Available online at www.sciencedirect.com

ScienceDirect

Int. J. Production Economics 106 (2007) 574–584

international journal of production economics

www.elsevier.com/locate/ijpe

Multiobjective single machine scheduling with nontraditional requirements

Uday Haral, Rew-Win Chen, William G. Ferrell Jr*, Mary Beth Kurz

Department of Industrial Engineering, Clemson University, Clemson, 110 Freeman Hall, Box 340920, Clemson, SC 29634-0920, USA

Received 19 April 2004; accepted 28 June 2006
Available online 2 October 2006

Abstract

Many scheduling problems encountered in practice are must address requirements that are not found in the literature like maximizing the number of jobs that have a particular color. These nontraditional requirements are sometimes an objective as when the desirability of a schedule increases with the number of jobs of the same color that are scheduled consecutively. Other times, the requirements take the form of constraints as in cases where it is forbidden to have more than two consecutive jobs with a particular color. To complicate the situation, most real scheduling problems are multiobjective. This research centers on bicriteria scheduling with nontraditional requirements using an experimental approach and a Random Keys Genetic Algorithm to find Pareto optimal solutions. We address both traditional and nontraditional requirements in a single machine job shop with 20–50 jobs.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Multicriteria scheduling; Nontraditional requirements

1. Introduction

Scheduling is truly unique topic because it is both a task performed in virtually every industry multiple times each day and the basis for academic research covering more than half a century. This research focuses on using an evolutionary algorithm to find good schedules for a single machine job shop with more than one objective and also considering nontraditional requirements, that is, objectives and constraints that are found in practice but not in the literature. These requirements can involve continuous variables as with time-based measures; however, our focus will be on nontraditional requirements involving attributes like color and a product feature.

To illustrate the type of problem upon which the research if focused, consider the common situation in which the master production schedule is sent to the shop floor for implementation. In practice, this schedule is almost always modified, sometimes dramatically, to accommodate a number of practical issues that are not considered by the central planning system. For example, assembling an automobile for the international market requires building cars with both right and left hand drive. A very important feature of a good schedule is to cluster as many of each drive side together as possible so parts are staged from one side for as long as possible. This is one example of a nontraditional requirement that is

*Corresponding author. Tel.: +1 864 656 2724; fax: +1 864 656 0795.
E-mail address: fwillia@ces.clemson.edu (W.G. Ferrell Jr).

0925-5273/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.ijpe.2006.06.018

an objective. On the other hand, a nontraditional constraint in this example might be that efficient product demands that at least n orders for cars with drive on the same side must be scheduled consecutively or the schedule is infeasible.

Our experience as well as the literature (e.g., Kondakci and Bekiroglu, 1997; Sarin and Hariharan, 2000) suggests that schedulers in industry must consider more than one objective; hence, this research will include multiple objectives. This research, then, explores a single machine job shop in which the schedule may include traditional and nontraditional requirements. As in the traditional single machine job shop problem structure, it is assumed that n jobs are to be scheduled and their properties are known with certainty before the schedule is created. As is common in this type of research, an experimental approach is used that creates approximate solutions using an algorithm or heuristic. Genetic algorithms (GA) are frequently employed and their performance can be made greatly enhanced by carefully tuning the parameters as illustrated by Pongcharoena et al. (2002). Another approach is to marry an evolutionary algorithm with another search procedure with one example found in Dagli and Sittisathanchai (1995). Here, a Random Keys Genetic Algorithm (RKGA) is utilized to generate approximate Pareto optimal solutions for a series of scenario-based problems. The methodology utilizes the weighting method to create a single fitness function and includes the nontraditional requirements via a technique inspired by penalty functions.

2. Overview of the literature

2.1. Multiobjective optimization

Consider a program with k ($k \geqslant 2$) conflicting objective functions ($f_i : \Re^n \rightarrow \Re$) that are to be minimized simultaneously. That is, we wish to find a solution, x, from the set of feasible solutions, X, that solves the problem $\text{Min}_{x\in X}\, f(x) = \{f_1(x), f_2(x), \ldots, f_k(x)\}$. Since it is assumed that the objectives conflict, there is no single value of x that minimizes all objectives simultaneously in the single objective sense, so "optimal" must be defined differently.

While optimal has been defined in a number of ways over the years for this problem, in this research we use the following well-known ideas. A decision vector $x^* \in X$ is efficient or Pareto optimal if there does not exist an $x \in X$, $x \neq x^*$ such that $f_i(x) \leqslant f_i(x^*)$ for $i = 1, \ldots, k$ with strict inequality holding for at least one index i. The objective function value corresponding to this point, $f(x^*)$, is referred to as nondominated. Further, a decision vector $x^* \in X$ is weakly efficient or weakly Pareto optimal if there does not exist a $x \in X$, $x \neq x^*$ such that $f_i(x) < f_i(x^*)$ for $i = 1, \ldots, k$ and the corresponding objective function value is called weakly nondominated. Since problems typically have a number of solutions that satisfy these properties, the collection is called the Pareto frontier.

