Pergamon

0305-0548(93)E0021-K

Computers Ops Res. Vol. 22, No. 1, pp. 85–99, 1995
Copyright © 1994 Elsevier Science Ltd
Printed in Great Britain. All rights reserved
0305-0548/95 $7.00+0.00

SCHEDULING IN A SEQUENCE DEPENDENT SETUP ENVIRONMENT WITH GENETIC SEARCH

PAUL A. RUBIN† and GARY L. RAGATZ‡

Department of Management, Eli Broad Graduate School of Management, Michigan State University, East Lansing, MI 48824-1121, U.S.A.

Scope and Purpose—This paper considers the application of a genetic search algorithm to a common sequencing problem, in which we wish to determine the sequence of a set of *n* jobs on a single machine that will minimize the total tardiness of the jobs. Each job has associated with it a processing time and a due date, by which we wish to complete the job. An unusual aspect of this problem is that the job setup times are sequence dependent. That is, the processing of each job on the machine is preceded by a setup whose duration depends on the immediate predecessor job. Although an optimal branch-and-bound algorithm for the problem is available, its computational requirements are impractical for large problems. In the paper, we describe a genetic search algorithm for the problem and compare the performance of the genetic search to a pure random search and the branch-and-bound algorithm, on a set of 32 test problems.

Abstract—Work on scheduling in the presence of sqeuence dependent setup times has generally focused on minimizing the total setup time (or cost) or minimizing the makespan of a set of jobs. We explore the problem of sequencing to minimize the total tardiness of a set of jobs in a single-stage process where setup times are sequence dependent. In particular, we examine the efficacy of using genetic search to develop near optimal schedules in this environment.

INTRODUCTION

One common simplification made in studies of shop scheduling is that setup times are independent of the sequence in which the jobs are processed. Setup times are then either assumed to be a part of the operation time of the jobs or are ignored altogether.

Although this assumption may be reasonable for some manufacturing systems, evidence suggests that in many systems, sequence dependent setup times are the rule rather than the exception. Panwalkar *et al.* [1] found, in a survey of industrial schedulers, that 70% of the schedulers reported at least some operations they schedule are subject to sequence dependent setup times. Thirteen percent of the schedulers reported that *all* the operations they schedule are subject to sequence dependent setups. Panwalkar *et al.* also reported that industrial schedulers overwhelmingly named meeting due dates as their most important scheduling criterion. The other choices available to them were minimizing processing time, minimizing setup time, and minimizing in-process inventory cost.

Despite the apparent practical importance of the problem, little work has appeared on scheduling to meet due dates in the presence of sequence dependent setup times. Rather, the work on scheduling in this environment has tended to focus on minimizing the total setup time (or cost) or minimizing the makespan of a set of jobs. In this paper, we explore the problem of sequencing to minimize the total tardiness of a set of jobs in a single-stage process where setup times are sequence dependent. In particular, we examine the efficacy of using genetic search [2] to develop near optimal schedules in this environment.

†Paul A. Rubin is an Associate Professor of Management Science in the Eli Broad Graduate School of Management at Michigan State University. He received degrees in mathematics from Princeton University (A.B.) and Michigan State University (Ph.D.). Dr Rubin's primary research interest is in the application of mathematical programming techniques to problems in operations management. His previous publications have appeared in an assortment of journals.

‡Gary L. Ragatz is an Associate Professor of Operations Management in the Eli Broad Graduate School of Management at Michigan State University. He holds a Ph.D. in operations management from Indiana University. His work on production scheduling and capacity planning has appeared in *Decision Sciences*, *The Journal of Operations Management*, *Production and Inventory Management Journal*, and *The International Journal of Production Research*.

PROBLEM DESCRIPTION

Suppose we have N jobs, indexed 1, 2, ..., N, that are all available for processing at time zero on a continuously available machine that can process only one job at a time. At time zero, the machine has completed processing job 0. For each job j to be done, there is a required processing time p_j, a due date d_j, and a setup time s_{ij} incurred when job j follows job i in the processing sequence.

Define Q to be a sequence of the jobs, $Q = \langle Q(0), Q(1), \ldots, Q(N)\rangle$, where $Q(j)$ is the index of the jth job in the sequence and $Q(0) = 0$. The completion time of the jth job in the sequence is

$$c_{Q(j)} = \sum_{k=1}^{j} [s_{Q(k-1)Q(k)} + p_{Q(k)}]$$

and the tardiness of the jth job in the sequence is

$$t_{Q(j)} = \max\{0, c_{Q(j)} - d_{Q(j)}\}.$$

We would like to find a sequence that minimizes the total tardiness of the jobs. Total tardiness for a sequence Q is

$$T_Q = \sum_{j=1}^{N} t_{Q(j)}.$$

Tardiness is a "regular" performance measure [3], which means that its value can be increased only by increasing the completion time of a job. To minimize a regular measure, it is sufficient to consider only permutation schedules (i.e. sequences containing no inserted idle time).

The problem is combinatorial in nature, and for problems with even a modest number of jobs to be sequenced the number of potential solutions is huge. A related problem, the simple tardiness problem (without sequence dependent setup times), has been shown to be NP-hard [4]. The inclusion of sequence dependent setup times does nothing to simplify the problem, so it seems likely that the sequence dependent version of the tardiness problem is NP-hard as well. The simple tardiness problem has been found to be amenable to solution through a combination of dominance relationships that rule out certain sequences as optimal, and branch-and-bound methodology.

Unfortunately, the nature of the problem changes when setup times are sequence dependent. In this environment, the dominance relationships used for the simple tardiness problem are not valid. Furthermore, most of the bounding procedures for the simple tardiness problem depend on knowing the makespan (total setup and processing time) of the full set of jobs or a subset of the set of jobs. The makespan for a set of jobs is fixed when setup times are sequence independent, but when setup times are sequence dependent, the makespan depends on the sequence of the jobs.

Another related problem, the travelling salesman problem (TSP), has also been attacked successfully using branch-and-bound methods. An important difference between the TSP and the tardiness problem, however, is that a job's *absolute* position in the sequence, in addition to its relative position, is important in the tardiness problem.

LITERATURE REVIEW

As mentioned above, most of the research on scheduling in the presence of sequence dependent setups has focused on minimizing setup time or makespan. In this section, we survey some of this work, and identify a few papers that have dealt with other objectives. We then briefly examine some research on the use of genetic search for solving combinatorial optimization problems.

Much of the work on scheduling in the presence of sequence dependent setups has dealt with static, single stage problems. For a single processor, minimizing total setup times equates to solving the well known TSP. Quite a few optimizing procedures have been proposed for the TSP [31]. The most popular approach in these procedures has been branch-and-bound [6]. An excellent summary and evaluation of these branch-and-bound methods is available in Balas and Toth [7]. Golden and Stewart [8] provide a summary of construction and improvement heuristics that have been developed for the TSP. Gavett [9], Lockett and Muhlemann [10], Hayes *et al.* [11], and White and Wilson [12] examined heuristic methods specifically in the context of the single-machine scheduling problem. Prabhakar [13] considered the case of maximizing profit from a group of parallel processors subject to sequence dependent setup times. The problem in this case was to sequence production runs of several different products, which could be split across processors, during a single planning period. No due dates nor measures of job tardiness were included.

