RANDOM KEYS FOR JOB SHOP SCHEDULING

James C. Bean
and
Bryan A. Norman
Department of Industrial and Operations Engineering
The University of Michigan
Ann Arbor, MI 48109-2117

Technical Report 93-7

January 1993

Random Keys for Job Shop Scheduling

James C. Bean and Bryan A. Norman
Department of Industrial and Operations Engineering
University of Michigan
Ann Arbor, MI 48109

January 27, 1993

Abstract

This paper considers the job shop scheduling problem to minimize total tardiness given multiple machines, ready times, sequence dependent setups, machine downtime and scarce tools. We develop a genetic algorithm based on random keys representation, elitist reproduction, Bernoulli crossover, and immigration style mutation. Computational results on problems from the auto industry are presented.

1 Introduction

Scheduling problems represent a rich domain in discrete optimization. Due to problem complexity many of these problems cannot be solved using traditional operations research techniques. Several heuristic methods have been applied to these problems with varied results. Because genetic algorithms have been used successfully to solve discrete optimization problems, scheduling problems have been a natural genetic algorithm application. Genetic algorithms have been applied to scheduling problems by Biegal and Davern [1990], Kanet and Sridharan [1991], Syswerda [1991], Falkenauer and Bouffouix [1991], Whitley, Starkweather and Shaner [1991], Dorndorf and Pesch [1992] and Lee and Hermann [1993] among others. Storer, Wu, and Vaccari [1992] present search and problem space strategies that could be applied using a genetic algorithm. Most of these applications use literal permutation encoding strategies. The code in this paper uses a nonliteral encoding. Literal versus nonliteral encoding is an open research question and the authors do not make any comparisons or state any conclusions concerning these two strategies.

Section 2 describes the job shop scheduling problem. The random keys encoding scheme, as applied to job shop scheduling, is described in Section 3. Sections 4 and 5 present the genetic algorithm and computational results for a set of real problems. Section 6 includes a summary and conclusions.

2 The Job Shop Scheduling Problem

In the general job shop problem there are n jobs to be processed through m machines. Different jobs may be processed on different sets of machines and there may be precedence requirements between jobs. The objective is to determine a sequence for placing the jobs on the machines that optimizes a given evaluation measure. A brief description of job shop scheduling terminology is presented below. For a more comprehensive discussion of scheduling problems and terminology see Baker [1974] and French [1982].

A job's *ready time* is the time a job is available for scheduling. If the ready time is zero then the job is available at the beginning of the study horizon. *Setup time* is the time required to prepare a machine to begin processing a particular job. Setup time may be sequence dependent, meaning that the setup time for a job on a machine depends on which job was previously run on the machine. *Machine uptime* represents the time that a machine is available to perform work. If a machine has periods of downtime due to routine maintenance or breakdowns jobs may only be scheduled on a machine during its uptime. A job has *tool constraints* if it requires a special tool or fixture in order to be processed. If different jobs require the same tool there may be a conflict concerning which

job should use a given tool at a particular time.

There are several different measures available for evaluating a job shop schedule. The measure considered in this paper is total tardiness. A job is tardy if it completes after its due date. In this case tardiness equals its completion time minus its due date. If a job completes prior to its due date then it has a tardiness of zero. The algorithms presented can handle many other objectives such as weighted earliness plus tardiness.

Job shop scheduling problems with a tardiness objective are sufficiently difficult that the single machine problem with nonzero ready times is NP hard even if the following simplifying assumptions are made: setup times are not sequence dependent, no machine downtime, and no special tool requirements. In this paper we consider the more difficult problem of minimizing total tardiness when there are multiple machines, nonzero ready times, sequence dependent setup times, machine downtime, and tool constraints.

3 Genetic Algorithm Encoding Using Random Keys

The random keys representation (Bean [1992]) encodes a solution with *random* numbers. These values are used as sort *keys* to decode the solution. Random keys eliminate the offspring feasibility problem by avoiding the issue. Chromosomal encodings are constructed that represent solutions in a soft manner. These encodings are interpreted in the objective evaluation routine in a way that avoids the feasibility problem. There are other encodings that use random variates (e.g. Bäck, Hoffmeister, and Schwefel [1991]) but not in the manner of random keys. Bagchi et al [1991] explore different scheduling problem encodings but none that are similar to random keys. The strength of random keys is its robustness. The approach works for many other problems. Bean [1992] describes the use for the traveling salesman problem, scheduling, vehicle routing, resource allocation, and quadratic assignment problems. Bean [1992] shows excellent computational results for identical machine scheduling and resource allocation and reasonable computational results for the quadratic assignment problem.

