A Random Keys Genetic Algorithm for Job Shop Scheduling

Bryan A. Norman
Department of Industrial Engineering
University of Pittsburgh Pittsburgh, PA 15261

James C. Bean
Department of Industrial & Operations Engineering
University of Michigan Ann Arbor, MI 48109

Technical Report 96-10

November 1996

Department of Industrial and Operations Engineering

The University of Michigan
College of Engineering
Ann Arbor, Michigan 48109-2117

A Random Keys Genetic Algorithm for Job Shop Scheduling

Bryan A. Norman
Department of Industrial Engineering
University of Pittsburgh Pittsburgh, PA 15261

James C. Bean
Department of Industrial & Operations Engineering
University of Michigan Ann Arbor, MI 48109

Technical Report 96-10

November 1996

A RANDOM KEYS GENETIC ALGORITHM FOR JOB SHOP SCHEDULING

Abstract

Many manufacturing settings exhibit a job shop structure where products must be routed through a series of different machines. We propose a genetic algorithm that uses the random keys encoding to schedule job shops. Computational results demonstrate the effectiveness of this approach relative to other methods that have been applied to the problem. The proposed methodology can also be extended to more complex scheduling domains.

Key Words: *automated scheduling, genetic algorithms, job shop*

Bryan A. Norman
Department of Industrial Engineering
University of Pittsburgh
1033 Benedum Hall, Pittsburgh, PA 15261
Phone: (412) 624-9841 Fax: (412) 624-9831
banorman@engrng.pitt.edu

James C. Bean
Department of Industrial & Operations Engineering
University of Michigan
Ann Arbor, MI 48109

August 1996

1 Introduction

Consider the $n/m/J/C_{Max}$ job shop problem. There are n jobs to be processed on m machines. Each job contains m operations and must visit each of the m machines. The operations must be completed in a specific order and each operation can only be completed on one machine. Therefore, there is a strict precedence ordering of the machine sequence for each job. However, the machine sequences for different jobs may vary. The objective is to minimize the completion time of the job that completes last. This problem assumes that all jobs are ready at time zero, there is no routing flexibility - each job has a specific operation sequence or routing through the machines, operations cannot be preempted, and all of the processing times are deterministic. This problem is described in detail in several scheduling texts see [1], [2], and [3].

This problem is of sufficient difficulty that it was not possible to prove the optimality of solutions for even small problem instances until recently. Carlier and Pinson [4] developed an efficient branch-and-bound method that utilized both improved bounding techniques and a new heuristic rule for branch selection. Using this algorithm, they were the first to prove the optimality of a solution for the 10 job and 10 machine problem proposed by [5]. The method found in [4] was enhanced by [6] who introduced improved lower bounds and a better heuristic branch selection rule which significantly reduced the run time for the branch-and-bound algorithm. These efforts have been extended by [7] and [8]. However, even these latest methods encounter long run times for large problems.

Recently, new heuristic techniques have also been introduced for this problem. The best constructive heuristics are the shifting bottleneck heuristic of [9] and the extension due to [10]. This method has performed well on many problems and has been utilized in conjunction with other heuristic methods ([11]).

Because of the difficulty of using branch-and-bound and developing effective special purpose heuristics for job shop scheduling problems, general heuristic search techniques

have been applied to these problems in recent years. Three of the most promising heuristic methods are genetic algorithms (GAs), tabu search, and simulated annealing. In this research we focus on genetic algorithm methodologies and provide a detailed discussion of this topic in the remaining sections. Those interested in the other methods should consult [12] and [13] for a good overview of the mechanics of tabu search. Applications of tabu search to job shop scheduling problems are discussed in [14], [15], [16], and [17]. Both [18] and [19] demonstrate how to use simulated annealing to model optimization problems. Applications to the job shop problem are discussed in [20] and [21].

2 Previous GA Approaches to Scheduling Problems

GAs have been applied to scheduling problems by several authors. The primary strategy that has been used is a literal permutation ordering encoding ([22], [23], [24], [25], [26], [27], and [28]). In the literal encoding strategy the chromosome has a one-to-one correlation with the job sequence. However, this encoding leads to infeasible job sequences if the single-point, multi-point or parameterized uniform crossover operator is used.

To overcome this difficulty several specialized operators have been developed to insure the feasibility of generated solutions. Some of these include PMX crossover ([29]), cycle crossover ([30]), edge recombination ([31]), order-based and position-based crossover ([32]), and linear order crossover (LOX) [25]. Unfortunately, when these operators are applied in the job shop scheduling context to ensure that the chromosomes remain feasible much of the information concerning the job sequence may be lost.

Kobayashi, Ono, and Yamamura [26] present some interesting results using a subsequence exchange crossover operator. They use a literal encoding to represent the sequence on each machine. They permit crossovers only if both parents have a subsequence containing the same operations for one or more of the machines. They proceed further to change

the resulting sequences into active ones using the active schedule generation method of Giffler and Thompson [33].

Croce, Tadei, and Volta [28], describe a genetic algorithm that uses a direct encoding to determine a preference list for the sequence of operations for each machine. To avoid infeasibility problems, crossovers can only occur between different machines. A limitation of their schedule construction mechanism is that it only constructs nondelay schedules. In job shop problems the best nondelay schedule may have an objective function value that is much larger than the optimum value (see [2] for an example.) To alleviate this difficulty they introduce a "lookahead" mechanism to insert idle time into schedules.