Using these definitions, a number of approaches have been developed to resolve multiobjective problems that are chronicled in many excellent references such as Miettinen (1998) and Collette and Siarry (2003). Some methodologies define a single measure and then seek the "best" solution relative to that measure. Other methodologies seek to generate all of the nondominated solutions and this is the approach taken here. Now these methods deal with the multiobjective problem, in some sense, by reducing it to a single objective programming problem using certain parameters and solving the single objective problem. The key concern is how to appropriately and systematically set or vary parameters to generate the whole nondominated set. Since we make no research contribution in this area, the following discussion is restricted to only those concept used in this research. Readers interested in additional information are directed to the references listed previously.

This research uses a modification of the $L_{p=0}$ metric that forms a linear combination of the k objectives using weights, λ_i so that the problem becomes $\text{Min}_{x\in X}\; z = \sum_{i=1}^{k} \lambda_i [f_i(x)]$. As with all L_p metrics, each λ_i is a scalar and $\sum_{i=1}^{k} \lambda_i = 1$. It is also relatively easy to solve the multiobjective optimization problem using this approach because basic strategies associated with single objectives apply. It should be noted, however, that it is impossible to guarantee that the entire set of nondominated solutions will be found with this approach (Eddy and Lewis, 2001) regardless of many values of the weights are selected.

2.2. Multiobjective scheduling

There is an ever-growing body of research on multiobjective scheduling and it would be impossible to pay proper credit to all contributors. Hence, we focus on a few references that are representative of work in the field and most directly applicable to this research.

Historically, the bicriteria scheduling problem has been particularly important. In 1988, Dileepan and Sen suggested that research at that time tended to use one of two approaches to these problems, namely, consider one of the criteria as the objective and the other as a constraint or combined the criteria into a single objective. Methods to solve the bicriteria problem have taken many forms including the use of an assignment model (Chen and Bulfin, 1989), branch and bound (Sen and Gupta, 1983), and dynamic programming (Cai and Lee, 2000), to name a few.

There have also been several literature reviews over the years. Fry et al. (1989) not only review the literature but suggest a classification scheme for multiobjective scheduling problems while Nagar et al. (1995) provide a good summary of the research related to regular measures up to early 1990s. Raghavachari (1988) provided a survey for scheduling problems having nonregular performance measures. The survey by T'Kindt et al. (2001) is an excellent paper that gives a technically insightful look by suggesting that the research can be placed in one of three categories: one-machine job shops, parallel machine job shops, and flow shops. Their extensive bibliography of more than 100 references contains at least half that relate to the bicriteria problem for a single machine. Regardless of the exact nature of the objectives, these studies offer a wide array of heuristics, branch-and-bound algorithms and other algorithms, MIP formulations, and dynamic programming models attempting to find the set of either strictly or weakly efficient solutions, or subsets of both. One particularly significant result is that the bicriteria problem is solvable in polynomial time if the formulation involves minimizing the sum of total completion time and maximum cost (Hoogeveen and van de Velde, 2001).

It is noted that the general scheduling problem is Nondeterministic Polynomial-time Complete (NPC) which means that all known algorithms that define an optimal solution require exponentially increasing computational time as the problem size increases; therefore, heuristic methods which provide approximate solutions are justified and are required when a practical situation can be modeled. Approximation algorithms like GA, tabu search, and simulated annealing are extensively used to find approximate solutions to such problems (Pirlot, 1999) although the combinatorial nature has been reported as a reason local search techniques are susceptible to becoming stuck in local optima (Ponnambalam et al., 2001). Khoo et al. (2000) concluded that since many reasonably sized scheduling problems can be resolved using GA, it should also work on the multiobjective problems with the weighted objective approach. Finally, Koppuraviuri (2000) investigated the effectiveness of using a GA to schedule flowshops with multiple nontraditional objectives and found the general approach feasible although the experiment work was very limited in scope.

2.3. Random-key genetic algorithm

GA's were introduced by Holland (1975) as a methodology for finding approximate solutions to complex problems by mirroring aspects of biological evolution. The basic framework he proposed involved representing solutions to a problem as "chromosomes" and generating future generations by "reproduction." Chromosomes are chosen to reproduce randomly and experience changes randomly, as organisms do in the natural environment. The probability that a particular chromosome is selected for reproduction is based on how well it solves the problem in a process analogous to survival of the fittest. The process of creating generations continues until some termination condition is reached, at which point the best chromosome is chosen as the solution.

The chromosomal representation of a solution is an important design feature of a GA. Chromosomes are generally strings of numbers that represent the solution to the problem or can be decoded to represent the solution. Sometimes the individual numbers are 0s and 1s but other possibilities exist like strings of non-negative integers. While many types of evolutionary operators have been introduced into the GA methodology based on their biological counterpart (e.g., mutation), a key operator is reproduction, and this particular operator is an important problem when applying GAs to sequence dependent problems like scheduling. For example, the simplest form of reproduction is to select two chromosomes from a generation, randomly select a point to split each into two pieces, and splice the front end of the first chromosome with the complementary end from the other and vice versa, to form two different chromosomes, each of correct length. For a scheduling problem, one possible chromosomal representation of the solution is simply an n-dimensional vector with the numbers 1 through n, and the solution would be to traverse the cities in the order presented in the chromosome. The infeasibility problem is easily seen in the simple example of crossover illustrated in Fig. 1.