An example of random keys is now provided for the context of job shop scheduling. For the single machine sequencing problem, generate a uniform $(0,1)$ random deviate for each job. When such a sequence of realizations is passed to the objective evaluation routine, sort them and sequence the jobs in ascending order of the sort. For a five job problem, the chromosome

$$(.46, .91, .33, .75, .51)$$

would be sorted to the sequence

$$3 \rightarrow 1 \rightarrow 5 \rightarrow 4 \rightarrow 2.$$

If our objective is to minimize total tardiness, this sequence can then be evaluated by calculating and summing tardiness. Note that many random key vectors would sort to the same sequence.

Crossovers are executed on the chromosomes, the random keys, not on the sequences. Consider two individuals:

$$(.46, .91, .33, .75, .51) \equiv 3 \rightarrow 1 \rightarrow 5 \rightarrow 4 \rightarrow 2$$

and

$$(.84, .32, .64, .04, .48) \equiv 4 \rightarrow 2 \rightarrow 5 \rightarrow 3 \rightarrow 1.$$

Using a traditional one-point crossover, assume that the crossover point is after the second gene. Then the two offspring are:

$$(.84, .32, .33, .75, .51) \equiv 2 \rightarrow 3 \rightarrow 5 \rightarrow 4 \rightarrow 1$$

and

$$(.46, .91, .64, .04, .48) \equiv 4 \rightarrow 1 \rightarrow 5 \rightarrow 3 \rightarrow 2.$$

Since any ordered set of numbers can be interpreted as a sequence, all offspring are feasible solutions.

Through the dynamic of the genetic algorithm, jobs that should be early in the sequence evolve low numbers and jobs late in the sequence evolve large numbers. The random keys simply serve as tags which the crossover operator uses to rearrange jobs.

This approach can be extended to multiple machine problems. Consider the m identical machine n job problem to minimize total tardiness. For each job generate an integer randomly in $\{1, \ldots, m\}$ and add a uniform $(0, 1)$ deviate. Now the integer part of any random key is interpreted as the machine assignment and the fractional parts sorted to sequence on each machine. A single sort gives the jobs assigned to machine 1 in processing order, followed by the jobs on machine 2, etc. Assuming that jobs are processed at their earliest possible time, a schedule can be constructed and evaluated for total tardiness.

We have successfully modified this approach for the complications of nonidentical machines, nonregular measures, nonzero ready times, sequence dependent setups and tool constraints. One modification is in the function evaluation routine. In the schedule construction phase, after the random keys are sorted, a forward pass enforces ready times and inserts sequence dependent setups. Ready times are also used to bias the random keys. Jobs with early ready times have a higher probability of being placed early in the sequence. Tool constraints are captured by updating a tool availability indicator. If a job requires a given tool then it may not start until that tool is available. Then the tool is marked unavailable for the duration of the job's processing.

Successfully solving problems that contain these additional complications is important because they occur in real job shops. In general, not all jobs are available at the beginning of the study horizon but arrive throughout the study horizon. Limited numbers of a given tool may exist and therefore jobs requiring the tool may not be scheduled at the same time. Sequence dependent setups are a natural result of the different tool requirements of the different jobs. These complications make the problem more difficult because optimal schedules often contain unforced idle time. In addition, good solutions may require sequencing that sacrifices local evaluations to obtain a better global evaluation.

4 Genetic Algorithm Operators

There are several types of genetic operators that may be used in the reproduction stage of a genetic algorithm. For details see Goldberg [1989]. For permutation ordering problems several specialized operators have been developed to insure the feasibility of generated solutions. Some of these include PMX Crossover (Goldberg [1985]), the subsequence-swap operator (Grefenstette et al [1985]), the subsequence-chunk operator (Grefenstette [1987]), other subsequence operators (Cleveland and Smith [1989]), edge recombination (Whitley et al [1989]), and forcing (Nakano [1991]). Because the random keys encoding preserves feasibility there is no need to generate specialized operators. The code described here uses clonal propagation, Bernoulli crossover, and immigration. The random keys representation is not limited to these operators and other operators remain to be investigated. However, these operators have performed well in tests completed at this time.