Both [34] and [35] use a binary GA where the chromosome contains one bit for each possible job pair. The precedence for the jobs is determined by the presence of a 0 or a 1 for the allele. This representation still results in chromosomes that translate to infeasible sequences following the use of single-point, multi-point, or uniform crossover. Nakano in [35] uses the mechanisms of "harmonization" and "forcing" to ensure that all chromosomes are translated to feasible sequences. Fox and McMahon [34] develop two new operators that they call "union" and "intersection" to share allele information between chromosomes. However, these operators require repairs in order to insure that chromosomes map to feasible sequences. The adverse effects of these repair operators on the chromosomes limit the effectiveness of these operators in the job shop scheduling context.

Fang, Ross, and Corne [36] apply an order based encoding to the job shop scheduling problem. Given a chromosome of length L, the possible allele values for each gene, g, are the integers in the range $(1, L+1-g)$. This encoding has the advantage that crossover can be applied to any two chromosomes and the resulting offspring will be feasible. However, this method may perform poorly because the interpretation of genes located late in the chromosome is highly dependent on the allele values of the genes located early in the chromosome ([37]). A change in a single allele value early in the chromosome can have a

dramatic effect on the resulting schedule.

Dorndorf and Pesch [11] present two interesting GA applications. The first, referred to as P-GA, uses a GA as a meta heuristic. Active schedules are constructed using the algorithm of [33]. At each step of the procedure a conflict set of operations is created. This set contains all the schedulable operations (predecessors have already been scheduled) that are competing for the machine that is required by the schedulable operation that would complete first if there was no resource constraint. Instead of selecting jobs from the conflict set randomly, or using a single priority rule, the selection of jobs from the conflict set is determined by the allele values. This method is very similar to the heuristic space method described in [38].

The second application of [11] is based upon using the shifting bottleneck procedure of [9]. The GA is used to determine the order in which the single machine problems will be solved in the enumeration tree of the procedure.

The genetic local search (GLS) methods of [21] represent an alternative GA based procedure for job shop scheduling. They utilize the disjunctive graph representation of the problem that was introduced by [46]. They define two neighborhood structures based on performing arc exchanges in this graph. A GA is used to explore exchanging different arcs and thus moving from one neighbor to another.

One of the more innovative applications of GAs to scheduling problems is the Problem Space method of [38]. This method begins from the premise that most scheduling problems have base heuristics that are fast and obtain good solutions, though not necessarily optimal ones. Such heuristics may include SPT, EDD, modified due date and more complicated variations. They note that, for difficult scheduling problems, such heuristics do not always find outstanding solutions. They go on to conjecture that if the data were perturbed, the heuristic may find a solution that is outstanding for the original problem. The problem space approach searches this set of perturbations using different search

mechanisms including GAs.

3 The Random Keys GA For Job Shop Scheduling

We now present a detailed discussion of the random keys GA (RKGA) methodology for the job shop scheduling problem. The discussion has three components. The first, is the random keys encoding and how it pertains to ordering problems such as those found in the scheduling domain. Second, we introduce the concept of a delay factor and how it enhances the effectiveness of the RKGA. The third aspect of our solution approach concerns our choice of operators and stopping criteria for the GA.

3.1 GA Encoding Using Random Keys

The random keys representation ([39]) encodes a solution with *random* numbers. These values are used as sort *keys* to decode the solution. These encodings are interpreted in the fitness evaluation routine in a way that avoids the feasibility problem. A similar idea was presented by [40]. They embedded their encoding in a genetic algorithm that dynamically adjusts its internal representation of the search space according to the problem being solved.

We now provide an example of the random keys encoding for the context of job shop scheduling. As a foundation, consider the single machine sequencing problem. Begin by generating a uniform $(0, 1)$ random variate for each job. When such a sequence of realizations is passed to the fitness evaluation routine, sort them and sequence the jobs in ascending order of the sort. For a five job single machine problem, consider the chromosome

$$(.29, .96, .17, .84, .48).$$

Because the smallest random key, .17, corresponds to gene 3, job 3 is the first job in the

sequence. The second job in the sequence is job 1 because .29, the second smallest random key, corresponds to gene 1. Continuing in this fashion, the sequence is

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

The random keys encoding has the advantage, over a literal encoding, that all crossovers produce feasible sequences. Crossovers are executed on the chromosomes, the random keys, not on the sequences. Consider these two chromosomes.

$$\text{Parent 1} = (.29,.96,.17,.84,.48) \equiv 3 \rightarrow 1 \rightarrow 5 \rightarrow 4 \rightarrow 2$$
$$\text{Parent 2} = (.73,.14,.43,.54,.32) \equiv 2 \rightarrow 5 \rightarrow 3 \rightarrow 4 \rightarrow 1$$

Using a traditional one-point crossover, assume that the crossover point is after the second gene. This results in the following two offspring.

$$\text{Offspring 1} = (.29,.96,.43,.54,.32) \equiv 1 \rightarrow 5 \rightarrow 3 \rightarrow 4 \rightarrow 2$$
$$\text{Offspring 2} = (.73,.14,.17,.84,.48) \equiv 2 \rightarrow 3 \rightarrow 5 \rightarrow 1 \rightarrow 4$$

Since any ordered set of random keys can be interpreted as a job sequence, all offspring are feasible solutions. The random keys simply serve as tags that the crossover operator uses to rearrange jobs. The random keys encoding works through the dynamic of the GA. Jobs that should be early in the sequence evolve low random key values and jobs that should be late in the sequence evolve large random key values. The random keys encoding permits the offspring resulting from the crossover operator to retain genetic information from both parents. Consider the previous example, offspring 1 retains the relative ordering of jobs 1 and 2 found in parent 1 and the relative ordering of jobs 3, 4, and 5 from parent 2.

