i-1} is the last gene of a parent, then select the first gene of the parent. Let a_i be the gene determined as follows:

1.1) If both of the genes already exist (construct cycles) in the new chromosome, then choose the highest gene which is lower than a_{i-1}.

1.2) If one construct a cycle and the other does not, then choose the latter as a_i.

1.3) If neither of the genes constructs a cycle and there exist genes lower than a_{i-1}, then choose the higher one as a_i.

1.4) If neither of the genes constructs a cycle and neither of them is lower than a_{i-1}, then choose the lower one as a_i. Let $r = a_i$. If $C_r - d$ is less than or equal to the mean of the two shortest genes among those position is not determined, then go to Step 3. If $C_r > d$, go to Step 2. Otherwise, repeat this step.

Step 2 (When gene r is spread over the due date)

If $d - (C_r - p_r) > C_r - d$, then select a gene s which satisfies $p_r - 2(C_r - d) < p_s < p_r$ and minimizes $|d - C_s|$. Let $a_i = s$. If there is no such genes let $a_i = r$. Go to Step 3. If $d - (C_r - p_r) \leq C_r - d$, then select a gene s which satisfies $p_s < p_r$ and minimizes $|d - C_s|$. Let $a_i = s$. If there is no such genes let $a_i = r$. Go to Step 3.

Step 3 Sequence the remaining genes in nondecreasing order and complete the schedule.

The two crossover procedures Algorithm VADX and Algorithm VASX are compared as follows:

Theorem 3 Let S_S and S_D be n-job schedules obtained from two parent schedules by Algorithm VASX and Algorithm VADX respectively. $f(S_D) \leq f(S_S)$.
Proof Let $S_S = \sigma r \pi$. Also let σr be the subsequence obtained by Step 1 of Algorithm VADX. Then the following three cases result by Algorithm VADX for the offspring schedule S_D.

i) $S_D = \sigma r \pi'$, where π' is a SPT sequence of jobs in π.
ii) $S_D = \sigma s \pi'$, where $C_s - d \geq 0$ and π' is sequenced in SPT.
iii) $S_D = \sigma s \pi'$, where $C_s - d < 0$ and π' is sequenced in SPT.

Case (i) results when there are no jobs that satisfy Equation (3.1) or (3.3). In this case it is clear by Lemma 1 that $f(S_D) \leq f(S_S)$. Case (ii) and (iii) result when there exists at least one job that satisfies Equation (3.1) or (3.3). In case (ii) $f(S_D) \leq f(S_S)$ by Theorem 1 and Theorem 2. Finally, case (iii) results when jobs satisfies Equation (3.1) or (3.3). Thus, $f(S_D) \leq f(S_S)$ by Lemma 2 and Lemma 3.

3.4 Local Improvement

The two heuristic crossover schemes developed in Section 3.3 are based on the V-schedules. However, in Algorithm VASX the V-schedule may be violated in either ascending or descending order. The same result is expected in applying Algorithm VADX even though the criterion of V-schedule is different. In Algorithm VADX jobs that complete before or on the due date may violate the V-schedule.

To improve the fitness of the offspring chromosomes a local search method is incorporated to the simple genetic algorithm. In this research the following local improvement is considered for the offspring generated by each heuristic crossover operator.

(1) Make a V-schedule around the shortest gene for the offspring chromosomes constructed by the Algorithm VASX.

(2) For the offspring constructed by the algorithm VADX, first create a V-schedule around the due date and then make it V around the shortest gene.

The local search in (2) has an effect of enforcing the shortest job around the due date. However, when a job is spread over the due date, the resulting chromosome may not be a V-schedule around the shortest job. Thus, the chromosome needs to be made V around the shortest job which reflects both Property 1 and Property 2 of Section 2.

4 Computational Results and Discussion

The performance of simple and hybrid genetic algorithms are presented. The power of hybrid genetic algorithm is illustrated by comparing the efficiency with a branch and bound procedure.