Consider two feasible schedules, $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8$ and $4 \rightarrow 3 \rightarrow 2 \rightarrow 1 \rightarrow 7 \rightarrow 8 \rightarrow 5 \rightarrow 6$ in the figure that are selected at random from the initial population. Each tour is split at a randomly selected point, after the second job in this example, and the ends are spliced together. The resulting schedules are, in general, likely infeasible since each new chromosome most often has several jobs scheduled twice and others not visited at all. In the example, the first schedule resulting from this operator, $1 \rightarrow 2 \rightarrow 2 \rightarrow 1 \rightarrow 7 \rightarrow 8 \rightarrow 5 \rightarrow 6$, has a schedule that contains jobs 1 and 2 twice while omitting jobs 3 and 4. This difficulty of maintaining feasibility when applying GA's to sequencing problems like scheduling has been addressed in a number of ways. Some have proposed "repair" algorithms to recreate feasible schedule after the genetic operator is applied; however, the "repair" algorithm can consume a considerable amount of time and can inhibit convergence (Michalewicz, 2000). A better alternative is to use an alternative chromosomal representation like the one introduced by Bean (1994) in which a random numbers encoding structure is used, resulting in the so-called Random Keys Genetic Algorithm (RKGA). The structure proposed by Norman and Bean (1999) for a multiple machine scheduling problem assigned a real number to each job. The part of the number to the left of the decimal was used to assign the machine and the part to the right of the decimal (e.g., the "fraction") was used to assign the job sequence. A simple construct using this structure for our example in Fig. 1 would be to let the number in

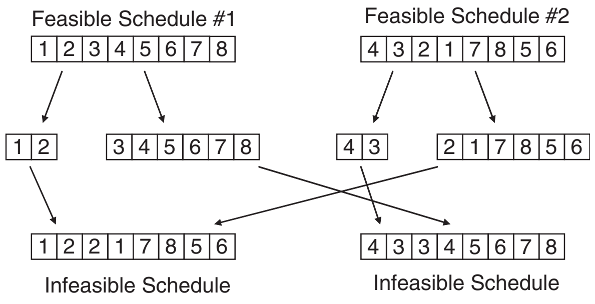

Fig. 1. Illustrative example of schedule infeasibility during traditional GA crossover.

the left-most slot correspond to the location in the schedule of job 1. The next slot to the right is the location for job 2, etc. Using this approach, the infeasible schedule at the right of Fig. 1 can be made feasible simply by how it is decoded. That is, the new decoding with now be $1 \rightarrow 4 \rightarrow 2 \rightarrow 3 \rightarrow 7 \rightarrow 8 \rightarrow 5 \rightarrow 6$ if we break ties left to right. Hence, the RKGA structure allows the proven evolutionary operators in GA's to be used on sequencing type problems like scheduling and maintain feasibility.

The mutation operator is important to GA's because it helps avoid becoming trapped in local optima. Implementing mutation in a tradition fashion would create an infeasibility problem similar to the one described for crossover; however, pairwise interchange is a technique that has been successfully used in scheduling metaheuristics and is our choice in this research (Croce, 1995). This technique operates by randomly selecting jobs i and j from the schedule and interchanging their positions. Croce (1995) describes the four different types of pairwise interchange techniques: (1) Adjacent Pairwise Interchange, (2) Non-Adjacent Pairwise Interchange, (3) Extraction and Forward Shifted Reinsertion, and (4) Extraction and Backward Shifted Reinsertion. We did not find any recommendation in the literature regarding one of these being preferred to the others, so this research uses adjacent pairwise interchange.

To summarize, this research has adapted the basic RKGA structure used by Bean (1994) to this scheduling application. Each generation contains of 20 chromosomes (schedules) and evolution proceeds using the crossover mechanism discussed earlier with adjacent pairwise interchange.

3. Research approach

To address the initial objective of investigating the impact of including realistic nontraditional requirements in developing a schedule for a single machine job shop, resolving a multiobjective scheduling problem is required. We have shown that the literature suggests RKGA could be an effective tool for generating Pareto optimal solutions and have elected to use an experimental research approach to pursue this possibility. The first task is to show that RKGA can, in fact, be used to generate point on the Pareto frontier, which is accomplished using two traditional objectives. Then, settings of the RKGA parameters are determined which yield good approximate solutions in a reasonable amount of time and, finally, this platform is adopted to experimentally investigate the inclusion of nontraditional requirements.

A classic single machine job shop has been assumed with all jobs and their attributes—traditional and nontraditional—known in advance. It is assumed that the schedule has more than one objective and might include some constraints related to the job attributes. The multiple objectives are converted to a single objective using the weighted sum approach and Pareto optimal solutions are subsequently found using RKGA. This type of experimental approach is widely used because the general scheduling problem is NPC; however, the significant drawback is that one must be very careful in generalizing any results and suggesting extensions to other scenarios.