Implementing a mutation operator for a real coded GA is difficult. For a binary encoding, mutation consists of flipping one bit of the gene with a given probability. It is

not obvious how to translate bit flipping in a binary encoding to a real valued encoding. Janikow and Michalewicz [42] discuss some possible mechanisms for implementing single gene mutation for real valued encodings. One of the methods they discuss involves perturbing the real value of an allele. We use a similar mechanism where each random key and delay factor can have its value increased or decreased by a fraction, β, of its current value.

3.2 Delay Factor Encoding

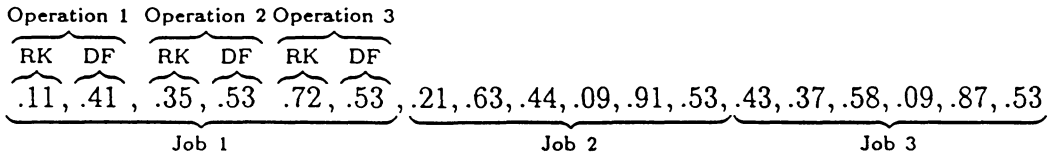
The random keys encoding was modified to enhance performance on job shop scheduling problems. The length of the gene is doubled so that there are two genes associated with each job. The first gene contains the $(0, 1)$ random key value that is used to determine the order in the sequence. Note that at each step of the algorithm only the random keys for the schedulable jobs are sorted to determine which job is placed next in the sequence. The second gene contains a *delay factor*. The delay factor is a $(0, 1)$ variate that indicates flexibility in moving this job after placement. Due to the precedence constraints found in job shop problems, sorting the random keys may lead to sequences with excessive inserted idle time. By permitting jobs to move earlier in the sequence, in a manner similar to a global left shift [1], it is possible to create schedules with significantly better objective function values. A concept similar to the delay factor was utilized in [38].

Job	Operation 1		Operation 2		Operation 3	
Number	Machine	Time	Machine	Time	Machine	Time
1	1	10	2	5	3	2
2	2	4	3	3	1	5
3	2	9	1	6	3	3

Table 1: Delay Factor Example Problem Data.

Consider the example problem data given in Table 1. A chromosome for this problem contains 18 genes. The value of 18 reflects the fact that there are 9 operations (3 jobs x 3

machines) and for each operation there is a random key (RK) and a delay factor (DF). A sample chromosome is given below with details about the genes associated with the first job.



Given this chromosome it is possible to construct a schedule. Initially, the first operation of each job is schedulable. Sorting the random keys of these operations results in operation one of job one being schedule first. Now job one's first operation is removed from the schedulable list and job one's second operation is added to the schedulable list. Continuing in this fashion and ignoring the delay factors the placement sequence is

$$o_{1,1} - o_{2,1} - o_{1,2} - o_{3,1} - o_{2,2} - o_{3,2} - o_{1,3} - o_{3,3} - o_{2,3}$$

where $o_{i,j}$ represents operation j of job i. The resulting schedule for this placement sequence is shown in Figure 1 and has a makespan of 35.

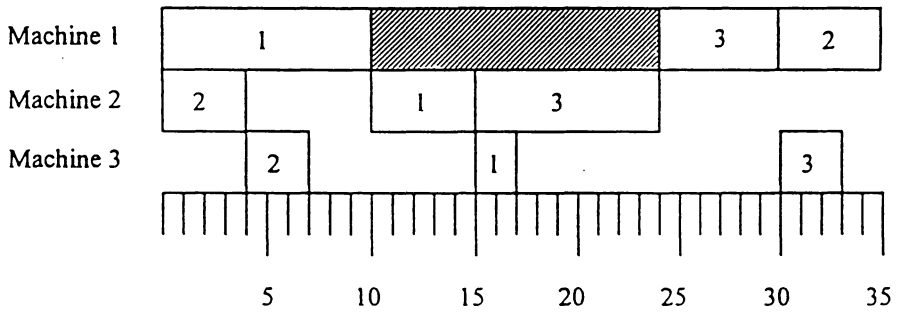


Figure 1: Schedule without Move Search.

This schedule contains undesirable idle time on the first machine between times 10 and 24, indicated by cross hatching, where $o_{2,3}$ could have been scheduled. A procedure,

titled *move search*, uses the delay factor and finds these situations and moves operations up in the schedule to utilize the idle time. The *move search* procedure is now described in more detail.

A sort of the random keys finds the next *candidate operation* to be scheduled. If performing the candidate operation on its assigned machine results in idle time on that machine, as in the case of $o_{3,2}$ in Figure 1, then a *move search* occurs. A *move search* analyzes the remaining schedulable operations that utilize the same machine as the candidate operation. The order in which *move search* reviews the operation list is determined by the random key values of the operations. If one of these operations can be placed on the candidate operation's machine and complete prior to the start time of the candidate operation (S_{CO}), then it is placed on the machine prior to the candidate operation. This is an example of performing a global left shift. Consider the schedule shown in Figure 1, $o_{2,3}$ can be started on on machine 1 at time 10 and complete prior to the start time of $o_{3,2}$. Therefore, *move search* would shift $o_{2,3}$ to start at time 10 on machine 1. The result is the schedule in Figure 2 that has an improved objective of 33 due to the reduced idle time on machine 1.