Clonal propagation (Goldberg [1989]), also called an elitist strategy, is accomplished by cloning the best individuals from one generation to the next. This method has the advantage of producing a best solution that is monotonically improving but has the potential for premature convergence. This is overcome by introducing high mutation rates.

Bernoulli crossover (called parameterized uniform crossover in Spears and De Jong [1991]) is used instead of the traditional single-point or multiple-point crossover. Two parents are chosen from the current population. For each gene a biased coin is tossed to determine which parent contributes its allele to each offspring. Consider the example given in Table 1.

Immigration is used instead of single gene mutation to replace genetic material that the current population may have lost. The immigration operator creates a new randomly generated individual to enter the next generation. This individual may then be selected as a parent for the crossover operation in the succeeding generation. This introduces a significant mutation effect which offsets the premature convergence effect of clonal propagation.

Table 1: Random crossover example

Coin Toss	H	H	T	H	T
Parent 1	*.46*	*.91*	.33	*.75*	.51
Parent 2	.84	.32	*.64*	.04	*.48*
Offspring	*.46*	*.91*	*.64*	*.75*	*.48*

5 Computational Results

To determine the effectiveness of the genetic algorithm eight problems from a large automaker were tested. The data sets contain approximately 350 jobs that are to be performed on two machines. The jobs have ready times and due dates that range throughout the study horizon. There is only one copy of each tool and multiple jobs require the same tool. Thus, there are tooling conflicts. In addition, the problems contain sequence dependent setups, periods of machine downtime, and periods of tool unavailability. The scheduling objective is to minimize total tardiness.

Due to the large problem size it was not possible to compute optimal solutions for these problems. Therefore, the genetic algorithm's solutions are compared to total tardiness lower bounds for each of the problems. The lower bound for each problem is the inherent tardiness for the problem. A job's inherent tardiness equals the maximum of zero and its release time plus its processing time minus its due time. Each of the problems was run ten times using a different random seed for each run. The code was implemented on a MasPar MP-1 massively parallel computer with 1024 processors.

Test results for the genetic algorithm are presented in Tables 2 and 3. Table 2 contains results regarding the search for the best possible schedule. Table 3 contains results for obtaining a solution within 5% of the lower bound. These Tables indicate the number of generations and cpu time required for each of the problems. Deviation from the total tardiness lower bounds is also presented. Because the objective is to find the best scheduling sequence, only the best solution obtained by the genetic algorithm is compared to the lower bound, not the average value for the entire population. As a benchmark for the genetic algorithm, random search using the random keys encoding was tested. The clonal propogation factor was set to 1, to retain the single best solution, and the rest of each generation was created using the immigration operator. The algorithm was run for 1500 generations. This represents about 1400 seconds of cpu time which is approximately 1.5 times as much cpu time as the genetic algorithm required. Table 4 contains results for the best solution found, number of generations, and cpu time required.

Table 2: Genetic Algorithm Results - Best Solution

	Number of Generations			CPU Time†			Percent Above Lower Bound		
Problem	Min	Median	Max	Min	Median	Max	Min	Median	Max
1	185	189	207	468	479	524	0.9	0.9	1.1
2	224	242	255	567	613	645	2.5	2.8	3.1
3	171	190	202	451	484	512	1.0	1.0	1.3
4	189	199	230	479	504	583	0.9	0.9	1.2
5	184	191	215	466	484	545	0.3	0.3	0.6
6	185	189	207	468	479	524	0.9	0.9	1.1
7	171	192	203	433	485	515	0.8	1.0	1.3
8	185	189	207	468	479	524	0.6	0.6	1.0

†seconds on MasPar MP-1

Table 3: Genetic Algorithm Results - Within 5% of Lower Bound

	Number of Generations			CPU Time†		
Problem	Min	Median	Max	Min	Median	Max
1	129	139	156	326	351	395
2	173	183	198	437	461	501
3	128	140	154	323	353	364
4	127	144	152	321	364	384
5	129	138	153	326	348	387
6	129	137	156	326	345	395
7	115	141	149	291	357	377
8	126	136	153	318	343	387