3.1. Objective and constraints

This research utilizes combinations of two traditional and two nontraditional requirements in single machine job shop scheduling with n jobs. The first traditional element is the objective to minimize total flow time. The flow time of job j, F_j, is defined as the difference between the completion time of the job, C_j, and the earliest possible start time. Since all jobs are available when the schedule is constructed, all can start at time zero and

$$F = \sum_{j=1}^{n} F_j = \sum_{j=1}^{n} C_j.$$

The second traditional element is also an objective, minimizing maximum tardiness. The tardiness of job j, L_j, is defined as the difference between completion time, C_j, and due date, d_j, if the job is late. Hence,

$$L_j = \max(C_j - d_j, 0)$$

and the maximum tardiness, L, is

$$L = \max_j [L_j] = \max_j [\max(C_j - d_j, 0)].$$

Note that the optimal solution for each of these objectives taken separately is well known when scheduling a single machine job shop. Ordering jobs using the shortest processing time first rule minimizes the total flowtime while ordering the jobs using the earliest due date first rule minimizes the maximum tardiness. These facts are used during the experimentation as benchmarks for the RKGA and to provide a theoretically known point on the Pareto frontier when either of these objectives is considered.

There are a number of nontraditional objectives and constraints that are encountered in practice but not addressed in the scheduling literature. For example, consider an assembly line that produces a single style car but with either left or right hand drive. It would be highly desirable to schedule as many automobiles as possible with the steering on one side before switching so that the line can be fed from one side for as long as possible. In this context, this is a nontraditional objective. On the other hand, consider a production line making a component in different colors that must be visually inspected by a human. It is well known that the effectiveness of the inspector is enhanced if the colors are routinely different, so a schedule in this situation might require that no more than a certain number of the same color components be schedules consecutively. This is an example of a nontraditional constraint. In our experience, schedules in real situation are influenced as much by these nontraditional objectives and constraints as they are by traditional measures like flow time and lateness.

In this research, we try to capture one nontraditional objective and one nontraditional constraint. To illustrate these, let each job to be scheduled possess one of three attributes: a, b, or c. The nontraditional objective is called maximum clustering and seeks to keep as many jobs with attribute a together a possible. The nontraditional constraint is called minimum spacing and requires that at least n jobs with attribute c be scheduled between two jobs with attribute b.

4. Experiments and results

4.1. RKGA feasibility

To explore the feasibility of using RKGA in the setting of bicriteria scheduling, a base case was established using the two traditional objectives. This scenario is found in the literature so a qualitative comparison could be made regarding the Pareto Frontier. Since the two objectives are minimize mean flowtime and minimize maximum tardiness, the weighted objective is

$$\text{Min } z = \lambda F + (1-\lambda)L$$
$$= \lambda\left(\sum_{j=1}^{n} C_j\right) + (1-\lambda)\left\{\max_j[\max(C_j - d_j, 0)]\right\},$$

where $0 \leqslant \lambda \leqslant 1$.

To empirically investigate the convergence of the RKGA, a 50 job problem was used and λ was set to 0.05, 0.5, and 0.95. The fitness value of the best schedule and the average of all schedules were recorded. All three values of the weights yielded similar results exemplified by Fig. 2 that reflects $\lambda = 0.5$.

The basic structure found in Fig. 2 was seen in all experiments. There are substantial improvements during nearly all generations at the beginning and then a period of no improvement in the fitness function. By generation 4000, the GA had found a reasonably good solution in all test cases after which time there is a relatively long period of no improvement. When the algorithm is extended to 15,000 generations, some of the scenarios revealed that a local optimum had been found and a further improvement was possible as illustrated by the example in the figure. In other cases, this was not seen and the solutions found after 4000 generations remained the best for the remaining generations.

Before finalizing the number of generations to use in the experimental study, we thought it important to compare these results with the alternate approach of replicating each experiment several times for 5000 generations but using a different starting point with each. The result was that this approach found a solution that was at least as good as the longer generation runs in all cases and, in some, the solution was better. For example, when the base case scenario was tested with $\lambda = 0.85$, replicating five replication using 5000 generations each yielded a best weighted fitness value of 212,711 while running the RKGA for 25,000 iteration produced 218,686.

Based on these experimental observations, it was decided that each of the experiments would be replicated five times with each replication consisting of 5000 generations. Although this has not been proven, our experience with these problems lead us to speculate that larger scheduling problems with randomly generated data appear to have a number of solutions that represent local minima with objective function values quite close. For example, consider a schedule involving 50 jobs that has several similar jobs located rather close together in a near optimal sequence. Switching the two would have a small impact on the objective function (i.e., the fitness function in the RKGA); hence, if the RKGA initially converges to one that is not globally optimal, it has a difficult time finding others that are slightly better.

4.2. Experimentation with the traditional and nontraditional requirements

Three different combinations of traditional and nontraditional requirements are investigated, namely: (1) two traditional objectives, (2) one traditional and one nontraditional objective, (3) one nontraditional objective and one nontraditional constraint. Each of these scenarios is investigated with environments that include 20, 30, and 50 jobs.