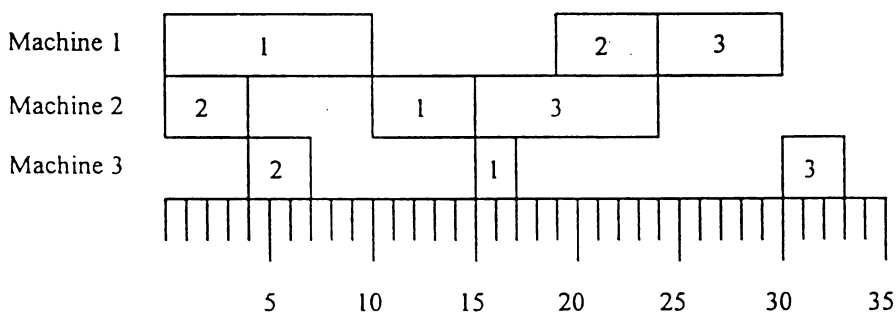


Figure 2: Schedule with Move Search.

Sometimes it may be a good idea to permit a move even if the moving operation would not complete prior to the candidate operations start time. Refer back to the 3 job and

3 machine example. There are 6 idle time units on machine 2 prior to $o_{1,2}$ but placing $o_{3,1}$ ahead of $o_{1,2}$ will delay $o_{1,2}$. However, this is still a good idea because $o_{1,2}$ is only delayed three time units in exchange for eliminating 6 units of idle time on machine 2. By permitting the move the makespan is reduced from 33 to 23. The validity of this move would be determined by using the delay factor for $o_{3,1}$, $DF_{o_{3,1}}$. If the earliest start time of operation $o_{3,1}$ plus its delay factor times its processing time is less than the start time for the candidate operation, or

$$S_{o_{3,1}} + DF_{o_{3,1}} * p_{o_{3,1}} < S_{CO}$$

then $o_{3,1}$ moves ahead of $o_{1,2}$.

The *move search* procedure is shown in Table 2. Recall that the candidate operation has already been selected and scheduling the candidate operation next must create idle time on its respective machine in order for the *move search* procedure to be invoked.

1. Let K represent the set of schedulable operations that require the same machine as the candidate operation. Order K based on the random key values of the operations.
2. If $K = \emptyset$, stop. Else set $a = 1$.
3. Let $o_{i,j}$ be the operation associated with element a of k, k_a.
4. If $S_{o_{i,j}} + DF_{o_{i,j}} * p_{o_{i,j}} < S_{CO}$ then $o_{i,j}$ becomes the candidate operation. Update S_{CO}.
5. If $a < |K|$, let $a = a + 1$ and return to step 3. Else stop.

Table 2: *Move Search* Procedure

There exists a trade-off between delaying the start of the candidate job and reducing the idle time of the machines. While it is good to reduce idle time it is important not to delay any operations that are on the longest path of the schedule that determines the makespan. Due to the complexity of job shop problems it is not possible to determine a general rule for determining the appropriate value for the delay factor for each operation. For this reason the delay factor has been added to the chromosome so that the GA can

dynamically modify it for each operation.

The resulting algorithm lets the random keys be the principal agent determining the operation sequence. Move searches are utilized to improve the search procedure. Deriving schedules using the combination of random keys and move searches results in the RKGA exploring a subset of the set of semi-active schedules that contains optimal schedules (for a detailed proof see [41]). Thus, there exists a region within the chromosome space that maps to the optimal solution to the problem. Note that the move search procedure does not reduce the search space to the the set of active schedules. Any operation can move in front of at most one previously scheduled operation. While it is desirable to reduce the search space to set of active schedules it is too computationally expensive to insure that all schedules constructed are provably active schedules.

3.3 GA Operators

We use standard GA operators in our implementation. Elitist reproduction is used to guarantee that our best solution is monotonically improving. Based on preliminary test results, we utilize parameterized uniform crossover (with a crossover probability of 0.7) in place of the traditional single-point or multiple-point crossover. After performing the crossover operation only the better of the two offspring is retained. Tournament selection is used in conjunction with the crossover operation to fill the next generation.

A parallel study on multiple-choice genetic algorithms showed that, without the tournament selection, parametrized uniform crossover returns an offspring with the same expected fitness as the average of the parents. If both offspring are added to the next generation, this result still holds. By retaining only the better of the two offspring in tournament selection the fitness of the offspring is substantially enhanced. We conjecture that the tournament plays a similar role in the random keys setting.

4 Computational Results

The RKGA for the job shop scheduling problem is compared with other GA approaches, simulated annealing, tabu search, and the shifting bottleneck heuristic. The test set consists of two problems from [5], a 10 job and 10 machine problem (M10x10) and a 20 job and 5 machine problem (M20x5), and 40 problems from [45], LA01 to LA40, with different sizes ranging from 10 to 30 jobs and 5 to 15 machines. These problems comprise the standard test set of job shop problems. For the test problems the following parameter settings were used for the RKGA: a variable population size equal to 300 plus 2 times the total number of operations in the problem, a maximum of 250 generations, 5 elite solutions, 6% of the solutions mutated with 50% of the genes mutated, a gene modification value, β, of 0.5, and a tournament size of 2. The degree of each mutation is high but only used in 6% of the population. This results in a mutation effect that is similar to introducing a few random members into the population. Empirical results indicate that if the GA does not improve the current best solution after a fixed number of successive generations then additional searching yields little improvement. Therefore, the stopping criteria for the GA was either the maximum number of generations or 75 generations without any improvement in the objective function.

