IIE Transactions

ISSN: 0740-817X (Print) 1545-8830 (Online) Journal homepage: http://www.tandfonline.com/loi/uiie20

A problem space algorithm for single machine weighted tardiness problems

SELCUK AVCI , M. SELIM AKTURK & ROBERT H. STORER

To cite this article: SELCUK AVCI , M. SELIM AKTURK & ROBERT H. STORER (2003) A problem space algorithm for single machine weighted tardiness problems, IIE Transactions, 35:5, 479-486, DOI: 10.1080/07408170304390

To link to this article: http://dx.doi.org/10.1080/07408170304390

Published online: 29 Oct 2010.

Submit your article to this journal

Article views: 43

View related articles

Citing articles: 21 View citing articles

Full Terms & Conditions of access and use can be found at
http://www.tandfonline.com/action/journalInformation?journalCode=uiie20

Download by: [Bilkent University] **Date:** 12 November 2017, At: 23:39

IIE Transactions (2003) **35**, 479–486
0740-817X © 2003 "IIE"
0740-817X/03 $12.00 + .00
DOI: 10.1080/07408170390187924

A problem space algorithm for single machine weighted tardiness problems

SELCUK AVCI[1], M. SELIM AKTURK[2] and ROBERT H. STORER[1,*]

[1]*Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA 18015, USA*
E-mail: rhs2@lehigh.edu
[2]*Department of Industrial Engineering, Bilkent University, Ankara, Turkey*
E-mail: akturk@bilkent.edu.tr

Received March 2000 and accepted May 2002

Downloaded by [Bilkent University] at 23:39 12 November 2017

We propose a problem space genetic algorithm to solve single machine total weighted tardiness scheduling problems. The proposed algorithm utilizes global and time-dependent local dominance rules to improve the neighborhood structure of the search space. They are also a powerful exploitation (intensifying) tool since the global optimum is one of the local optimum solutions. Furthermore, the problem space search method significantly enhances the exploration (diversification) capability of the genetic algorithm. In summary, we can improve both solution quality and robustness over the other local search algorithms reported in the literature.

1. Introduction

In this paper we develop a Problem Space Genetic Algorithm (PSGA) for the single machine total weighted tardiness problem, $1||\sum w_jT_j$. The results are quite encouraging relative to the best algorithm in the literature (Crauwels *et al.*, 1998). In addition to developing an effective algorithm for an important problem, we also investigate some key questions regarding PSGA's and provide insight into their performance. PSGA's have been shown in previous research to be quite effective for a variety of scheduling problems (Storer *et al.*, 1992, 1995). Intuitive explanations have been offered to account for their good performance, but little evidence exists to substantiate these conjectures. In this paper we also provide insight into the behavior of PSGA's by investigating the effect of various base heuristics on PSGA performance. In this section we first review research on the single machine total weighted tardiness problem, followed by a review of PSGA.

1.1. *The single machine total weighted tardiness problem*

The single machine total weighted tardiness problem, $1||\sum w_jT_j$, may be stated as follows. A set of jobs (indexed $1,\ldots,n$) is to be processed without interruption on a single machine that can process one job at a time. All jobs become available for processing at time zero. Job j has an integer processing time p_j, a due date d_j, and a positive weight w_j. A weighted tardiness penalty is incurred for each time unit of tardiness T_j if job j is completed after its due date d_j. The problem can be formally stated as: find a schedule S that minimizes $f(S) = \sum_{j=1}^{n} w_jT_j$.

Lawler (1977) showed that the problem is strongly NP-hard. Emmons (1969) derived several dominance rules that restrict the search for an optimal solution to the unweighted problem. Rinnooy Kan *et al.* (1975) extended these results to the weighted tardiness problem. The Branch and Bound (BB) algorithm of Potts and Van Wassenhove (1985) can solve problems with up to 50 jobs. Akturk and Yildirim (1998) proposed a new dominance rule and a lower bounding scheme for the $1||\sum w_jT_j$ problem that can be used to reduce the time complexity of any exact approach.