Barnes and Vanston [5] considered an objective function involving weighted flow time and setup cost in a single machine setting. Picard and Queyranne [14] suggest that an algorithm they developed for the weighted tardiness problem could be extended to consider sequence dependent setups. In both cases, however, only setup *cost*, and not setup *time*, is sequence dependent. This offers the advantage that the makespan of the set of jobs (or a subset of the jobs) does not depend on the ordering of the jobs.

Several authors have considered multi-stage problems. Corwin and Esogbue [15], Srikar and Ghosh [16, 17], and Gupta and Darrow [18] studied algorithms for minimizing makespan in the sequence dependent flowshop problem. Gupta [19] presented a branch-and-bound based algorithm for minimizing total setup time in the N-job, M-machine job-shop problem.

A few authors have included due dates or product demand as a constraint on a static problem in which the objective is to minimize setup time. These include Glassey [20] and Driscoll and Emmons [21], who studied single machine problems, and Uskup and Smith [22], who studied a two-stage flowshop. Although these formulations include consideration of due dates or demand rates, a zero tardiness or zero stockout schedule must exist in order for the algorithms to work.

Ragatz [23] developed a branch-and-bound procedure for the minimum tardiness problem with sequence dependent setup times. The bounds used in the procedure were weak, and the performance of the procedure was highly dependent on problem characteristics. In the tests reported, even on problems of 16 jobs, the procedure was unable to solve to completion (in 1.5 million nodes) five out of 32 test problems. Ragatz recommended that further work be conducted in two areas: (1) development of stronger bounding procedures or stronger dominance tests, and (2) development of good heuristic approaches to the problem.

The conclusion from a review of the literature involving sequence dependent setup scheduling methods is that we know a good deal, at least in the single machine case, about scheduling to minimize setup times or flow times. Understanding of multi-stage systems is much less complete. In neither setting, however, is very much known about scheduling to meet due dates. The results reported by Ragatz suggest that the minimum tardiness problem with sequence dependent setup times is difficult to solve optimally.

In recent years, the use of heuristic search procedures for combinatorial optimization problems has attracted attention. Goldberg [2] provides a good introduction to genetic search algorithms, including a summary of applications. Glover and Greenberg [25] offered a review of five methods: genetic search, neural networks, simulated annealing, tabu search, and target analysis. They concluded that, while these methods have shown promise in some investigations, experience with them is not wide enough to generalize. In particular, they cited the need to study whether these methods, or variants of them, are effective on important real world problems.

Several authors have studied the use of genetic search for solving the travelling salesman problem. Goldberg and Lingle [24] and Grefenstette *et al.* [26] examined representations for the TSP and developed reproduction operators that guarantee valid solutions to the problem. Oliver *et al.* [27] extended this exploration of reproduction operators. Each of these studies also examined the effects of various algorithm parameters such as population size and mutation rates on the performance of the algorithm. These studies must be considered exploratory, and none offered general conclusions about the efficacy of genetic search for the TSP.

Davis [28] presented an application of genetic search to a simple job shop scheduling problem. The focus of the paper was on developing a workable representation of the problem. Only a single example problem was presented, with very limited computational experience.

The work reported to date on the use of genetic search for combinatorial optimization problems is encouraging enough to warrant further study, but this body of work is still too incomplete to make any firm projections about the success of genetic search on a wide range of combinatorial problems.

GENETIC SEARCH MODEL

In this section, we document the design of our genetic search procedure.

Population size

We performed our experiments using populations of 40 schedules in each generation. Earlier tests with population sizes of 30 and 50 produced results not significantly different from those with size 40. Several runs employing a population size of 20 yielded somewhat poorer performance, suggesting that 20 might be too small a gene pool. Runs with populations of 60 and 80 did not show any significant improvement in rate of convergence. Since computation time grows roughly linearly with population size, lack of noticeable improvements in convergence rates discouraged us from using larger population sizes.

Coding

We represent a schedule as a permutation vector of the index set $\{1, \ldots, N\}$. The basic unit of information (*allele*) is the (integer) job index. We selected this in preference to binary encoding, and manipulation of "genetic" information at the bit level, to avoid dealing with large numbers of candidate schedules that would not represent valid permutation vectors.

Reproduction

In designing a method of reproduction, we sought to produce offspring that would be valid permutation vectors retaining desirable traits of the parent schedules. Several crossover procedures meeting this description have been proposed for the travelling salesman problem (see, for example, Oliver *et al.* [27]). While our problem has the same feasible set as a travelling salesman problem, the difference in objective affects our interpretation of what constitutes a desirable trait. In the travelling salesman problem, the absolute position of a node in a tour is less important than its position relative to other nodes. Indeed, when the arc costs are symmetric, the reverse of any tour has the same objective value as the original tour. Since our interest is in minimizing tardiness, however, we expect the absolute position of each job to be relevant to the total cost of the schedule. Intuition suggests that jobs with early (late) due dates should tend to appear early (late) in the schedule. Thus a "strong" parent should pass along both information on the adjacency of pairs of jobs and information on the absolute positioning of the jobs.

Table 1 illustrates our reproduction scheme, using an example with six jobs. At the outset, we partition the jobs into two subsets, here $G_1 = \{1, 2, 3\}$ and $G_2 = \{4, 5, 6\}$. Our practice is to order the jobs according to due date and partition into sets with early (late) due dates. A few experiments in which we partitioned the jobs according to short or long processing times exhibited no improvement in search performance.

We arbitrarily designate the two parent schedules "mother" and "father". For each offspring produced, one parent dictates the set of slots in the schedule to be assigned to each group of jobs. We then insert the jobs from the first group into the set of slots allocated to them, in the same

Table 1. Example of reproduction (Group 1 = {1, 2, 3}; Group 2 = {4, 5, 6}

Mother = ⟨2 6 5 3 1 4⟩
Father = ⟨1 4 3 6 5 2⟩

Sequencing within groups		Assignment of slots to groups*	

Group 1	Group 2	Mother
		Group 1 → {a, d, e}
		Group 2 → {b, c, f}
			Father
			Group 1 → {a, c, f}
			Group 2 → {b, d, e}

Mother ⟨2, 3, 1⟩	Mother ⟨6, 5, 4⟩	⟨2 6 5 3 1 4⟩†	⟨2 6 3 5 4 1⟩
	Father ⟨4, 6, 5⟩	⟨2 4 6 3 1 5⟩	⟨2 4 3 6 5 1⟩
Father ⟨1, 3, 2⟩	Mother ⟨6, 5, 4⟩	⟨1 6 5 3 2 4⟩	⟨1 6 3 5 4 2⟩
	Father ⟨4, 6, 5⟩	⟨1 4 6 3 2 5⟩	⟨1 4 3 6 5 2⟩‡

*Schedule slots are labelled a through f consecutively.
†Clone of mother.
‡Clone of father.

relative order that they appear in one parent, and do the same (possibly using a different parent's order) with the jobs from the second group. Since a parent must be designated for each of three operations (grouping the slots, sequencing the first group, sequencing the second group), a single pair of parents can produce a "litter" of up to $2 \times 2 \times 2 = 8$ offspring, that will include one clone of each parent. Note that in the event that either group of jobs has the same relative sequence in both parents, the children will not all be distinct: if the parents sequence one group the same, there will be four pairs of twins, while if the parents sequence both groups the same (but in different slots) there will be two sets of quadruplets.