4.1. Performance of the Simple Genetic Algorithms

In this section we discuss the computational results of the simple genetic algorithm based on the two heuristic crossover schemes VASX and VADX. No mutation is employed in the algorithm. The word "simple" is used in the sense that no local search method is employed in the algorithm. The two genetic algorithms are implemented using C on

the IBM PS/2 model 55SX with the coprocessor. In each algorithm the population of size N=100 is considered with 100 % crossover rates. Each algorithm stops when the improvement of the average penalty in one generation is less than 0.01 % of that in the preceding generation. Table 1 shows the performance of the two genetic algorithms with the nine small examples [1,14]. The performance of the best chromosome generated by each algorithm is very close to the optimal solution. The error bound is within 2 %.

4.2. Performance of the Hybrid Genetic Algorithm

As discussed in Section 3.4, the local search clearly improves the fitness of a child chromosome with the aid of the V-schedule property. In this section we examine the performance of the hybrid genetic algorithm which links the local improvement to the simple genetic algorithm based on the VADX. The hybrid algorithm is compared with the Branch and Bound Algorithm by Szwarc [14] with medium size problems. The two algorithms are implemented in the same environment as in Section 4.1. The parameters used in the hybrid algorithm are also same as in the simple genetic algorithm.

Table 1. Performance of the Simple GA

Problem	# of	Type of	Best Chromosome	Best		Average		Worst	
#	Jobs	Crossover	obtained	Penalty	Generation *	Penalty	Generation **	Penalty	Generation **
1	6	VASX	5 4 1 2 3 6	189	203	13	207	13	
		VADX	5 4 1 2 3 6	189	1	189	9	189	9
		Optimum	5 4 1 2 3 6	189					
2	6	VASX	6 3 2 1 4 5	355	0	355	16	355	16
		VADX	6 3 1 2 4 5	358	2	370	14	374	14
		Optimum	6 3 2 1 4 5	355					
3	6	VASX	6 4 1 2 3 5	387	5	392	14	393	14
		VADX	6 5 1 2 3 4	393	0	393	5	393	5
		Optimum	6 4 1 2 3 5	387					
4	6	VASX	5 4 1 2 3 6	265	1	267	16	268	16
		VADX	5 4 1 2 3 6	265	1	265	13	265	13
		Optimum	5 4 1 2 3 6	265					
5	9	VASX	9 6 5 3 2 1 4 7 8	274	4	274	16	274	16
		VADX	9 7 4 2 1 3 5 6 8	275	3	279	10	281	10
		Optimum	9 7 3 2 1 4 5 6 8	274					
6	14	VASX	14 12 8 8 5 2 4 3 1 6 7 10 11 13	1110	9	1115	19	1116	19
		VADX	13 12 10 8 5 4 1 2 3 6 7 9 11 14	1093	3	1093	21	1093	21
		Optimum	13 11 10 8 5 4 1 2 3 6 7 9 12 14	1092					
7	14	VASX	14 13 10 6 5 4 3 1 2 7 8 9 11 12	1623	13	1623	29	1623	29
		VADX	14 13 9 7 6 4 3 1 2 5 8 10 11 12	1612	5	1612	18	1612	18
		Optimum	14 12 10 8 6 3 1 2 4 5 7 9 11 13	1603					
8	14	VASX	14 13 10 5 4 2 1 3 6 7 8 9 11 12	1766	8	1779	24	1780	24
		VADX	13 11 9 7 6 4 1 2 3 5 8 10 12 14	1743	4	1743	24	1743	24
		Optimum	13 11 9 7 5 4 1 2 3 6 8 10 12 14	1742					
9	14	VASX	14 11 6 4 3 1 2 5 7 8 9 10 12 13	1845	5	1845	28	1845	28
		VADX	12 9 8 7 2 1 3 4 5 6 10 11 13 14	1823	7	1831	17	1832	17
		Optimum	12 9 7 6 3 2 1 4 5 8 10 11 13 14	1820					

* represents the generation first appeared
** represents the last generation

A set of ten test problems are generated for n=20, 30 and 40 respectively. In each set three different due dates 0.3Δ, 0.6Δ and 0.9Δ are considered. In each problem the processing times p_i are randomly generated from a discrete uniform distribution over [1, 100].