†seconds on MasPar MP-1

Table 4: Random Search Results

	Number of Generations			CPU Time†			Percent Above Lower Bound		
Problem	Min	Median	Max	Min	Median	Max	Min	Median	Max
1	734	949	1484	687	888	1389	92.2	99.4	105.3
2	446	1034	1313	417	967	1229	101.1	108.8	114.7
3	734	949	1484	687	888	1389	92.0	100.9	106.3
4	226	835	1152	212	782	1078	99.8	106.7	110.5
5	431	873	1152	403	817	1078	91.2	96.6	104.2
6	734	949	1484	687	888	1389	92.2	99.4	105.3
7	607	1301	1437	568	1218	1345	88.9	93.8	103.3
8	734	949	1484	687	888	1389	91.6	99.0	104.9

†seconds on MasPar MP-1

6 Summary and Conclusions

The genetic algorithm performed well on all eight data sets. The median (over the ten seeds) best solution found was within 1% of the lower bound for seven of the data sets and was within 3% for the eighth. The genetic algorithm produced good results across the different random seed values. The maximum deviation from the lower bound was less than 4% for all eight data sets.

The data in Tables 2 and 3 support the conclusion that the genetic algorithm requires increasing amounts of time to obtain solutions that are very close to optimal. Approximately 35-40% more computation time is required to obtain a solution that is 2-4% above the lower bound as compared to 5% above the lower bound. The results from Table 4 indicate that random search is not an efficient way to solve this problem. The best random search solution found in all the testing was 89% above the lower bound. The median values of the solutions found were almost double the values found by the genetic algorithm. Figure 1 provides a comparison between the genetic algorithm and the random search for a single run of problem eight. The genetic algorithm converges to a solution that is very close to the lower bound much faster than the random search. This same behavior was exhibited by all the random seeds for each of the eight problems. Due to the large size of the solution space it is highly improbable that a good solution would ever be selected randomly. The use of the crossover operator to share good partial sequences was vital to the success of the genetic algorithm.

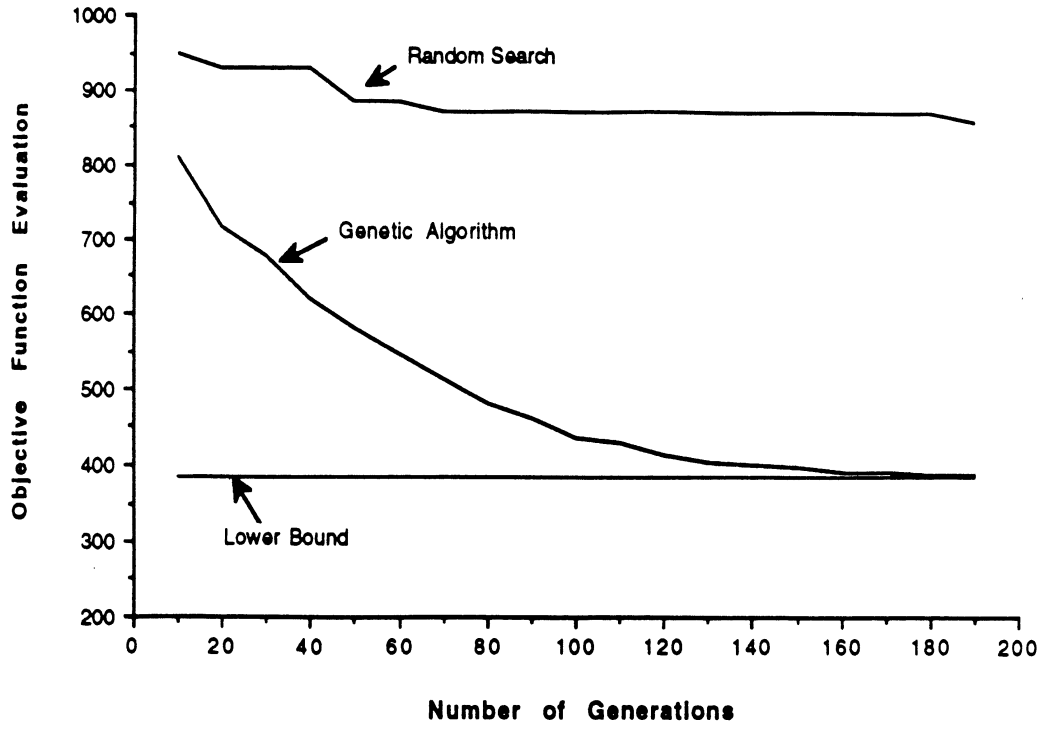

Figure 1: Genetic Algorithm and Random Search Comparison.