Referring to the illustration in Table 1, the "child" ⟨2 4 3 6 5 1⟩ is produced as follows. First, the "father" determines the partitioning of slots. In this case, the first, third and sixth slots (*a*, *c* and *f*) will be allocated in some order to the first group of jobs, since those are the slots the first group occupies in the "father" schedule. Next, the "mother" determines the sequencing of the first three jobs: within their allocated slots, job 2 will appear first, then job 3, and finally job 1. Last, the "father" schedule assigns the order in which the second group of jobs will fill their allocated slots: job 4, then job 6, then job 5.

Winnowing

Our reproduction procedure generates up to eight distinct offspring at a time. We were concerned that, particularly when using population sizes of 30–50, a few parents might account for the entire next generation, resulting in excessive narrowing of the genetic pool. Accordingly, we set a maximum number of offspring to retain from any litter, with the offspring to be retained chosen randomly without replacement. Retention of offspring is *not* based on their fitness, with the one exception that the best offspring in a litter is always kept if it improves on the current incumbent solution. In our experiments, winnowing litters of eight to two or four did not show any conclusive gains, and we abandoned it to conserve computation time (winnowing not only consumes time itself in sampling, it requires additional matings to produce the necessary number of offspring to populate the next generation). We continue to winnow the last litter in each generation if not all eight children are required to fill out the next generation.

Mutation

Since our coding structure treats a schedule as a permutation vector, as opposed to a binary string, mutation must be implemented by some type of exchange of position among the jobs. In particular, then, we mutate the schedule as a whole, rather than a piece of it in isolation, and so the mutation probability is the probability a schedule is modified, not the probability that an individual element (job) within a schedule is modified.

In our early experiments, we mutated a schedule by randomly interchanging two jobs. With this mutation process, and the reproduction scheme explained above, genetic search appeared to progress quite slowly. We then changed to a mutation procedure that tests *seriatim* all interchanges of adjacent pairs of jobs, accepting each interchange if it improves the schedule, until no further improvements are noted. In the event that a schedule being mutated cannot be improved by adjacent pairwise swaps, the mutation procedure exchanges a randomly chosen pair of jobs, without regard to improvement or degradation of the total tardiness of the schedule. This mutation procedure was considerably slower than randomly exchanging a pair of jobs, but ensured that mutation would improve most individual schedules.

With the new form of mutation, genetic search progressed more rapidly, although each generation took longer to produce. Acceptable progress was made in experiments with a 16 job test problem. On a 25 job test problem, though, we found the rate of progress to be too slow, and so we modified the mutation procedure to attempt *seriatim all* pairwise swaps, without restriction to contiguous pairs of jobs. In one experiment with a 25 job test problem, using the adjacent exchange mutation procedure, five replications of 200 generations with a 10% mutation rate produced schedules that on the average were approx. 1.8% worse than the best solution obtained by branch-and-bound. After changing to the nonadjacent exchange mutation procedure, five replications of only 100 generations of the same problem, with a 1% mutation rate, produced schedules that on the average were only 0.22% worse than the best known solution, and in one replication matched the best known solution. The price for this improvement in convergence rate, unfortunately, is a sharp increase in the computation time each individual mutation takes.

Immigration

A key principle of genetic search is the maintenance of diversity in the gene pool. The primary mechanism for maintaining diversity is the reproductive process. In some of our early experiments with a 16 job problem, however, it appeared that the population was becoming too in-bred. Accordingly, we added a provision for each new generation to include some randomly generated "immigrants". Gains from immigration were not obvious, but forming roughly 10% of each new generation through immigration seemed to lead to reasonable performance.

Fitness

The value of each schedule is the total tardiness of all jobs. Our fitness measure, which we seek to maximize, is the reciprocal of the tardiness. To avoid division by zero, we increase the tardiness by one before inverting it. With this fitness measure, the probability (before scaling) of selecting a schedule for reproduction is approximately inversely proportional to its tardiness.

After several generations, differences in tardiness among the schedules tend to be a comparatively small percentage of total tardiness, with the result that the selection probabilities (before scaling) tend to resemble a uniform distribution. We experimented with squaring the fitness function, to force greater diversity in the selection probabilities (and emphasize the choice of good parents). This seemed to accelerate progress of the search in the early generations, but appeared to hinder its progress in later stages, and so we returned to the original fitness measure. We also tried using the reciprocal of the difference between the tardiness of a schedule and a lower bound on optimal tardiness. Again, this seemed to induce slightly faster improvement in early stages of a run and slighly slower improvement later.

Scaling

Following the lead of Goldberg [2], we rescale the fitness function $f(\,)$ in each generation prior to randomly selecting the parents, using the affine transformation $g(x)=a\cdot f(x)+b$. We adopted the parameter values suggest in Goldberg's book: if

$$f_{\min}>\frac{h\cdot f_{\text{avg}}-f_{\max}}{h-1}$$

then

$$\left\{\begin{aligned} a&=\frac{(h-1)\cdot f_{\text{avg}}}{f_{\max}-f_{\text{avg}}}\\ b&=\frac{f_{\text{avg}}\cdot(f_{\max}-h\cdot f_{\text{avg}}}{f_{\max}-f_{\text{avg}}}\end{aligned}\right\};$$

otherwise

$$\left\{\begin{aligned} a&=\frac{f_{\text{avg}}}{f_{\text{avg}}-f_{\min}}\\ b&=\frac{-f_{\min}\cdot f_{\text{avg}}}{f_{\text{avg}}-f_{\min}}\end{aligned}\right\}.$$

Here $f_{\max}$, $f_{\min}$ and f_{avg} denote the maximum, minimum and mean unscaled fitness values of the current population, and h is the desired expected frequency with which the best member of the current population will be selected for mating. Recommended values for h in Goldberg's book range from 1.2 to 2.0. We performed most of our experiments with $h=1.5$, and did not find performance to be particularly sensitive to the choice of h.

Elitism

Throughout our experiments, we adopted the practice of *elitism* [29], which requires that the best known solution always be included in the genetic pool. At the start of the reproduction stage, we expand the parent population by adding the incumbent schedule(s).

Uniqueness

Our implementation of genetic search includes a provision to eliminate duplicate schedules in each generation. This is done during the reproduction step, leading to the removal of some offspring. Consequently, the number of offspring created during any one generation exceeds the size of the population. As one would expect, the ratio of the number of offspring created to the number kept, after adjustment for winnowing as discussed above, rose as runs were extended to more generations and as population sizes fell, both factors that would foster homogeneity in the population; the ratio fell as mutation and immigration rates rose, both of which would foster heterogeneity. The ratio also rose as the proportion of each litter winnowed fell, since retaining a higher proportion of siblings leads to homogeneity. Typical values of the ratio ranged between 1.4 and 2.7 offspring produced per offspring kept (excluding the effect of winnowing). The relatively large ratios suggest that duplication of schedules is a frequent occurrence. This is consistent with evidence from a few experiments in which uniqueness was not enforced: in those cases, populations quickly lost all diversity. In one run with 25 jobs and a population size of 40, after 200 generations without enforcement of uniqueness, the population contained only five distinct schedules, one of which appeared 35 times.