The results of the 20, 30 and 40 job scheduling problems are shown in Table 2, 3 and 4 respectively. In each table the following due dates are used:

$$d = \begin{cases} 0.3\Delta & \text{for problems 1,4,7 and 10} \\ 0.6\Delta & \text{for problems 2,5 and 8} \\ 0.9\Delta & \text{for problems 3,6 and 9} \end{cases} \tag{4.1}$$

From the three tables it can be easily seen that the solution quality of the hybrid genetic algorithm is comparable to the optimal solution by the branch and bound method. In each problem the error bound of the best chromosome is within 2 % of the optimal solution. Even the worst performance does not exceed the optimal schedule by more than 3 %.

From the point of computational demand the genetic algorithm outperforms the branch and bound method. The CPU times (in seconds) of the partial enumaration procedure are largely dependent on the ratio of d/Δ. However, the genetic algorithm seems to be independent of the tightness of the due dates. The three tables also indicate that growth of CPU seconds of the branch and bound algorithm is exponential to the number of jobs while that of the genetic algorithm is expected to be linear.

The performance of heuristic crossover and local search in the genetic algorithm is also compared. Figure 2 shows the portions of the two operators to the cost improvement in problems with 40 jobs. Very large portion of the cost improvement is due to the crossover operator VADX. However, the contribution by local search is noticeable as the due date becomes less tight.

Finally, the performance of the evolution strategy is further investigated with a set of big problems. Table 5 shows the results of n = 50, 60 and 80. Unfortunately, the solution quality of the genetic algorithm could not be examined due to the combinatorial explosion of the branch and bound procedure. However, notice from the table that the increase of CPU seconds of average chromosomes slows down despite of the increased problem complexity.

5 Conclusions

Genetic algorithms are developed for n-job scheduling problem in a single machine. The basic framework and operators are discussed. The crossover operator which is the most important issue in genetic algorithms is extensively examined for the sequencing problem with symmetric earliness and tardiness penalties. Two heuristic crossover schemes: Algorithm VASX and Algorithm VADX are developed to produce better chromosomes that fit well to the V-schedule. The simple genetic algorithms based on the two crossover methods produced near optimal schedules the error bound of which are within 2 % of the optimal solutions. The performance of the Algorithm VADX is proved to be better than the Algorithm VASX.

A hybrid genetic algorithm is also developed by incorporating the local search technique to improve the solution quality. The local improvement has the effect of forcing

Table 2. Comparison of GA and B&B (20 Jobs)

Problem #	GA						B & B	
	Best		Average		Worst			
	Penalty	CPU time	Penalty	CPU time	Penalty	CPU time	Penalty	CPU time
1	4201	0.93	4214	15.61	4216	15.61	4178	1.21
2	3263	9.49	3315	17.32	3319	17.32	3235	7.25
3	2823	6.8	2838	21.06	2840	21.06	2806	10.0
4	5617	4.68	5673	16.96	5680	16.96	5594	1.43
5	4337	1.1	4365	26.41	4371	26.41	4301	10.65
6	3764	3.18	3773	21.49	3776	21.49	3721	22.69
7	4726	2.63	4764	15.06	4767	15.06	4666	1.04
8	3643	2.75	3693	11.21	3696	11.21	3606	8.35
9	3161	6.47	3188	15.35	3190	15.35	3115	11.69
10	5009	7.29	5068	13.94	5072	13.94	4992	1.65

Table 3. Comparison of GA and B&B (30 Jobs)

Problem #	GA						B & B	
	Best		Average		Worst			
	Penalty	CPU time	Penalty	CPU time	Penalty	CPU time	Penalty	CPU time
1	10787	1.16	10869	14.04	10884	14.04	10728	35.16
2	8365	5.39	8415	19.66	8417	19.66	8324	675.19
3	8230	7.42	8271	22.25	8278	22.25	8180	7059.74
4	13708	2.09	13728	25.5	13730	25.5	13650	52.57
5	10666	1.92	10668	15.71	10669	15.71	10593	1689.51
6	9263	6.7	9263	21.15	9263	21.15	9226	6286.93
7	10629	7.42	10731	17.68	10749	17.68	10565	41.19
8	8395	1.21	8395	12.52	8395	12.52	8218	1015.14
9	7224	22.47	7248	23.67	7254	23.67	7182	4642.85
10	12244	4.65	12269	19.69	12272	19.69	12153	61.85