Six papers from the literature that represent the best GA results that have been presented at this time include: [36], [26], [28], [11], [21], and [47]. These methods were described in Section 2. Comparisons with each of these are described in more detail below.

Fang, Ross, and Corne [36] present results on two problems by [5] - M10x10 and M20x5. Their average solutions over ten runs for these two problems are 977 and 1215 respectively and the best solutions for the 10 runs are 949 and 1189 respectively. The RKGA was run using 10 different random number seeds and the average over all 10 and the best solution over the first 5 seeds is noted. The best of only the first 5 seeds was used in order to provide a more fair comparison with other algorithms since some of these other

algorithms were run with only 5 seeds. From the data in the first two rows of Table 3 it is clear that the RKGA finds better solutions on average for these two problems. A comparison of the best solution found yields similar results (even though only 5 of the RKGA seeds were used) as the RKGA attained a value of 937 for the 10x10 problem and 1178 for the 20x5 problem. Fang, Ross, and Corne [36] do not present computation times for their algorithm but they do indicate how many function evaluations the algorithm performed. The parameter settings for the RKGA resulted in the RKGA using fewer function evaluations. It is difficult to make strong conclusions based on only two problems, however the RKGA seems to find solutions that are better than those found by the GA of Fang, Ross, and Corne and with fewer function evaluations.

Kobayashi, Ono, and Yamamura [26] also only present results for M10x10 and M20x5. They perform 100 trials on M10x10 and find a best solution of 930 and an average solution of 939. These results are similar to those seen for the RKGA in Table 3. However, their results indicate that they examine 1.6 x 10^6 solutions in each trial which is significantly more than the RKGA examines. Additionally, their method must take additional time to convert each solution to a provable active schedule. They do not provide computation times for their approach but these two factors make it appear that it would be significantly higher than that of the RKGA method. It is difficult to compare the methods based on the results of only two problems. It would be interesting to see if they can determine robust parameter settings that would perform well on all of the problems in the test set of [45].

Croce, Tadei, and Volta [28] tested their algorithm on M10x10, M20x5, and 8 of the problems of [45]. For each problem they ran the algorithm using 5 different random number seeds and they present the best and average solution found for these 5 seeds. Their results are given in Table 3. Comparing their results with those of the RKGA it is apparent that the RKGA has equal or better performance on average and for the best solution found over 5 seeds for every problem tested. A paired t test indicated that these

Problem	Size	Optimal or Best Known	RKGA		Croce et al.		Dorndorf & Pesch			Aarts et al.		Storer et al.
			Avg.	Best	Avg.	Best	P-GA	SB-GA(40)	SB-GA(60)	GLS1	GLS2	PSGA
M10x10	10x10	930	945	937	965	946	960	-	-	935	945	952
M20x5	20x5	1165	1176	1165	1199	1178	1249	-	-	1165	1167	-
LA01	10x5	666	666	666	666	666	666	666	-	666	666	666
LA02	10x5	655	661	655	-	666	681	666	-	668	659	-
LA03	10x5	597	599	597	-	666	620	604	-	613	609	-
LA04	10x5	590	592	590	-	-	620	590	-	599	594	-
LA05	10x5	593	593	593	-	-	593	593	-	593	593	-
LA06	15x5	926	926	926	926	926	926	926	-	926	926	-
LA07	15x5	890	890	890	-	-	890	890	-	890	890	-
LA08	15x5	863	863	863	-	-	863	863	-	863	863	-
LA09	15x5	951	951	951	-	-	951	951	-	951	951	-
LA10	15x5	958	958	958	-	-	958	958	-	958	958	-
LA11	20x5	1222	1222	1222	1222	1222	1222	1222	-	1222	1222	-
LA12	20x5	1039	1039	1039	-	-	1039	1039	-	1039	1039	-
LA13	20x5	1150	1150	1150	-	-	1150	1150	-	1150	1150	-
LA14	20x5	1292	1292	1292	-	-	1292	1292	-	1292	1292	-
LA15	20x5	1207	1207	1207	-	-	1237	1207	-	1207	1207	-
LA16	10x10	945	958	945	989	979	1008	961	961	977	977	981
LA17	10x10	784	784	784	-	-	809	787	784	791	791	794
LA18	10x10	848	850	848	-	-	916	848	848	856	858	860
LA19	10x10	842	853	851	-	-	880	863	848	863	859	860
LA20	10x10	902	908	907	-	-	928	911	910	913	916	-
LA21	15x10	1047*	1062	1055	1113	1097	1139	1074	1074	1084	1085	-
LA22	15x10	927	936	933	-	-	998	935	936	954	944	-
LA23	15x10	1032	1032	1032	-	-	1072	1032	1032	1032	1032	-
LA24	15x10	935	977	966	-	-	1014	960	957	970	981	-
LA25	15x10	977	995	987	-	-	1014	1008	1007	1016	1010	-
LA26	20x10	1218	1218	1218	1248	1231	1278	1219	1218	1240	1236	-
LA27	20x10	1236*	1269	1256	-	-	1378	1272	1269	1308	1300	-
LA28	20x10	1216	1241	1241	-	-	1327	1240	1241	1281	1265	-
LA29	20x10	1160*	1188	1179	-	-	1336	1204	1210	1290	1260	-
LA30	20x10	1355	1355	1355	-	-	1411	1355	1355	1402	1386	-
LA31	30x10	1784	1784	1784	1784	1784	-	-	-	1784	1784	-
LA32	30x10	1850	1850	1850	-	-	-	-	-	1850	1850	-
LA33	30x10	1719	1719	1719	-	-	-	-	-	1719	1719	-
LA34	30x10	1721	1721	1721	-	-	-	-	-	1737	1730	-
LA35	30x10	1888	1888	1888	-	-	-	-	-	1894	1890	-
LA36	15x15	1268	1300	1287	1330	1305	1373	1317	1317	1324	1311	1305
LA37	15x15	1397	1432	1418	-	-	1498	1484	1446	1449	1450	1458
LA38	15x15	1184*	1232	1217	-	-	1296	1251	1241	1285	1283	1239
LA39	15x15	1233	1260	1258	-	-	1351	1282	1277	1279	1279	1258
LA40	15x15	1222	1256	1234	-	-	1321	1274	1252	1273	1260	1258
Paired t test α value					.005	.012	$< .001$	.005	.033	$< .001$	$< .001$	.004