TEST PROBLEMS

For this study, we generated test problems based on a subset of the experimental design used by Ragatz [23], and evaluated the performance of genetic search on the minimum tardiness problem with sequence dependent setup times. For comparison purposes, we also tried applying our mutation operator to a random sample of sequences. Each test problem was solved 20 times by genetic search and 20 times by random mutation. We also attempted to solve each of these problems using Ragatz's branch-and-bound procedure, to provide a benchmark for the performance of the genetic search heuristic.

Table 2. Experimental results

	Experimental design			Branch-&-Bound		Genetic search		Random mutation	
No. of jobs	Processing time variance	Tardiness factor	Due date range	Final solution	Seconds to match G. S.	% Worse than B-&-B	Run time (s)	% Worse than B-&-B	Run time (s)
15	Low	Low	Narrow	90	8.0	11.1	64.9	5.3	69.1
15	Low	Low	Wide	0	0.1	0.0	1.3	0.0	0.3
15	Low	Moderate	Narrow	3418	16.3	0.6	61.7	0.1	58.6
15	Low	Moderate	Wide	1067	23.8	0.0	5.2	0.0	1.4
15	High	Low	Narrow	0	0.2	0.0	0.2	0.0	0.3
15	High	Low	Wide	0	0.2	0.0	1.2	0.0	0.3
15	High	Moderate	Narrow	1861	38.1	0.7	50.6	0.1	67.1
15	High	Moderate	Wide	5660	12.7	0.3	57.2	0.1	37.2
25	Low	Low	Narrow	266	4.9	3.8	137.2	3.5	167.7
25	Low	Low	Wide	0	0.7	0.0	2.6	0.0	1.2
25	Low	Moderate	Narrow	3511	6.8	0.1	216.4	0.2	168.0
25	Low	Moderate	Wide	0	0.7	0.0	2.6	0.0	1.2
25	High	Low	Narrow	0	4091.7	0.0	6.4	0.0	3.1
25	High	Low	Wide	0	0.7	0.0	3.0	0.0	1.3
25	High	Moderate	Narrow	7225	293.3	1.8	187.3	1.7	178.4
25	High	Moderate	Wide	2327	4712.8	−5.3	7.1	−7.2	3.4
35	Low	Low	Narrow	60	1449.5	2.1	294.0	−2.5	427.4
35	Low	Low	Wide	0	2.1	0.0	5.7	0.0	3.2
35	Low	Moderate	Narrow	18,381	5652.1	−0.4	63.4	−0.6	66.7
35	Low	Moderate	Wide	19,454	2756.9	0.1	409.2	0.4	367.2
35	High	Low	Narrow	291	244.1	10.9	373.7	9.7	340.6
35	High	Low	Wide	0	2.1	0.0	5.4	0.0	3.3
35	High	Moderate	Narrow	13,339	1111.3	0.7	361.6	0.9	490.1
35	High	Moderate	Wide	7069	5371.1	−21.1	11.1	−19.5	8.6
45	Low	Low	Narrow	120	9.6	52.4	981.4	51.5	551.7
45	Low	Low	Wide	0	5.3	0.0	11.6	0.0	6.7
45	Low	Moderate	Narrow	27,139	5.4	0.8	807.1	1.2	759.7
45	Low	Moderate	Wide	16,041	6464.9	−0.4	822.0	−0.3	620.7
45	High	Low	Narrow	234	5.3	41.1	1036.2	41.3	690.1
45	High	Low	Wide	0	5.3	z0.0	11.9	0.0	7.3
45	High	Moderate	Narrow	25,395	6286.2	−0.7	80.8	−0.7	76.4
45	High	Moderate	Wide	24,335	5954.5	−0.9	62.8	−1.3	47.3

We ran eight test problems at each of four levels (15, 25, 35, 45) for the number N of jobs to be scheduled, for a total of 32 test problems. For each value of N, the eight test problems represented all treatments from a $2 \times 2 \times 2$ experimental design. The first factor, processing time variance, was set either low or high. The second, called the *tardiness factor*, was set either to a low or to a moderate value. The tardiness factor is roughly equivalent to the expected percentage of jobs in a randomly generated sequence that would be tardy. The experience of Rinooy Kan *et al.* [30] with sequence independent setups and Ragatz [23] with sequence dependent setups suggests that branch-and-bound has an easy time finding optimal solutions when the tardiness factor takes an extreme value (near 0 or 100) and a difficult time when it takes a moderate value (near 50). The final experimental factor, the range of due dates in the job sequence, was set either narrow or wide. Ragatz's experiments indicated that problems are harder for branch-and-bound to solve when the due date range is narrow.

We limited the branch-and-bound procedure to two million nodes. It solved all eight test problems with 15 jobs to optimality, but only four of the 25 job problems and two each of the 35 and 45 job problems. For the remaining problems, we used the lowest-tardiness job sequence found by the branch-and-bound procedure (the current incumbent solution) when the procedure was stopped as a benchmark for the genetic search heuristic. We also tracked the progress of the branch-and-bound procedure as it improved upon its initial solution.

All problems were solved on a 33 MHz Intel 80486-based workstation running MS-DOS. For comparison puroposes, computation times are shown below, but the reader should note that the programs were written by different programmers, using different languages (Watcom WATFOR-77 for the search methods, Borland Turbo Pascal for the branch-and-bound procedure), and so run times should be viewed as highly approximate.

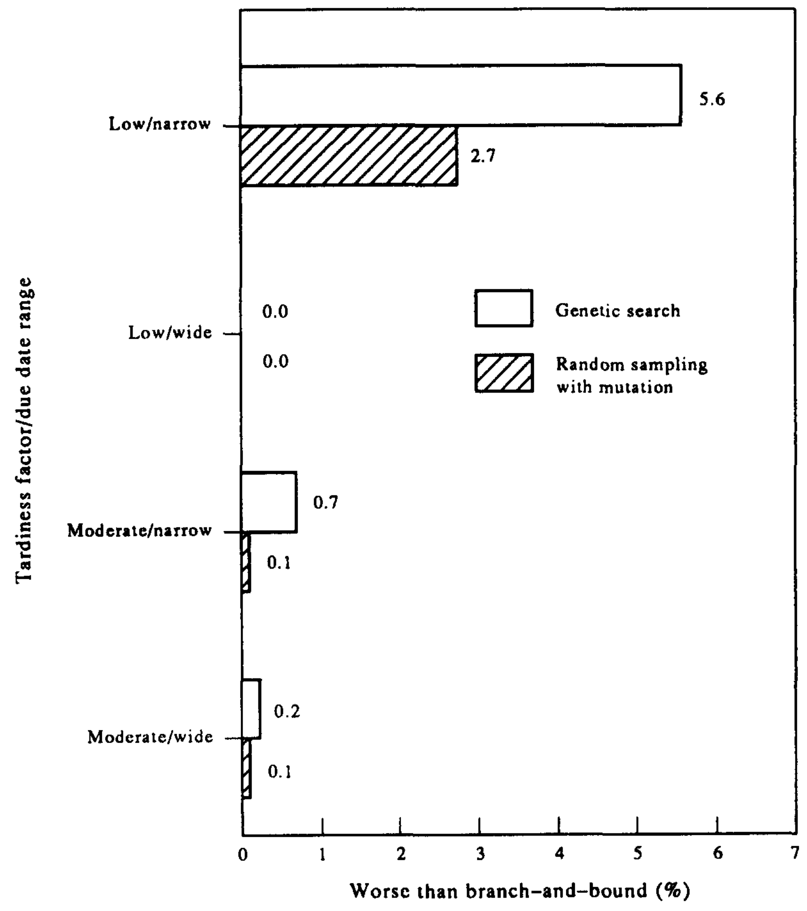

Fig. 1. Solution quality. 15 Job problems.