7 Acknowledgment

We would like to thank Darrell Whitley for many helpful comments. This research was supported in part by National Science Foundation grants DDM-9018515 and DDM-9202849 to the University of Michigan and by the National Science Foundation Graduate Fellowship Program.

REFERENCES

Bäck, T., F. Hoffmeister, and H. Schwefel [1991], "A Survey of Evolution Strategies," **Proceedings of the Fourth International Conference on Genetic Algorithms**, 2-9.

Bagchi, S., S. Uckun, Y. Miyabe, and K. Kawamura [1991], "Exploring Problem-Specific Recombination Operators for Job Shop Scheduling," **Proceedings of the Fourth International Conference on Genetic Algorithms**, 10-17.

Baker, K. [1974], **Introduction to Sequencing and Scheduling**, Wiley.

Bean, J. C. [1992], "Genetics and Random Keys for Sequencing and Optimization," Technical Report 92-43, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, Michigan 48109. To appear in **ORSA Journal on Computing**.

Biegal, J. and J. Davern [1990], "Genetic Algorithms and Job Shop Scheduling," **Computers and Industrial Engineering**, Vol. 19, 81-91.

Cleveland, G.A. and S. F. Smith [1989], "Using Genetic Algorithms to Schedule Flow Shop Releases," **Proceedings of the Third International Conference on Genetic Algorithms**, 160-169.

Dorndorf, U. and E. Pesch [1992], "Evolution Based Learning in a Job Shop Environment," Working Paper. INFORM - Institut für Operations Research und Management GmbH, Pascalstraße 23, D-5100 Aachen, F.R.G.

Falkenauer E. and S. Bouffouix [1991], "A Genetic Algorithm for Job Shop," **Proceedings of the 1991 IEEE International Conference on Robotics and Automation**, 824-829.

French, S. [1982], **Sequencing and Scheduling**, Halstead Press.

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

Goldberg, D. E. and R. Lingle Jr. [1985], "Alleles, Loci, and the Traveling Salesman Problem," **Proceedings of the First International Conference on Genetic Algorithms**.

Grefenstette, J. J., R. Gopal, B. Rosmaita, and D. Van Gucht [1985], "Genetic Algorithms for the Traveling Salesman Problem," **Proceedings of the First International Conference on Genetic Algorithms**.

Grefenstette, J. J. [1987], "Incorporating Problem Specific Knowledge into Genetic Algorithms," **Genetic Algorithms and Simulated Annealing**, (ed. L. Davis) Morgan Kaufman Publishers.

Kanet, J. J. and V. Sridharan [1991], "PROGENITOR: A genetic algorithm for production scheduling," **Wirtschafts Informatik**, Vol. 33, 332-336.

Lee, C. and J. Herrman [1993], "Decision Support Systems for Dynamic Job Shop Scheduling," **Proceedings of the 1993 NSF Design and Manufacturing Systems Conference,** Charlotte, NC, Vol. 2, 1119-1123.

Nakano, R. [1991], "Conventional Genetic Algorithm for Job Shop Problems," **Proceedings of the Fourth International Conference on Genetic Algorithms**, 474-479.

Spears, W. M. and K. A. De Jong [1991], "On the Virtues of Parameterized Uniform Crossover," **Proceedings of the Fourth International Conference on Genetic Algorithms**, 230-236.

Storer, R. H., S. D. Wu, and R. Vaccari [1992], "New Search Spaces for Sequencing Problems With Application to Job Shop Scheduling," **Management Science**, Vol. 38, No. 10, 1495-1509.

Syswerda, G. [1991], "Schedule Optimization Using Genetic Algorithms," in **Handbook of Genetic Algorithms**, (ed. L. Davis), Van Nostrand, 332-349.

Whitley, D., T. Starkweather, and D. Fuquay [1989], "Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination Operator," **Proceedings of the Third International Conference on Genetic Algorithms**, 133-140.

Whitley, D., T. Starkweather, and D Shaner [1991], "The Traveling Salesman and Sequence Scheduling: Quality Solutions Using Genetic Edge Recombination," in **Handbook of Genetic Algorithms**, (ed. L. Davis), Van Nostrand, 350-372.