Table 4. Comparison of GA and B&B (40 Jobs)

Problem #	GA						B & B	
	Best		Average		Worst			
	Penalty	CPU time	Penalty	CPU time	Penalty	CPU time	Penalty	CPU time
1	22178	3.08	22326	18.9	22356	18.9	22143	1466.35
2	17432	14.08	17432	35.88	17432	35.88	17308	76002.773
3	15207	13.44	15219	20.26	15220	20.26		>>
4	21625	1.26	21806	19.05	21833	19.05	21506	1975.78
5	16963	10.71	16963	31.37	16963	31.37	16768	94177.719
6	14848	12.63	14848	20.1	14848	20.1		>>
7	22103	2.3	22253	13.12	22257	13.12	22019	1697.25
8	17286	20.85	17286	29.46	17286	29.46	17183	170458.969
9	15211	6.96	15211	18.56	15211	18.56		>>
10	19358	5.93	19443	17.62	19450	17.62	19230	2298.08

>> more than 48 hours of CPU time

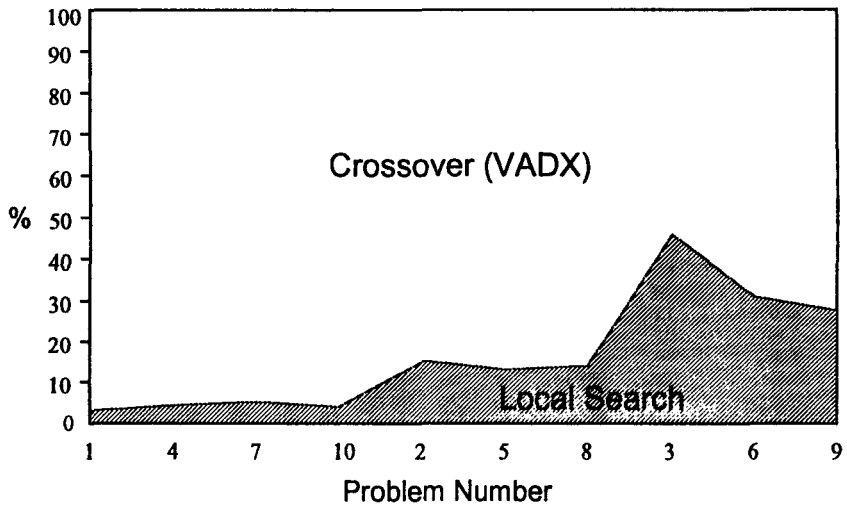

Figure 2. Portions of the Cost Improvement by the Two Operators

Table 5. Performance of GA with Large Sized Problems

Problem	Performance	n = 50		n = 60		n = 80	
#		Penalty	CPU time	Penalty	CPU time	Penalty	CPU time
1	Best	29223	6.64	48738	2.69	74905	11.64
	Avg.	29422	21.19	48901	31.2	74905	34.69
	Worst	29467	21.19	48918	31.2	74908	34.69
2	Best	23098	10.9	38545	1.2	59779	12.93
	Avg.	23227	18.36	38545	25.8	59788	19.02
	Worst	23237	18.36	38545	25.8	59789	19.02
3	Best	20140	13.91	35327	7.15	52300	6.37
	Avg.	20152	27.96	35327	18.2	52300	21.64
	Worst	20157	27.96	35327	18.2	52300	21.64
4	Best	27806	1.05	45714	1.32	93865	2.41
	Avg.	27817	31.47	44869	32.76	93865	2.41
	Worst	27820	31.47	45921	32.76	93865	14.17
5	Best	21782	9.77	35997	19.64	73379	13.46
	Avg.	21859	30.81	35997	32.27	74562	22.8
	Worst	21862	30.81	35997	32.27	74599	22.8
6	Best	19391	12.75	31798	17.49	64564	3.72
	Avg.	19391	20.88	31871	36.28	64565	24.76
	Worst	19391	20.88	31874	36.28	64568	24.76
7	Best	30012	3.24	47018	15.84	80057	14.01
	Avg.	30178	19.79	47291	28.58	80057	35.96
	Worst	30188	19.79	47303	28.58	80057	35.96
8	Best	23579	1.15	37139	1.15	62835	1.54
	Avg.	23724	20.98	37139	30.0	62837	30.31
	Worst	23750	20.98	37139	30.0	62858	30.31
9	Best	20725	10.6	32623	12.09	55465	3.83
	Avg.	20730	19.1	32799	18.45	55465	21.79
	Worst	20734	19.1	32825	18.45	55465	21.79
10	Best	30188	5.6	41999	2.35	70161	14.1
	Avg.	30433	14.16	42256	17.33	70237	33.8
	Worst	30480	14.16	42275	17.33	70250	33.8