RESULTS

All results reported here are based on genetic search runs with a population size of 40, a mutation probability of 0.005, four immigrants per generation, and a target reproduction frequency of 1.5 for the current incumbent. We imposed elitism and uniqueness throughout all runs, and winnowed offspring only when necessary to preserve uniqueness or to truncate the last litter of a generation. Besides mutating randomly selected offspring, we automatically mutated each new incumbent encountered.

Table 2 summarizes the results (averaged across 20 replications for both search procedures) for each of the 32 problems we solved. Results for genetic search and random mutation show (as a percentage of the tardiness of the branch-and-bound solution) the average amount by which the tardiness of the search solution exceeded that of the branch-and-bound solution. Negative values indicate that the average search solution was superior to the best solution found by branch-and-bound within two million nodes. Figures 1–4 show graphically the averages of these results across the two levels of the processing time variance factor.

Genetic search ran for 300 generations or until it matched or improved on the branch-and-bound solution (which was either the optimum or the best incumbent found in two million nodes). We assigned random sampling with mutation a run time limit (ranging from 130 s with 15 jobs to 1500 s with 45 jobs) approximating the time required by genetic search to process a full 300 generations. As with genetic search, random mutation runs stopped as soon as they matched or improved on the branch-and-bound solution. The time we report for branch-and-bound is the time it required either to locate a solution with tardiness no worse than the average tardiness of the genetic search replications or, when that average was never achieved, the time branch-and-bound

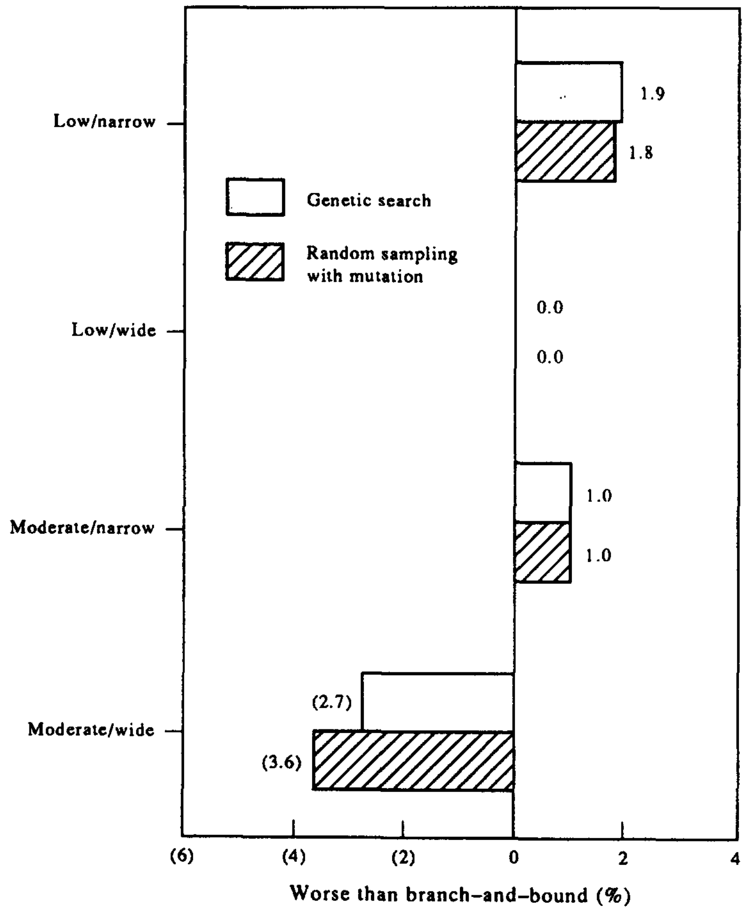

Fig. 2. Solution quality. 25 Job problems.

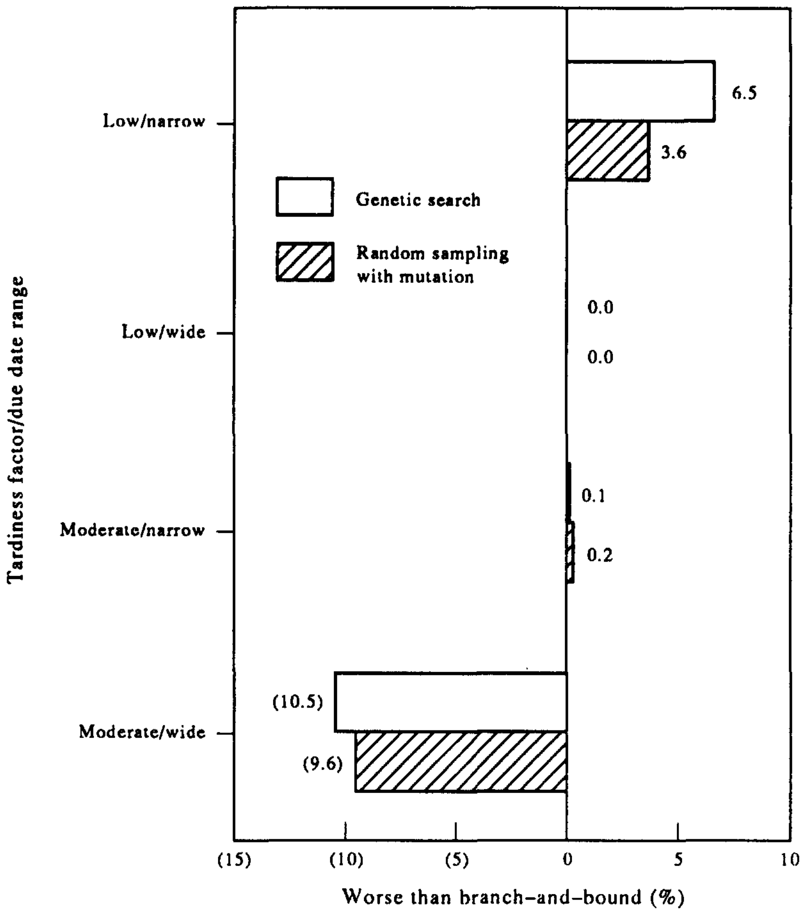

Fig. 3. Solution quality. 35 Job problems.

took to process two million nodes. These times, again averaged across the two levels of the processing time variance factor, are shown in Figs 5–8.