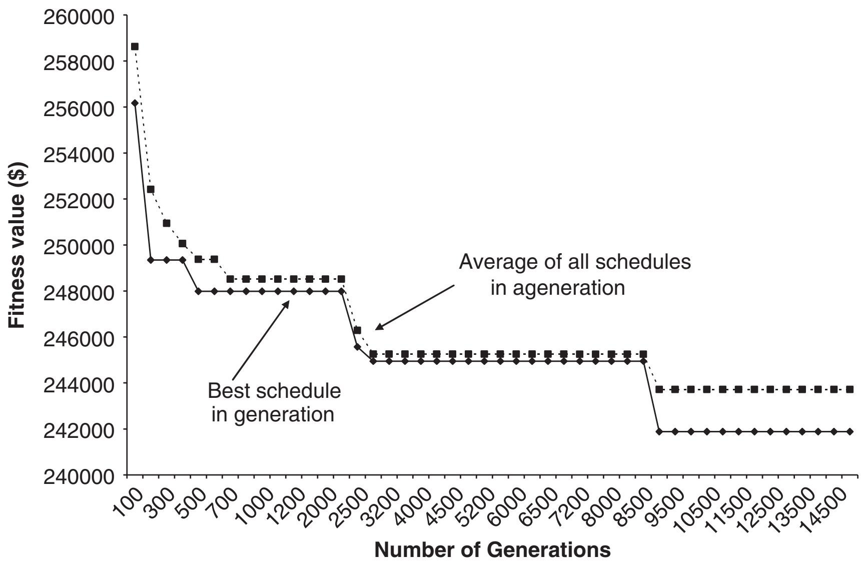

Fig. 2. Fitness values for the average and best schedule in a generation.

To quantify nontraditional requirements, a penalty function approach is used that are analogous to those found in algorithms targeted at constrained nonlinear optimization problems. In the case of nontraditional objectives, a progressive penalty is used to reflect the degree to which schedules do not meet the objective. For nontraditional constraints, a severe penalty is imposed if the constraint is not met while a zero penalty is assigned if it is met. These penalty values are included in the fitness function so that schedules that infeasible objectives or less desirable objectives are selected less often in the subsequent generation. For example, a function like the one illustrated in Fig. 3 was used to model maximum clustering. In this example, there were 30 jobs with attribute a and 20 with other attributes. Using the function illustrated in Fig. 3, if all 30 jobs were consecutive in the schedule there would be no penalty; however, if only 20 were consecutive the penalty would be \$3333. The nontraditional constraints were modeled similarly but with a step function that had a value of 0 if the constraint is satisfied and a very large number if not.

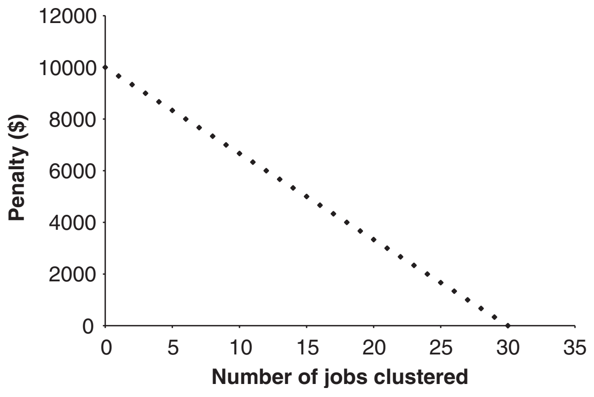

Fig. 3. Structure of penalty function used with nontraditional requirements.

4.2.1. Two traditional objectives

The Pareto frontier for the 20 job problem with the criteria of minimizing maximum tardiness and minimizing total flowtime is illustrated in Fig. 4. This figure illustrates the Pareto frontier that is generated by varying the weights between 0 and 1 and the general shape is consistent with similar research found in the literature. Although it is impossible to see on the graph, several of the points were obtained for more than one value of λ; that is, some of these solutions were either optimal for several different weights or there was a local minimum that the algorithm could not escape. Notice that with the small number of jobs, only four unique levels of tardiness were observed in the solutions found by the RKGA. The cluster at 4490 time units of maximum tardiness was obtained for values of λ between 0.95 and 0.8 while cluster at 4364 were found for λ between 0.8 and 0.5. The final points were determined for the other values of λ.

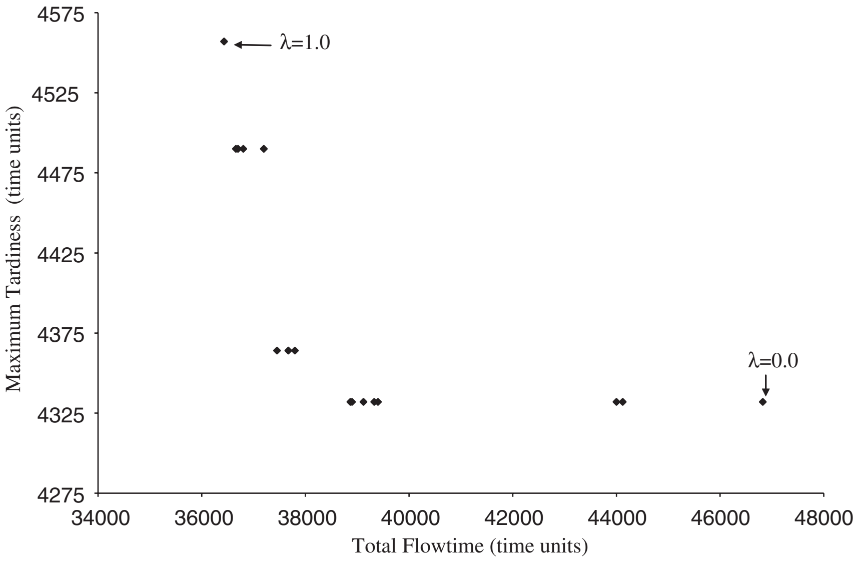

Fig. 4. Pareto frontier for two traditional objectives, 20 jobs.