* Represents the best known solution, no provably optimal solution exists for this problem.

Table 3: RKGA Job Shop Comparison.

differences were significant at a .005 confidence level for the average solution and a .012 confidence level for the best solution found. However, there is a trade-off since Croce, Tadei, and Volta's algorithm requires 3 to 5 times less cpu time than the RKGA.

Dorndorf and Pesch [11] tested several GA based algorithms using the problems found in [45]. The first, referred to as P-GA, was run using only one random number seed and these results are presented in Table 3. Because they only tested one random number seed it is best to compare the result with the average result over 10 seeds for the RKGA. The data indicate that the RKGA produces results that are equal to or better than the P-GA for every problem tested. A paired t test indicated that the average difference was significant at a confidence level less than .001. The computation times for the two methods are similar.

Dorndorf and Pesch's [11] second approach is based on the shifting bottleneck heuristic of [9]. This method spawned two algorithms, SB-GA(40) and SB-GA(60) (which only differ in their population sizes), which use a GA to determine in what order the single machine scheduling problems should be solved in the shifting bottleneck procedure. Both algorithms were run for two different seeds and the average of these two seeds is presented in Table 3. Note that SB-GA(60) was only run on the larger problems that SB-GA(40) had difficulty solving. A comparison of the RKGA with SB-GA(40) indicates that the two algorithms provide similar solutions for many problems. As the problem size increases, problems LA26 to LA40, the RKGA provides better solutions, on average, for each problem. A paired t test showed that the RKGA provides better average results with a confidence level of .005. The results for the SB-GA(60) method are closer to the RKGA results. Again, the RKGA seems to provide better results for the larger problems but the difference is not as significant as in the case of SB-GA(40). A paired t test for the RKGA and SB-GA(60) indicated that the RKGA finds better results on average with a confidence level of .033. The RKGA requires about 4 times more computation time than SB-GA(40) and about 2 times more computation time than SB-GA(60). Additionally,

the RKGA has the advantage that is readily applicable to additional problem contexts that contain release times, due times, and alternative routings while the SB-GA methods are limited to problems without these complexities because they utilize the shifting bottleneck procedure.

Test results for the genetic local search (GLS) methods of [21] are shown in the columns GLS1 and GLS2 (these reflect the two different neighborhood structures) of Table 3. The values in the table represent the average solution found over 5 runs. The RKGA finds better solutions on average than both methods. Comparisons with both GLS1 and GLS2 show that RKGA finds equal or better solutions in all but 5 instances. A paired t test comparison with both methods indicated that the RKGA finds better solutions on average with a confidence level less than .001 for both GLS1 and GLS2. The computation times for both genetic local searches are slightly, as much as 20%, less than those for the RKGA.

Storer, Wu, and Park's [47] problem space genetic algorithm (PSGA) presents an additional GA-based method. This algorithm was tested on M10x10 and 10 of the problems from [45]. They present the best solution for their algorithm using one large population and for keeping the best solution generated by 20 small populations. Table 3 only contains the results for the 20 small populations strategy because that performed better. They only report the results for one seed for the 20 small populations so comparisons with the average RKGA performance for each problem are most appropriate. The parameters used for the RKGA insured that it performed fewer schedule evaluations than the PSGA. The results show that the RKGA found equal or better solutions on average for 10 of the 11 problems and the one where it found worse solutions the difference was very small. A paired t test indicated that the difference between the two methods was significant at a .004 confidence level.

In addition to the six GA based procedures, simulated annealing and tabu search have been applied to job shop scheduling problems. The best SA results are from [20]. Norman [41] presents a detailed comparison between the RKGA and the their simulated

annealing algorithm. Overall the two methods provided solutions of similar quality but the RKGA solutions were slightly better and required less computation time. Tabu search has been applied to job shop scheduling by [14], [15], [16], and [17]. The best results have been found by the algorithms of [15] and [16]. These algorithms exploit the problem structure in forming the neighborhood for the tabu search. The resulting algorithms find solutions that are on average less than 1% better than those found by the RKGA and require less computation time. However, because they exploit the specific problem structure found in these test problems they are not as readily generalized to more complex problems. The RKGA can be applied to job shop scheduling problems with multiple, nonidentical machines, nonzero ready times, sequence dependent setups, tool constraints, and precedence as seen in [41].