Table 3 is an attempt to summarize the comparative performance of the two search procedures, which produced similar results, to that of branch-and-bound. In it we show, for the eight experimental treatments, at which problem sizes each of the following occurred: search produced better solutions (invariably in less time) than branch-and-bound; search produced slightly inferior solutions, but found them substantially faster than did branch-and-bound; or branch-and-bound found better solutions in less time than search used. Two of the eight combinations, in which the tardiness factor and due date range both took their "easy" levels (low tardiness factor, wide due date range), produced trivial problems, containing multiple solutions with zero tardiness. Two other problems (out of 32) were trivial and are omitted from the table, as is one nontrivial problem (25 jobs, high processing time variance, moderate tardiness factor, narrow due date range), in which search found solutions moderately (nearly 2%) worse than branch-and-bound in substantially (about 38%) less time.

CONCLUSIONS

From Table 3, it appears that processing time variance is not a determinant of comparative performance. We interpret Table 3 as follows. When the tardiness factor is moderate (the harder level) and due date range is wide (the easier level), either search technique produces "good" solutions faster than does the branch-and-bound procedure. When the tardiness factor is low (the easier level) and due date range is narrow (the harder level), branch-and-bound is superior to either search technique. When both factors are at their easy levels (low tardiness factor, wide due date range), the problem is trivial, and all three techniques locate an optimal schedule almost immediately: in

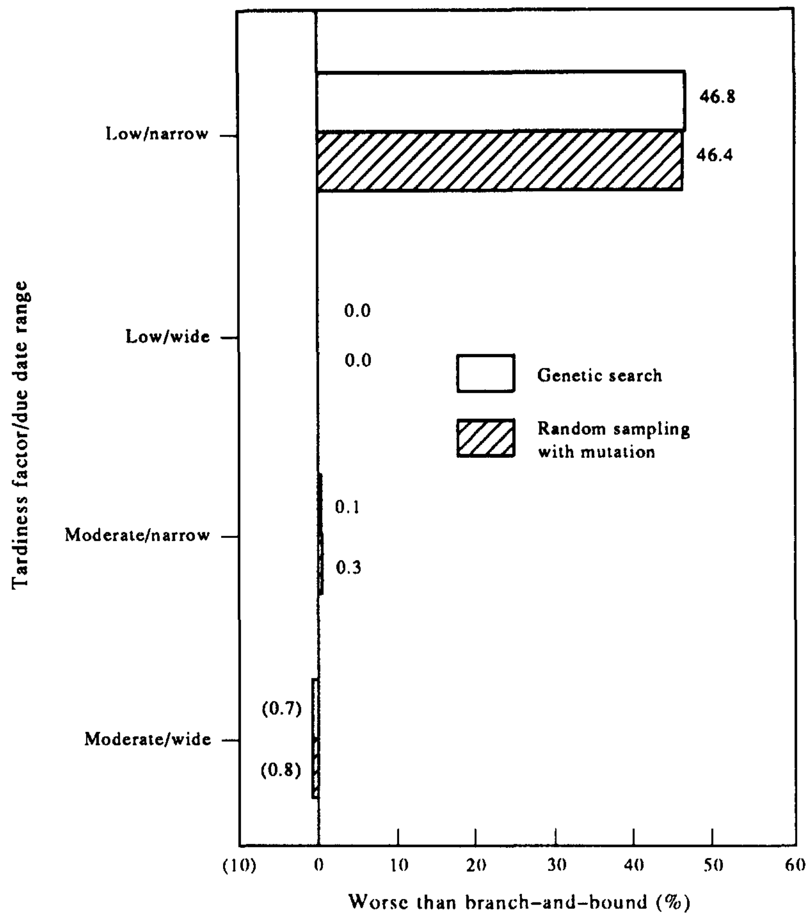

Fig. 4. Solution quality. 45 Job problems.

fact, sequencing jobs according to their due dates appears to work in general. Finally, when both factors are at their hard levels (moderate tardiness factor, narrow due date range), neither search nor branch-and-bound appears to be consistently superior.

Genetic search and random sampling with mutation appear to behave very similarly, with perhaps a slight edge in our experiments accruing to random mutation. We need to perform further tests, including additional tuning of the genetic search parameters, before drawing definitive conclusions. One consideration to note is that as the number N of jobs increases, the time required for mutation grows much faster than the time required for reproduction. Thus for equal execution times, the ratio of the number of mutations done in a genetic search run to the number done in a random mutation run will converge to unity, assuming that the mutation rate for genetic search is held fixed. We need to experiment with reduction of the mutation rate as N increases, to see if the reproduction mechanism can provide genetic search with an edge.

SUMMARY

The results of our experiments suggest that for some minimum tardiness scheduling problems with sequence dependent setups, the genetic search algorithm is competitive with, if not superior to, branch-and-bound in finding "good" (but not necessarily optimal) schedules fairly quickly. For finding *optimal* solutions, however, branch-and-bound has one inescapable advantage: while both algorithms can find an optimal schedule, only branch-and-bound can recognize in a finite number of steps that it has done so.

One avenue of future research lies in the integration of the genetic search and branch-and-bound procedures. With tighter bounds, it might be expeditious to perform a few fairly short runs with genetic search, and use the best solution found by it as an initial incumbent in the branch-and-bound

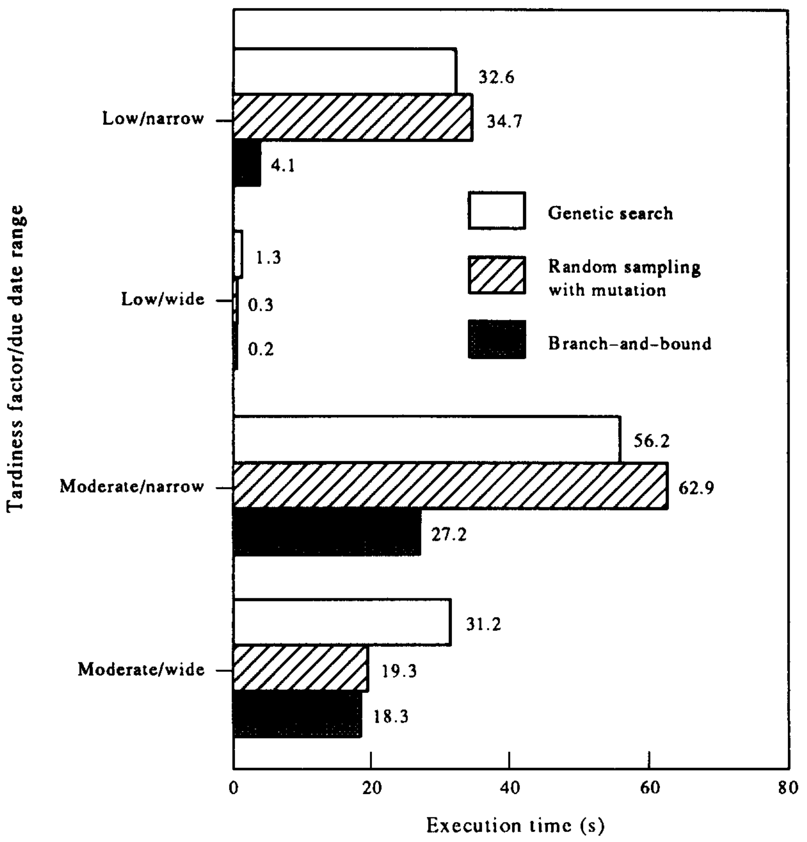

Fig. 5. Execution time. 15 Job problems.