To gain some insight into the nature of the solutions, Table 1 shows the schedules that correspond to these points on this Pareto frontier for the 20 job problem. The solutions are identified by the value of λ used to generate them as indicated in the top row. Notice how certain jobs tend to always be at the first of the schedule like 2, 10, 14, and 18 while others like 1 and 3 are always at the bottom. As the value of each objective function become more or less important, there seem to be clusters of jobs in the middle that gradually rearrange to produce the various solutions. One observation here is that for these objectives and this example, there appear to be some schedules that are reasonably robust to the varying weights. While this is clearly a function of the particular parameters of the specific problem, it would be a very important result were it to occur in practice.

Table 1
Schedules for selected values of λ, two traditional objectives, 20 jobs

Weights (λ)													
1	0.9	0.8	0.7	0.6	0.5	0.5	0.4	0.4	0.3	0.3	0.2	0.1	0
2	14	2	2	2	2	2	2	2	2	2	2	2	11
10	10	10	18	10	10	10	10	10	10	18	14	9	5
18	12	18	14	14	18	18	14	14	14	10	10	10	15
14	5	14	17	18	14	14	18	17	18	14	12	14	19
17	2	17	12	17	17	17	17	9	17	17	15	18	2
12	9	12	20	12	20	12	12	19	12	12	19	17	14
20	15	20	9	9	9	20	15	18	16	9	5	4	18
9	17	15	10	15	4	9	19	12	20	5	18	12	17
15	18	19	4	20	12	4	20	15	9	19	17	19	3
19	20	9	13	19	19	11	9	5	19	4	20	5	16
5	19	5	11	4	5	5	5	20	5	11	9	20	20
4	13	13	5	6	15	15	7	13	15	20	13	13	9
13	4	4	15	7	13	19	8	7	13	15	4	7	10
7	7	7	19	16	7	3	13	11	4	16	7	16	8
11	16	11	6	11	16	7	4	16	7	8	16	11	13
16	11	6	7	5	11	16	16	6	11	13	11	15	4
6	6	1	16	8	6	8	11	8	6	6	6	6	6
8	1	8	1	1	8	13	6	3	8	7	8	3	12
1	3	3	3	3	3	6	3	4	3	3	3	8	7
3	8	16	8	13	1	1	1	1	1	1	1	1	1

Fig. 5 shows the Pareto frontier for the 50 job problem. Notice the clustering along the discrete values of tardiness has vanished, a fact that we believe is due to the increase in the number of possible schedules and, hence, more possible values of tardiness. Although the solutions for these Pareto points are not listed to conserve space, this conjecture is supported by the fact that the pattern of certain jobs being located in certain general areas of the schedule is not seen as it is in Table 1. For reference, these experiments were executed on a Sun Fire UNIX machine with two 900 MHz processors and 8 GB of RAM. On average, each replication of the experiments including 20, 30, and 50 jobs took about 150, 180, and 900 s, respectively.

4.2.2. *One traditional and one nontraditional objective*

We now turn attention to nontraditional requirements. This first case explores one traditional objective (minimize total flowtime) and one nontraditional

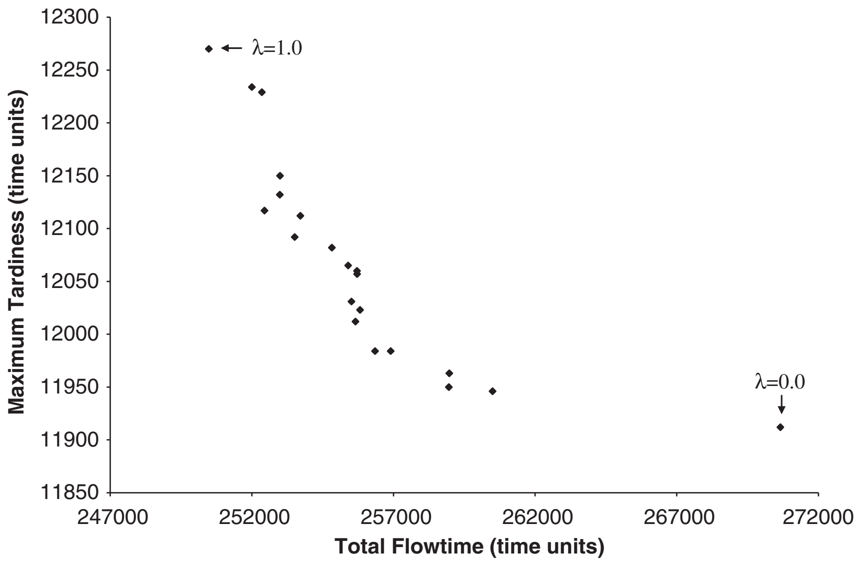

Fig. 5. Pareto frontier for two traditional objectives, 50 jobs.

objective (minimize clustering of an attribute). While the problems with 20 and 30 jobs revealed a Pareto frontier similar to the two traditional objectives, the 50 job case appeared a bit different as illustrated in Fig. 6. Notice that the points tend to lie on near parallel lines with respect to the maximum clustering objective.