the shortest job near to the due date, which make the sequence near V-schedule around both the due date and the shortest job. The power of the hybrid genetic algorithm is illustrated with medium and big sized problems. It outperforms the branch and bound method in CPU times. The genetic algorithm solved the 80 job scheduling problems in 20 to 30 seconds which could not be handled with the partial enumeration method due to the combinatorial explosion. The error bound of the solution by the hybrid method is less than 2 % in medium sized (n=20, 30, 40) problems.

References

[1] U. Bagchi, R. S. Sullivan and Y. L. Chang, "Minimizing Mean Absolute Deviation of Completion Times about a Common Due Date", *Naval Research Logistics Quarterly*, Vol. 33, pp. 227-240, 1986.

[2] K. R. Baker and G. D. Scudder, "Sequencing with Earliness and Tardiness Penalties: A Review", *Operations Research*, Vol. 38, pp. 22-36, 1990.

[3] G. A. Cleveland and S. F. Smith, "Using Genetic Algorithms to Schedule Flow Shop Releases", *Proceeding of an International Conference on Genetic Algorithms and Their Applications*, pp. 160-169, 1989.

[4] L. Davis, "Job Shop Scheduling with Genetic Algorithms", *Proceeding of an International Conference on Genetic Algorithms and Their Applications*, pp. 136-140, 1985.

[5] L. Davis, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.

[6] F. Glover and H. J. Greenberg, "New Approaches for Heuristic Search: A Bilateral Linkage with Artificial Intelligence", *European Journal of Operational Research*, Vol. 39, pp. 119-130, 1989.

[7] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.

[8] J. J. Grefenstette, R. Gopal, B. J. Rosmaita and D. Van Gught, "Genetic Algorithms for the Travelling Salesman Problem", *Proceeding of an International Conference on Genetic Algorithms and Their Applications*, pp. 160-168, 1985.

[9] N. G. Hall, W. Kubiak and S. P. Sethi, "Earliness-Tardiness Scheduling Problems, II: Deviation of Completion Times about a Restrictive Common Due Date", *Operations Research*, Vol. 39, pp. 847-856, 1991.

[10] P. Jog, J. Y. Suh and D. V. Gught, "The Effects of Population Size, Heuristic Crossover and Local Improvement on a Genetic Algorithm for the Travelling Salesman Problem", *Proceeding of an International Conference on Genetic Algorithms and Their Applications*, pp. 110-115, 1989.

[11] G. V. Laszewski, "Intelligent Structural Operators for the K-way Graph Partitioning Problem", *Proceeding of an International Conference on Genetic Algorithms and Their Applications*, pp. 45-52, 1991.

[12] D. Smith, "Bin Packing with Adaptive Search", *Proceeding of an International Conference on Genetic Algorithms and Their Applications*, pp. 160-169, 1989.

[13] P. S. Sundararaghavan and M. U. Ahmed, "Minimizing the Sum of Absolute Lateness in Single-Machine and Multimachine Scheduling", *Naval Research Logistics Quaterly*, Vol. 31, pp. 325-333, 1984.

[14] W. Szwarc, "Single -Machine Scheduling to Minimize Absolute Deviation of Completion Times from a Common Due Date", *Naval Research Logistics Quaterly*, Vol. 36, pp. 663-673, 1989.

Chae Y. Lee
Department of Management Science
Korea Advanced Institute
of Science and Technology
373-1 Kusung Dong, Taejon, Korea