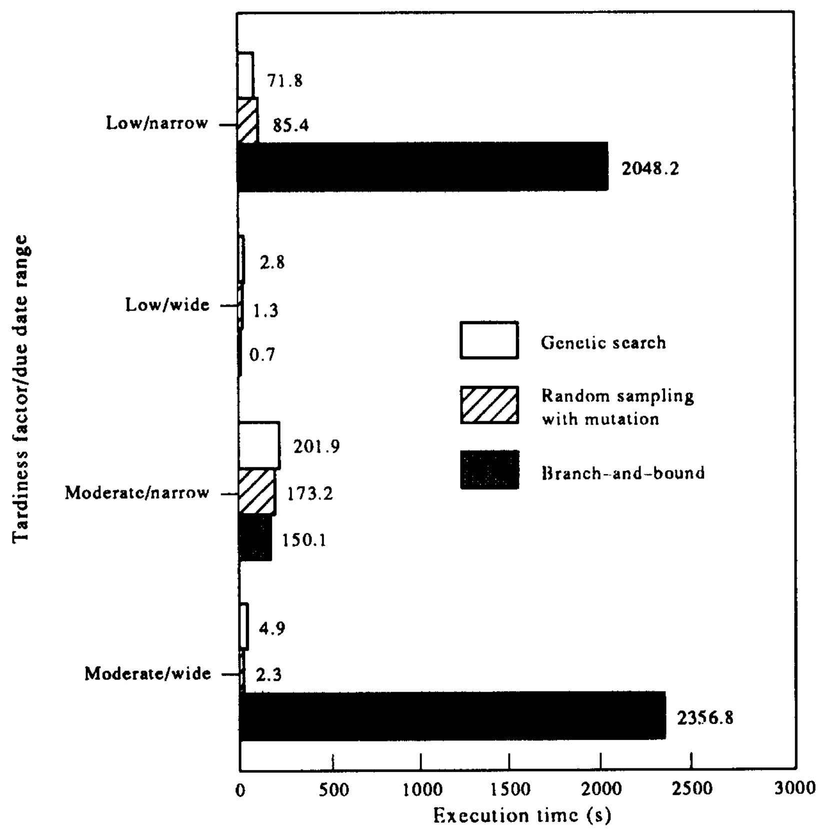

Fig. 6. Execution time. 25 Job problems.

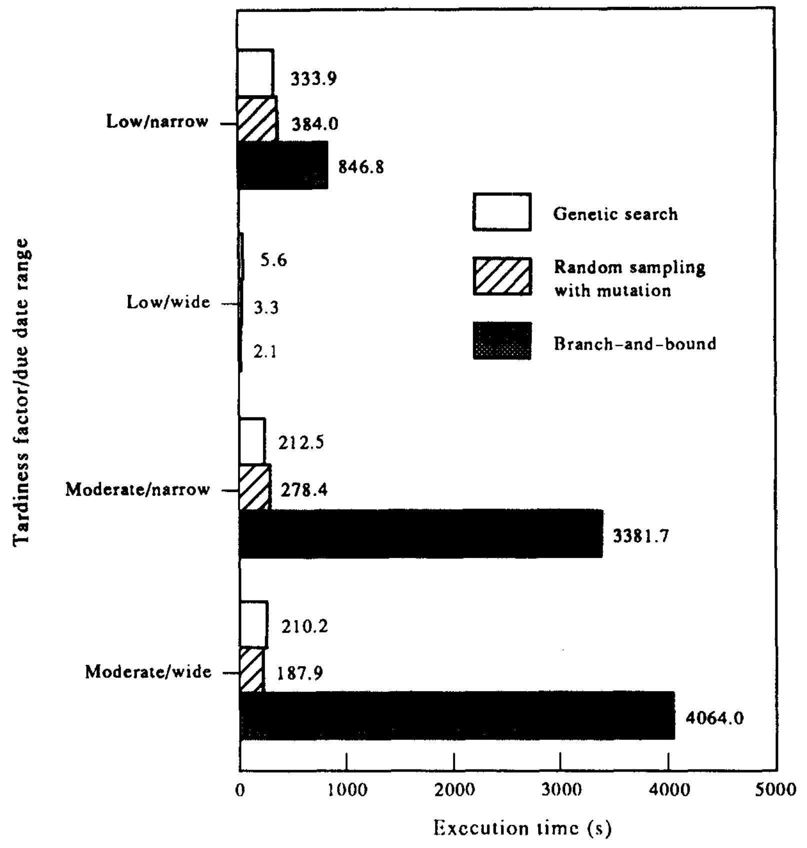

Fig. 7. Execution time. 35 Job problems.

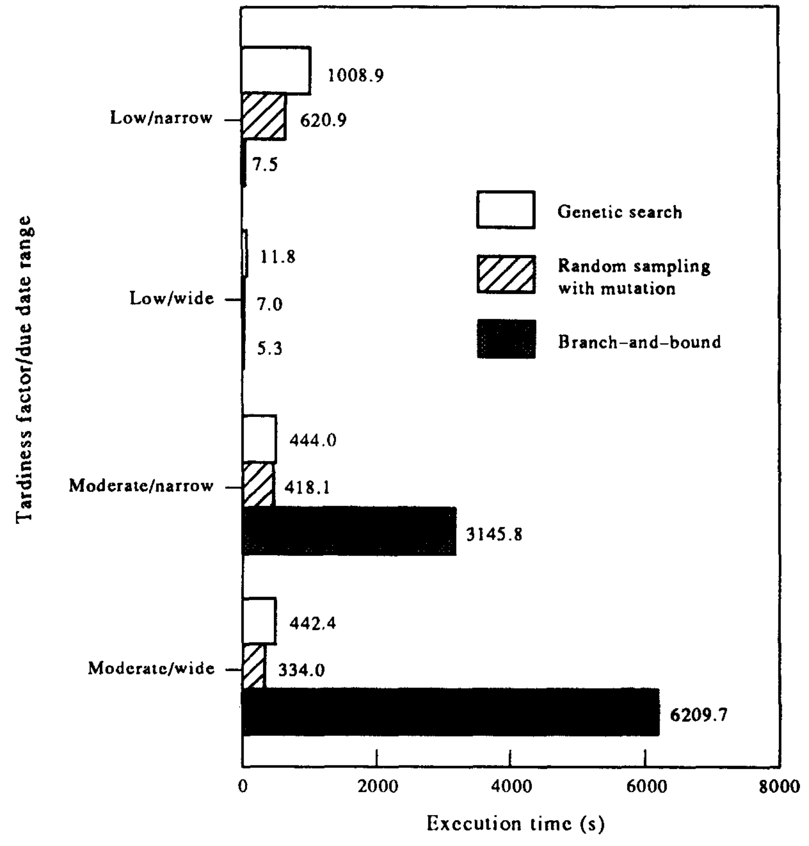

Fig. 8. Execution time. 45 Job problems.