Recall that this objective is based on the clustering of an attribute with three attributes; hence, it appears that this discrete nature creates preferred levels for this objective and the RKGA tends to converge to local minimum of the traditional objective on the levels of the nontraditional one. As noted before, we suspect that by randomly assigning the job data, the clusters might actually reflect one true minimum with two or three jobs switched to provide a small difference in objective function value but not enough to redirect the RKGA.

4.2.3. *One nontraditional objective and one nontraditional constraint*

We use the term nontraditional constraint to describe situations like those in nontraditional objectives but now the relationships are required; hence, they are now constraints. The example used here centers on minimum spacing in which a certain number of jobs with a particular attribute are required to be between two jobs with another attribute. This situation has been captured using a penalty function but now, rather than a smooth function reflecting a degree of discontent as in Fig. 3, we use a step function that applies a severe penalty to any schedule that does not meet the requirement and zero penalty to schedules that have the minimum spacing required.

To test this approach, the fitness function in the RKGA is modified to be the sum of a penalty function similar to Fig. 3 to model the maximum clustering objective plus the step function to model the nontraditional constraint. The magnitude of the penalty associated with not meeting the constraint was on the order of twice the maximum penalty associated with the nontraditional objective. Scenarios involving 20, 30, and 50 jobs were resolved and, in all cases, the schedules met the minimum spacing constraint. Hence, we conclude that the step penalty function is an effective means of modeling a nontraditional constraint.

One extension from our experience in practice was to see if this approach could be used for "soft" constraints; that is, a constraint that could be violated if the payoff was sufficiently large. To explore this extension, we treated the nontraditional constraint as a second objective and varied the weights as we did in the two objective scenarios. The nontraditional constraint was again modeled as a step function with each violation incurring maximum penalty. Weights, the sum of which are one, were associated with each and subsequently varied as before. Our intent idea was that when the weight

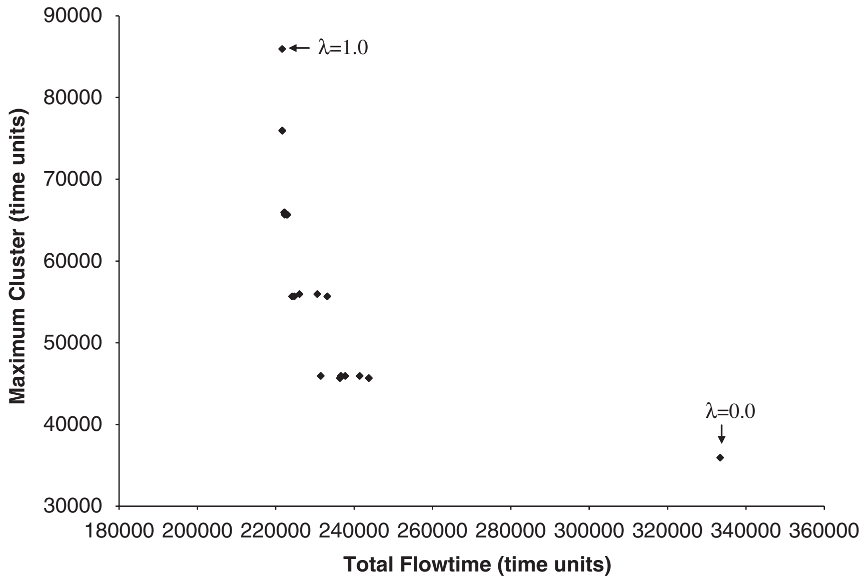

Fig. 6. Pareto frontier for one traditional and one nontraditional objective, 50 jobs.

Table 2
Objective function values for one nontraditional objective and one nontraditional constraint

Weights (λ)	Maximum cluster	Minimum spacing	Fitness value
1	17297	7000	17297
0.9	17567	4000	16210
0.8	17297	4000	14637
0.75	17297	4000	13972
0.7	17297	3000	13008
0.65	17297	2000	11943
0.6	17297	0	10378
0.5	19459	1000	10229
0.4	17297	1000	7518
0.3	17297	0	5189
0.2	18108	0	3621
0.1	27567	0	2756
0	57297	0	0

associated with the nontraditional constraint is reduced, the constraint become soft. At some point, we expected to see the constraint being violated as the objective function became more important and this is exactly what happened. The objective function values for 30 job problem are given in Table 2.

When the weight assigned to minimum spacing is zero ($\lambda = 1$) the constraint is violated frequently as expected; however, as the weight on minimum spacing is increased, the constraint is satisfied more often until it is satisfies completely for all weights beyond a certain point. This result suggests that this approach to modeling soft constraints in the RKGA is consistent with intuition.