Implicit enumerative algorithms for the total weighted tardiness problem, such as the BB algorithm proposed by Potts and Wassenhove (1985), guarantee optimality but require considerable computational resources in terms of both computation time and memory. It is important to note that due to the nature of the problem, the number of local minima is very large with respect to neighborhoods based on pairwise interchange. Therefore, several heuristics and dispatching rules have been proposed to generate good, but not necessarily optimal solutions as discussed in Potts and Van Wassenhove (1991). The

*Corresponding author

0740-817X © 2003 "IIE"

obvious disadvantage of these methods is that solutions generated by simple heuristic methods may be far from the optimum. Crauwels *et al*. (1998) present several local search heuristics for the $1||\sum w_jT_j$ problem. They introduce a new binary encoding scheme to represent solutions, together with a heuristic to decode the binary representations into actual sequences. This binary encoding scheme is also compared to the usual permutation representation for descent, simulated annealing, threshold accepting, tabu search and genetic algorithms on a large set of problems.

1.2. *Problem space genetic algorithms*

Problem Space Genetic Algorithms have been used successfully in the past on various scheduling problems (Storer *et al*., 1995). Embedded within a PSGA is a constructive base heuristic $H : \boldsymbol{P} \rightarrow S$ which maps a problem instance data vector $\boldsymbol{P}$ to a solution sequence S. Given any solution sequence S, the objective function $V(S)$ (total weighted tardiness) can be calculated. Let $\boldsymbol{\delta}$ be a vector of perturbations. One can then define an optimization problem on $\boldsymbol{\delta} \in \mathscr{R}^{\mathscr{N}}$, the space of perturbations:

$$\min_{\boldsymbol{\delta}} V[H(\boldsymbol{P}+\boldsymbol{\delta})].$$

A PSGA uses the perturbation vector as the encoding of a solution (or chromosome). A chromosome (perturbation vector) $\boldsymbol{\delta}$ is 'decoded' into a sequence by applying $H(\boldsymbol{P}+\boldsymbol{\delta})$, and its value obtained by applying $V[H(\boldsymbol{P}+\boldsymbol{\delta})]$. Unlike many applications of genetic algorithms to sequencing problems, standard crossover operators may be applied under this encoding.

The reason for the success of PSGA's has, in the past, been explained *intuitively* by observing that good solutions will tend to lie near the point $\boldsymbol{\delta} = \mathbf{0}$ in problem space. This makes sense since we expect the heuristic to provide reasonable solutions to the original problem when perturbations are small. Since the neighborhood around $\boldsymbol{\delta} = \mathbf{0}$ consists mainly of good solutions, searching this neighborhood yields good results. In this paper we provide empirical evidence of this observation by embedding various base heuristics within the PSGA.

In Section 2 we develop a set of PSGA's for the problem that differ from each other by the embedded base heuristic, and by the search algorithm employed. The reason for developing these different versions of the PSGA is to show the effect of the base heuristic on problem space neighborhood quality, and to provide insight into what makes the algorithm successful. In Section 3 we conduct tuning experiments, then describe the computational testing in Section 4. In Section 5 we present the results, including a comparison to the results of Crauwels *et al*. (1998) and a discussion of the insight gained about the PSGA.

2. Algorithm development

In this section we develop a set of PSGA's for the $1||\sum w_jT_j$ problem. Version one uses the simple Apparent Tardiness Cost (ATC) dispatching rule by Morton and Pentico (1993) as the base heuristic. The second version augments ATC with the Global Dominance (GD) rules of Rinnooy Kan *et al*. (1975). The final version of the base heuristic uses Local Dominance Rules (LDR) proposed by Akturk and Yildirim (1998) in addition to the global dominance rules. The properties of the dominance rules guarantee that the solution generated by version 2 is at least as good as that generated by version 1, and similarly that version 3 solutions dominate version 2. An algorithmic description of version 3 (ATC + GD + LDR) is discussed below. For version 1 (ATC), we skip Steps 2.1, 2.2, 2.3 and 2.8. For version 2 (ATC + GD), we skip Steps 2.1 and 2.8. We first present the algorithm then describe each of the steps.

Algorithm:

Step 0. (Problem reduction pre-processing): Apply the global dominance rules of Rinnooy Kan *et al*. (1975). Let β be the set of jobs that are assigned to the first positions due to this rule, B be the completion time of set β, and the position index $k = |\beta|$.

Step 1. (Problem space genetic algorithm): Create the initial population by randomly generating the perturbation vectors as chromosomes.

Step 2. (Base heuristic): For each individual perturbation vector $\boldsymbol{i}$ (or equivalently for every chromosome in the GA population), set the current time $t = B$, $k = k + 1$ and solve the following base heuristic.

Step 2.1. If $t > t_l$ then the remaining unscheduled jobs are sequenced using the SWPT rule, i.e., in non-increasing order of w_j/p_j, and go to Step 2.8.