An additional comparison involves the RKGA and the shifting bottleneck methods of [9] and [10]. These represent the best "constructive" methods for determining job shop schedules. The RKGA finds solutions that on average are 1% better than any of the shifting bottleneck methods. But the shifting bottleneck procedures are much faster than the RKGA because they are constructive procedures that are tailored to this specific problem. However, the shifting bottleneck methods cannot be easily adapted to more complex problems containing sequence dependent processing times, tooling constraints, and alternative part routings as the RKGA can.

The final comparison involves the RKGA with delay factors and move search procedure to the RKGA without these elements. Given the GA parameters described earlier in this section, the RKGA with delay factors and move search finds solutions that are on average 3% better than the RKGA operating without any type of move search procedure and 1% better than the RKGA utilizing only the move search but without any delay factors (or essentially setting the delay factor to always equal 1.0). Paired t tests indicated that both of these differences are significant at a confidence level less than .001.

5 Conclusions and Further Research

We present a new genetic algorithm encoding for the job shop scheduling problem. This method performs well compared to other general genetic algorithm approaches in terms of the quality of the solutions found. Moreover, the method performs well relative to GA methods that are tailored to this specific problem structure, such as the method of [21] and [11]. The RKGA is also competitive with other heuristic methods such as the best simulated annealing and tabu search implementations that are designed specifically for this problem class. Additionally, the RKGA finds better solutions than the shifting bottleneck constructive methods although it requires more computation time. The proposed RKGA has the added advantage that it can be readily extended for application to more complicated scheduling environments. These environments may include the following complexities: multiple, nonidentical machines, nonzero ready times, sequence dependent setups, tool constraints, precedence, and alternative routings. This makes the random keys encoding a robust encoding that can be applied effectively to multiple complex scheduling environments.

References

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

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

[3] Morton, T. E. and D. W. Pentico, Heuristic Scheduling Systems, John Wiley and Sons, 1993.

[4] Carlier, J. and E. Pinson, "An Algorithm for Solving the Job-Shop Problem," *Management Science*, Vol. 35, 164-176, 1989.

[5] Muth, J. F. and G. L. Thompson, Industrial Scheduling, Prentice Hall, 1963.

[6] Applegate, D. and W. Cook, "A Computational Study of the Job-Shop Scheduling Problem", *ORSA Journal on Computing,* Vol. 3, 149-156, 1991.

[7] Carlier, J. and E. Pinson, "Adjustment of heads and tails for job-shop problem," *European Journal of Operational Research*, Vol. 35, 164-176, 1994.

[8] Brucker, P., B. Jurisch, and A. Krämer, "The job-shop problem and immediate selection," *Annals of Operations Research,* Vol. 50, 73-114, 1994.

[9] Adams, J., Balas, E., and Zawack D., "The shifting bottleneck procedure for job shop scheduling", *Management Science,* Vol. 34, 391-401, 1988.

[10] Balas, E., J. K. Lenstra, and A. Vazacopoulos, "The One Machine Problem with Delayed Precedence Constraints and its Use in Job Shop Scheduling," *Management Science,*, Vol. 41, 94-109, 1995.

[11] Dorndorf, U. and E. Pesch, "Evolution Based Learning in a Job Shop Environment," *Computers and Operations Research*, Vol. 22, 25-40, 1995.

[12] Glover, F., "Tabu Search - Part I," *ORSA Journal on Computing*, Vol. 1, 190-206, 1989.

[13] Glover, F., "Tabu Search - Part II," *ORSA Journal on Computing*, Vol. 2, 4-32, 1990.

[14] Barnes, J. W. and M. Laguna, "A Tabu Search Experience in Production Scheduling," *Annals of Operations Research*, Vol. 41, 141-156, 1993.

[15] Dell'Amico M. and M. Trubian, "Applying Tabu Search to the Job-Shop Scheduling Problem," *Annals of Operations Research*, Vol. 41, 231-252, 1993.

[16] Nowicki, E. and C. Smutnicki, "A Fast Taboo Search Algorithm for the Job Shop Problem," *Management Science*, Vol. 42, 797-813, 1996.

[17] Taillard, E., "Parallel Tabu Search Techniques," *ORSA Journal on Computing*, Vol. 6, 108-117, 1994.

[18] Kirkpatrick, S., C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, Vol. 220, 671-680, 1983.

[19] Černy, V., "Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm," *Journal of Optimization Theory and Applications*, Vol. 45, 41-51, 1985.

[20] Van Laarhoven, P. J. M., E. H. L. Aarts, and J. K. Lenstra, "Job Shop Scheduling by Simulated Annealing," *Operations Research*, Vol. 40, 113-125, 1992.

[21] Aarts, E. H. L., P. J. M. Van Laarhoven, J. K. Lenstra, and N. L. J. Ulder, "A Computational Study of Local Search Algorithms for Job Shop Scheduling", *ORSA Journal on Computing,* Vol. 6, 118-125, 1994.

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

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

[24] Syswerda, G., "The Application of Genetic Algorithms to Resource Scheduling," *Proceedings of the Fourth International Conference on Genetic Algorithms*, 502-508, 1991.

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

[26] Kobayashi, S, I. Ono, and M. Yamamura, Syswerda, G., "An Efficient Genetic Algorithm for Job Shop Scheduling Problems," *Proceedings of the Sixth International Conference on Genetic Algorithms*, 506-511, 1995.