5. Conclusions

In this research, we have documented a methodology by which nontraditional requirements frequently found in practice and based on job attributes can be included in quantitative approaches to finding schedules for single machine job shops. This methodology is based on penalty functions and includes the ability to accommodate the nontraditional requirements as part of the objective function, as a hard constraint that must be met for a schedule to be considered feasible, or as a soft constraint that might be violated if the benefit is sufficiently large. In addition, it was experimentally shown that RKGA is an effective technique for finding approximate Pareto optimal solutions when $L_{P=0}$ metric is used. In particular, good solutions are found by the 15,000th generation with a population size of 20; however, we found that running more replications of 5000 generations each where each replication had a randomly selected initial population was even more effective in finding the best solution for a scenario. This experimentally based knowledge was then used to investigate three scenarios: (1) two traditional objectives, (2) one traditional and one nontraditional objective, and (3) one nontraditional objective and one nontraditional constraint. Each of these scenarios was imposed on single machine job shops with 20, 30, and 50 jobs. The results suggested that using the RKGA and the $L_{P=0}$ measure effectively finds good solutions across all scenarios. Several interesting nuances were noted such as the tendency, in certain situations, for segments of the schedule to remain essentially unchanged for over a range of weights. We hypothesize that this robustness could be valuable in practice because it suggests a decision maker might not be required to precisely define the relative weights between the objectives; rather, solving the problem for a "best guess" will yield a solution that will have segments of the schedule that are optimal over some range of weights.

References

Bean, J., 1994. Genetic algorithms and random keys for sequencing and optimization. ORSA Journal on Computing 6, 154–160.

Cai, X., Lee, C., 2000. Multiprocessor task scheduling to minimize maximum tardiness and the total completion time. IEEE Transactions on Robotics and Automation 16, 824–830.

Chen, C., Bulfin, R., 1989. Scheduling unit processing times jobs on a single machine with multiple constraints. Computers and Operations Research 17, 1–7.

Collette, Y., Siarry, P., 2003. Multiobjective Optimization: Principles and Case Studies. Springer, Berlin.

Croce, F., 1995. Generalized pairwise interchanges and machine scheduling. European Journal of Operations Research 83, 310–319.

Dagli, C.H., Sittisathanchai, S., 1995. Genetic neuro-scheduler: A new approach for job shop scheduling. International Journal of Production Economics 41, 135–145.

Eddy, J., Lewis, K., 2001. Effective generations of Pareto sets using genetic programming. In: ASME design Engineering Technical Conferences DETC2001/DAC-21094.

Fry, T., Armstrong, R., Lewis, H., 1989. A framework for single machine multiple objective sequencing research. Omega 17 (6), 595–607.

Holland, J., 1975. Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor.

Hoogeveen, J., van de Velde, S., 2001. Scheduling with target start times. European Journal of Operational Research 129, 87–94.

Khoo, L., Lee, S., Yin, X., 2000. A prototype genetic algorithm-enhanced multiobjective scheduler for manufacturing systems. International Journal of Advanced Manufacturing Technology 16 (2), 131–138.

Kondakci, S.K., Bekiroglu, T., 1997. Scheduling with bicriteria: Total flowtime and number of tardy jobs. International Journal of Production Economics 53, 91–99.

Koppuraviuri, V., 2000. Dynamic scheduling of jobs with multiple attributes using genetic algorithms. Master's Thesis, Clemson University.

Michalewicz, Z., 2000. Evolutionary computation 1. In: Back, T., Fogel, D.B., Michalewicz, Z. (Eds.), Basic Algorithms and Operators. Philadelphia Institute of Physics Publishing, Bristol, pp. 56–61.

Miettinen, K., 1998. Nonlinear Multiobjective Optimization. Kluwer Academic Publishers, Boston.

Nagar, A., Haddock, J., Heragu, S., 1995. Multiple and bicriteria scheduling: A literature survey. European Journal of Operational Research 81, 88–104.

Norman, B., Bean, J., 1999. A genetic algorithm methodology for complex scheduling problems. Navel Research Logistics 46, 199–211.

Pirlot, M., 1999. General local search methods. European Journal of Operational Research 92, 493–511.

Pongcharoena, P., Hicksa, C., Braidena, P.M., Stewardsonb, D.J., 2002. Determining optimum Genetic Algorithm parameters for scheduling the manufacturing and assembly of complex product. International Journal of Production Economics 78, 311–322.

Ponnambalam, S., Ramkumar, V., Jawahar, N., 2001. A multi-objective genetic algorithm for job shop scheduling. Production Planning and Control 12 (8), 764–774.

Raghavachari, M., 1988. Scheduling problems with non-regular penalty functions—A review. Operations Research 25 (3), 144–164.

Sarin, S.C., Hariharan, R., 2000. A two machine bicriteria scheduling problem. International Journal of Production Economics 65, 125–139.

Sen, T., Gupta, S., 1983. A branch and bound approach to solve a bicriterion scheduling problem. IIE Transactions 15, 84–88.

T'Kindt, V., Billaut, J., Proust, C., 2001. Multicriteria scheduling problems: A survey. RAIRO Operations Research 35, 143–163.