Step 2.2. For all unscheduled jobs at time t, determine the set of eligible jobs using the global dominance rule.

Step 2.3. If only one job is eligible, i.e., job m, then schedule job m at position k, and set $t = t + p_m$ and $k = k + 1$, otherwise go to Step 2.4. If there are no other jobs remaining then go to Step 2.8, otherwise go to Step 2.1.

Step 2.4. For each eligible job j at time t, calculate the ATC priorities as follows:

$$a_j(t) = \frac{w_j}{p_j} \times \exp\big(-\max\,(0, d_j - t - p_j) / (l \times \bar{p})\big).$$

Step 2.5. The ATC priorities, $a_j(t)$, are normalized into the interval [0,1] yielding $n_j(t)$

Downloaded by [Bilkent University] at 23:39 12 November 2017

as follows. Let $a_{\min}(t) = \min_j a_j(t)$ and $a_{\max}(t) = \max_j a_j(t)$ then

$$n_j(t) = (a_j(t) - a_{\min}(t)) / (a_{\max}(t) - a_{\min}(t)).$$

Step 2.6. Perturb the job priorities as follows: $n_j(t) = n_j(t) + \delta_j$.

Step 2.7. Select the job j which has the highest perturbed normalized priority $n_j(t) + \delta_j$, and schedule it next in the sequence. Set $t = t + p_j$ and $k = k + 1$. If there are any unscheduled jobs go to Step 2.1.

Step 2.8. (Local dominance rule): Improve the sequence generated above for a given perturbation vector $\boldsymbol{i}$ by applying the Local Dominance Rule (LDR) based on the Adjacent Pairwise Interchange (API) method proposed by Akturk and Yildirim (1998). Calculate the total weighted tardiness for the given perturbation vector, denoted by $V(\boldsymbol{i})$.

Step 3. If the generation number is less than the limit continue, otherwise stop.

Step 4. Calculate a fitness $f(\boldsymbol{i})$ for each perturbation vector $\boldsymbol{i}$ of the current generation as follows:

$$\text{Let } V_{\max} = \max_{\boldsymbol{i}} V(\boldsymbol{i}), \text{ then}$$

$$f(\boldsymbol{i}) = (V_{\max} - V(\boldsymbol{i}))^{\pi} \Big/ \sum_i (V_{\max} - V(\boldsymbol{i}))^{\pi}.$$

Step 5. Perform 'evolutionary processes' to get the next generation using the fitness distribution and updated perturbation vectors. Go to Step 2.

In Step 0, we generate a $n \times n$ 0-1 global dominance matrix, in which an entry of one indicates that the job in row i globally dominates the job in column j due to the global dominance theorem by Rinnooy Kan *et al*. (1975). Whenever jobs i and j satisfy this theorem, an arc (i, j) is added to the precedence graph along with any other arcs that are implied by the transitive property. In this matrix, RowSum(i) and ColSum(i) give the number of jobs guaranteed to be succeeding or preceding job i, respectively, in an optimum sequence. Let N be the number of unscheduled jobs. If RowSum(i) $= N - 1$ then job i will be scheduled to the first available position. Similarly, if ColSum(j) $= N - 1$ then job j will be scheduled to the last available position. If we proceed iteratively in the same manner, we can fix certain jobs to the first and last positions. As a result, we can reduce the problem size for certain problem instances, and implement the local search on this reduced set of jobs. We also use this global dominance matrix to find a set of eligible jobs in Step 2.1. A job is called eligible if it is not globally dominated by some other unscheduled job at time t.

Potts and Van Wassenhove (1991) applied an API method starting with the heuristic sequence obtained by applying the ATC rule. The ATC rule is a composite dispatching rule that combines the Shortest Weighted Processing Time (SWPT) rule and the minimum slack rule. Under the ATC rule jobs are scheduled one at a time; that is, every time the machine becomes free, a priority index is computed for each remaining job j. The job with the highest priority index is then selected to be processed next. The priority index is a function of the time t at which the machine becomes free as well as p_j, w_j, and d_j of the remaining jobs. We set the look-ahead parameter $l = 2$ as suggested by Morton and Pentico (1993), and $\bar{p}$ is the average processing time of the remaining unscheduled jobs.