Table 3. Comparison of methods

Processing time variance	Tardiness factor	Due date range	Search beats B-&-B (quality and time)	Search beats B-&-B (time only)	B-&-B beats search
Low	Low	Narrow		35	15, 25, 45
		Wide		Trivial	
	Moderate	Narrow	35		15, 25, 45
		Wide	45	15, 35	
High	Low	Narrow	25		35, 45
		Wide		Trivial	
	Moderate	Narrow	45	35	15
		Wide	25, 35, 45		15

procedure. This could result in early pruning of portions of the enumeration tree, accelerating the branch-and-bound solution. Moreover, should the branch-and-bound algorithm reach a point where the tree contains a large number of live nodes that cannot be fathomed, and where the incumbent solution has not improved over a long stretch of iterations, it might be efficacious to interrupt branch-and-bound and run a long genetic search, hoping to obtain an improved solution that will allow fathoming of some nodes. At present, however, the looseness of the bounds available in the branch-and-bound procedure suggests that a good incumbent provided by genetic search might not significantly accelerate pruning of the enumeration tree.

Other directions for future research deal with the genetic search algorithm itself. More work is needed to measure its performance, particularly as the number of jobs increases. Other methods of reproduction and mutation need to be tested. Also, since genetic search is not an inherently finite algorithm, a bounding procedure is needed, to give the user a sense of how likely it is that additional generations will produce an improvement in the current solution.

REFERENCES

1. S. S. Panwalkar, R. A. Dudek and M. L. Smith, Sequencing research and the industrial scheduling problem. In *Symposium on the Theory of Scheduling and Its Applications* (Edited by S. E. Elmaghraby), pp. 29–38 (1973).
2. D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison–Wesley, Reading, Mass. (1989).
3. K. R. Baker, *Introduction to Sequencing and Scheduling*. Wiley, New York (1974).
4. J. Du and J.Y.-T. Leung, Minimizing total tardiness on one machine is NP-hard. *Math. Opns Res.* **15**, 483–495 (1990).
5. J. W. Barnes and L. K. Vanston, Scheduling jobs with linear delay penalties and sequence dependent setup costs. *Opns Res.* **29**, 146–160 (1981).
6. A. J. Hoffman and P. Wolfe, History. In *The Traveling Salesman Problem* (Edited by E. L. Lawler, J. K. Lenstra, A. H. G. Rinnoy Kan and D. B. Shmoys), pp. 1–15. Wiley, Chichester (1985).
7. E. Balas and P. Toth, Branch and bound methods. In *The Traveling Salesman Problem* (Edited by E. L. Lawler, J. K. Lenstra, A. H. G. Rinooy Kan and D. B. Shmoys), pp. 361–401. Wiley, Chichester (1985).
8. B. L. Golden and W. R. Stewart, Empirical analysis of heuristics. In *The Travelling Salesman Problem* (Edited by E. L. Lawler, J. K. Lenstra, A. H. G. Rinooy Kan and D. B. Shmoys), pp. 207–249. Wiley, Chichester (1985).
9. J. W. Gavett, Three heuristic rules for sequencing jobs to a single production facility. *Mgmt Sci.* **11**, B-166–176 (1965).
10. A. G. Lockett and A. P. Muhlemann, A scheduling problem involving sequence dependent changeover times. *Opns Res.* **20**, 895–902 (1972).
11. R. D. Haynes, C. A. Komar and J. Byrd, The effectiveness of three heuristic rules for job sequencing in a single production facility. *Mgmt Sci.* **19**, 575–580 (1973).
12. C. H. White and R. C. Wilson, Sequence dependent set-up times and job sequencing. *Int. J. Prod. Res.* **15**, 191–202 (1977).
13. T. Prabhakar, A production scheduling problem with sequencing considerations. *Mgmt Sci.* **21**, 34–42 (1974).
14. J. C. Picard and M. Queyranne, The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling. *Opns Res.* **26**, 86–110 (1978).
15. B. D. Corwin and A. O. Esogbue, Two machine flow shop scheduling problems with sequence dependent setup times: a dynamic programming approach. *Naval Res. Logistics Q.* **21**, 515–523 (1974).
16. B. N. Srikar and S. Ghosh, An optimal algorithm to minimize makespan in the sequence dependent flowshop. Working Paper, College of Administrative Sciences, The Ohio State University (1984).
17. B. N. Srikar and S. Ghosh, A MILP model for the *n*-job, *m*-stage flowshop with sequence dependent setup times. *Int. J. Prod. Res.* **24**, 1459–1474 (1986).
18. J. N. D. Gupta and W. P. Darrow, The two-machine sequence dependent flowshop scheduling problem. *Eur. J. Opl Res.* **24**, 439–446 (1986).
19. S. K. Gupta, *N* jobs and *M* machines job-shop problems with sequence dependent set-up times. *Int. J. Prod. Res.* **20**, 643–656 (1982).
20. C. R. Glassey, Minimum changeover scheduling of several products on one machine. *Opns Res.* **16**, 342–352 (1968).
21. W. C. Driscoll and H. Emmons, Scheduling production on one machine with changeover costs. *AIIE Trans.* **9**, 388–395 (1977).

22. E. Uskup and S. B. Smith, A branch-and-bound algorithm for two-stage production sequencing problems. *Opns Res.* **23**, 118–136 (1975).
23. G. L. Ragatz, Scheduling to minimize tardiness on a single machine with sequence dependent setup times. Working Paper, Department of Management, Michigan State University (1989).
24. D. E. Goldberg and R. Lingle Jr, Alleles, loci, and the traveling salesman problem. *Proc. Int. Conf. Genetic Algorithms and their Applications*, pp. 154–159 (1985).
25. F. Glover and H. J. Greenberg, New approaches for heuristic search: a bilateral linkage with artificial intelligence. *Eur. J. Opl Res.* **39**, 119–130 (1989).
26. J. Grefenstette, R. Gopal, B. Rosmaita and D. Van Gucht, Genetic algorithms for the traveling salesman problem. *Proc. Int. Conf. Genetic Algorithms and their Applications*, pp. 160–168 (1985).
27. I. M. Oliver, D. J. Smith and J. R. C. Holland, A study of permutation crossover operators on the traveling salesman problem. *Proc. Second Int. Conf. Genetic Algorithms and their Applications*, pp. 224–230 (1987).
28. L. Davis, Job shop scheduling with genetic algorithms. *Proc. Int. Conf. Genetic Algorithms and their Applications*, pp. 136–140 (1985).
29. K. A. De Jong, An analysis of the behavior of a class of genetic adaptive systems. Unpublished doctoral dissertation, University of Michigan (1975).
30. A. H. G. Rinooy Kan, B. J. Lageweg and J. K. Lenstra, Minimizing total costs in one-machine scheduling. *Ops Res.* **23**, 908–927 (1975).
31. M. Bellmore and G. L. Nemhauser The travelling salesman problem: a survey. *Opns Res.* **16**, 538 (1968).