[27] Gupta, M. C., Y. P. Gupta, and A. Kumar, "Minimizing Flow Time Variance in a Single Machine System Using Genetic Algorithms," *European Journal of Operational Research*, Vol. 70, 289-303, 1993.

[28] Croce, F., R. Tadei, and G. Volta, "A Genetic Algorithm for the Job Shop Problem," *Computers and Operations Research*, Vol. 22, 15-24, 1995.

[29] Goldberg, D. E. and R. Lingle Jr., "Alleles, Loci, and the Traveling Salesman Problem," *Proceedings of the First International Conference on Genetic Algorithms,* 154-159, 1985.

[30] Oliver, I. M., Smith, D. J., and Holland J. R. C., "A Study of Permutation Crossover Operators on the Traveling Salesman Problem," *Genetic Algorithms and their Applications: Proceedings of the Second International Conference*, 224-230, 1987.

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

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

[33] Giffler, B. and Thompson, G. L., "Algorithms for solving production scheduling problems." *Operations Research*, Vol. 8, 487-503, 1960.

[34] Fox, B. R. and M. B. McMahon, "Genetic Operators For Sequencing Problems," **Foundations of Genetic Algorithms**, Morgan Kaufmann, 284-300, 1991.

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

[36] Fang, H. L., P. Ross, and D. Corne, "A Promising Genetic Algorithm Approach to Job-Shop Scheduling, Rescheduling, and Open-Shop Scheduling Problems," *Proceedings of the Fifth International Conference on Genetic Algorithms*, 375-382, 1993.

[37] Grefenstette, J. J., R. Gopal, B. Rosmaita, and D. Van Gucht, "Genetic Algorithms for the Traveling Salesman Problem," *Proceedings of the First International Conference on Genetic Algorithms*, 160-8, 1985.

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

[39] Bean, J. C., "Genetics and Random Keys for Sequencing and Optimization," *ORSA Journal on Computing,* Vol. 6, No. 2, 154-160, 1994.

[40] Shaefer, C. G. and S. J. Smith, "The ARGOT Strategy II: Combinatorial Optimizations," Technical Report, Thinking Machine Corporation, 1988.

[41] Norman, B. A., "The Random Keys Genetic Algorithm for Complex Scheduling Problems," Unpublished Ph.D. Dissertation, University of Michigan, Ann Arbor, 1995.

[42] Janikow, C. Z. and Z. Michalewicz, "An Experimental Comparison of Binary an Floating Point Representations in Genetic Algorithms," *Proceedings of the Fourth International Conference on Genetic Algorithms*, 31-36, 1991.

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

[44] Davis, L., **Handbook of Genetic Algorithms**, (ed. L. Davis), Van Nostrand, 1991.

[45] Lawrence, S. Supplement to "Resource Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques." GSIA, Carnegie Mellon University, Pittsburgh, PA, October 1984.

[46] Roy, B. and B. Sussmann, Les Problèmes d'ordonnancement avec contraintes disjonctives, Note DS n.9 bis, SEMA, Montrouge, 1964.

[47] Storer, R. H., S. D. Wu, and I. Park, "Genetic Algorithms in Problem Space for Sequencing Problems," *Proceedings of a Joint US-German Conference on Operations Research in Production Planning and Control,* 584-597, 1992.

AIIM SCANNER TEST CHART # 2

Spectra

4 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz:'',./?$0123456789

6 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

8 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

10 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

Times Roman

4 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

6 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

8 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

10 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

Century Schoolbook Bold

4 PT **ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789**

6 PT **ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789**

8 PT **ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789**

10 PT **ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789**

News Gothic Bold Reversed

4 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

6 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

8 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

10 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

Bodoni Italic

4 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

6 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

8 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

10 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

Greek and Math Symbols

4 PT ΑΒΓΔΕΞΘΗΙΚΛΜΝΟΠΦΡΣΤΥΩΧΨΖαβγδεξθηικλμνοπφρστυωχψζ≧∓'',. / ≦±=≠°><⊁⊀⩾⩽≡

6 PT ΑΒΓΔΕΞΘΗΙΚΛΜΝΟΠΦΡΣΤΥΩΧΨΖαβγδεξθηικλμνοπφρστυωχψζ≧∓'',. / ≦±=≠°><⊁⊀⩾⩽≡

8 PT ΑΒΓΔΕΞΘΗΙΚΛΜΝΟΠΦΡΣΤΥΩΧΨΖαβγδεξθηικλμνοπφρστυωχψζ≧∓'',. / ≦±=≠°><⊁⊀⩾⩽≡

10 PT ΑΒΓΔΕΞΘΗΙΚΛΜΝΟΠΦΡΣΤΥΩΧΨΖαβγδεξθηικλμνοπφρστυωχψζ≧∓'',. / ≦±=≠°><⊁⊀⩾⩽≡

White

Black

Isolated Characters

e	m	1	2	3	a
4	5	6	7	o	°
8	9	0	h	I	B

A4 Page 6543210

MESH HALFTONE WEDGES

65

85

100

110

133

150

A4 Page 6543210

RIT ALPHANUMERIC RESOLUTION TEST OBJECT, RT-1-71

PRODUCED BY GRAPHIC ARTS RESEARCH CENTER

ROCHESTER INSTITUTE OF TECHNOLOGY, ONE LOMB

MEMORIAL DRIVE, ROCHESTER, NEW YORK 14623