We use the global dominance theorem as a static dominance rule, and also employ a time-dependent local dominance rule proposed by Akturk and Yildirim (1998) in Step 2.8 to improve the initial sequence given by the ATC + GD rule. They show that the arrangement of adjacent jobs in an optimal schedule depends on their start times. For each pair of jobs i and j that are adjacent in an optimal schedule, there can be a critical value t_{ij} (denoted as breakpoint) such that i precedes j if processing of this pair starts earlier than t_{ij} and j precedes i if processing of this pair starts after t_{ij}. As a result, they state a general rule that provides a sufficient condition for schedules that cannot be improved by adjacent job interchanges. They show that if any sequence violates the proposed dominance rule, then switching these jobs either reduces the total weighted tardiness or leaves it unchanged. Furthermore, let t_l be the maximum breakpoint. Akturk and Yildirim (1998) also show that if $t > t_l$ then the SWPT rule gives an optimum sequence for the remaining unscheduled jobs. Therefore, we use this rule in Step 2.1 to find an optimal sequence for the remaining jobs on hand after a time point t_l.

3. Tuning experiments

With many GA tuning parameters, it is necessary to conduct separate experiments to determine appropriate values for each parameter and to assure that the final results are not biased by tuning. To maintain unbiasedness, we generated a set of problems used for tuning independent of the problems ultimately used in testing. Based on some initial trials, an experiment was designed to study the tuning parameters at the levels given in Table 1.

The last two factors in the experiment are range of due date 'RDD' and tardiness factor 'TF'. These factors determine the type of problem generated. By varying the RDD and TF factors, the experiments cover a broad range of problem types. This will help find tuning parameter values that work well across the range of possible problem types. For each of the 25 combinations of RDD

Downloaded by [Bilkent University] at 23:39 12 November 2017

Table 1. The values of the tuning parameters

Factors	*Number of levels*	*Values*				
POPSIZE	2	50	100			
Perturbation magnitude θ	3	0.5	1.0	2.0		
Selectivity π	2	2	4			
% SEXUAL	2	80%	100%			
Crossover type	2	Single point	Uniform			
MUTPROB	2	0.01	0.05			
RDD	5	0.2	0.4	0.6	0.8	1.0
TF	5	0.2	0.4	0.6	0.8	1.0

Downloaded by [Bilkent University] at 23:39 12 November 2017

and TF, one instance with 100 jobs was generated. A full factorial experiment was conducted over all tuning factors. For each combination of tuning parameters and for each test problem, two algorithm runs were made with different random number seeds, yielding two replicates (note that five replicates are used later in testing).

In our first pass analysis of the results of the experiment we observed that the variance was not constant across the factors RDD and TF. As non-constant variance violates the basic assumptions of the Analysis Of Variance (ANOVA), remedial measures were necessary. The reason for non-constant variance was readily apparent. Some combinations of RDD and TF produced easy problems with 'loose' due dates. Regardless of the tuning parameter values, the PSGA easily found a solution with zero total weighted tardiness for these easy problems. Other combinations of RDD and TF produced problems with a spectrum of difficulty levels. In general we observed higher variance on harder problems. Within each of the 25 cells formed by combinations of RDD and TF, we had 192 responses. To stabilize the variance we transformed each response to 'percentage from best solution' among the 192 in each cell. We then performed an ANOVA on the transformed data.

From the results of an ANOVA, we first note that crossover type and selectivity parameter π have no significant effect on the results. We also note the percent sexual reproduction has little effect. Perturbation magnitude θ, population size, and mutation probability all showed significant effects. It is expected that population size would be significant since twice as many solutions are generated with POPSIZE 100 as opposed to 50. As one would expect, both solution quality and computation time to increase with population size. In the testing phase we use both population sizes to examine the marginal benefits of generating more solutions.

The perturbation magnitude θ was also significant as expected. The experimental results show that $\theta = 0.5$ performed poorly, and that $\theta = 1$ was marginally better than $\theta = 2$. Our experience with problem space search methods indicates that performance is poor when θ is too small, but that performance is reasonably robust when θ is larger than its optimal value. The experimental results match precisely what we have learned from experience. Since $\theta = 1$ provided the best results, we chose this level in subsequent testing.

The final significant tuning parameter was mutation probability. A closer examination revealed an interaction between population size and mutation probability. When both mutation probability and population size were at their low levels (0.01 and 50 respectively), algorithm performance was poor. At all other combinations of levels, performance was roughly the same. Mutation is necessary to maintain diversity in the gene pool of a genetic algorithm. This is especially true in problem space genetic algorithms where we have found that aggressive selectivity works well. Our conclusion is that when both POPSIZE and MUTPROB are at their low levels, diversity is lost too quickly leading to poor performance. Since we test with both small and large population sizes, and since mutation probability interacts with POPSIZE, we decided not to fix MUTPROB *a priori*, but rather to examine its effects in testing as well. For a further discussion on the use of different genetic operators in the PSGA framework, such as how the perturbation vectors are found in each generation and how mutation is performed on each perturbation vector, we refer to Storer *et al*. (1992, 1995).

We proceeded to the testing phase having fixed the following tuning parameters: perturbation magnitude $\theta = 1.0$, selectivity $\pi = 4$, and the crossover type is single point. We also investigated the merits of several short GA runs (five runs of 200 generations) against a single longer run (1000 generations). The performance did not vary significantly thus we will use one long run in testing.

To summarize, we will test various PSGA algorithms by varying the parameters listed in Table 2.

Table 2. The parameters used in testing the PSGA algorithms

Parameter	*Level 1*	*Level 2*	*Level 3*
Base heuristic	ATC	ATC + GD	ATC + GD + LDR
POPSIZE	50	100	
MUTPROB	0.01	0.05	

Downloaded by [Bilkent University] at 23:39 12 November 2017

4. Description of experiments

To test the efficiency of the proposed PSGA's, the required programs were coded in the C language, compiled with a Borland compiler, and run on a Gateway 2000 model GP6-400 PC Pentium II 400 MHz with a memory of 96 MB RAM. The proposed algorithms were tested on a series of randomly generated problems developed by Crauwels *et al*. (1998). In addition, we also generated 125 new test problems for $n = 200$ using the same uniform distributions for p_j, w_j, RDD and TF. These new test problems and the computer codes for the PSGA and LDR methods may be obtained from the authors.

For each problem instance and factor combination, the algorithm was run five times with different random number seeds. The various algorithms were then compared in terms of the *average deviation* and the *maximum deviation* from the optimum (or best known) solution. The average and maximum were taken over the entire set of five replicates of each of 25 problem instances. The number of times (out of 125) that an optimum (or best known) solution was found was denoted as *total match*, and the *average CPU times* in seconds was also reported. Results are summarized in Table 3 for each value of n. We also report the average number of generations in which the best solution is found.

The best solution values appearing in Crauwels *et al*. (1998) for the 100 job problems are given on J.E. Beasley's OR-Library web site (*http://mscmga.ms.ic.ac.uk/jeb/orlib/wtinfo.html*) as file wtbest100a. Subsequent unpublished research has found slightly better solutions for some problems. These solutions are also available on the same web site. In our tables, comparison to the published results of Crauwels *et al*. (1998) appear in the row labeled $n = 100$ (A). Comparisons to the best known (unpublished) solutions to date appear in the row labeled $n = 100$ (B). In some cases we found better solutions to the 100 job problems than those of Crauwels *et al*. (1998) ($n = 100$ (A)). These are indicated as the '*number of improvements*'. The *total match* values in row $n = 100$ (A) indicate the number of times we found the exact same solution as Crauwels *et al*. (1998).

In order to find a 'best known solution' to each one of the 125 instances for the 200 job problems, we took a very long run with our best PSGA (using the LDR method). We used POPSIZE = 100, number of generations = 5000 (as opposed to 200 in all experiments), %SEXUAL = 0.8, MUTPROB = 0.01, $\theta = 1.0$, $\pi = 4$, and single point crossover. The algorithm was then run 10 times using different random number seeds, and the best solution found is reported as the best known.

5. Discussion of results

We begin by discussing the importance of the base heuristic in the performance of the PSGA, and then compare our results to the best known algorithms in the literature.

5.1. *Effects of the base heuristic*

We developed PSGA's with three different base heuristics, ATC, ATC + GD, and ATC + GD + LDR. One obvious question is how well each heuristic performs on its own when they run on the original problem data. Therefore, we present their results in parentheses in Table 3, which indicate that the ATC + GD + LDR method provides a significant improvement over the ATC rule in terms of average and maximum deviation. Furthermore, each successive base heuristic dominates the previous one. For each version, we tried several sets of GA parameter values. The results were quite striking. As the base heuristic improves, the overall PSGA algorithm performance also improves significantly. By improving the base heuristic, we were able to improve the quality of the neighborhood searched by the GA, and thus significantly improve the overall solution quality in all measures. Moreover, all of the PSGA's provide a significant improvement over the corresponding single pass heuristics. It is also worth noting that this improved performance comes with a relatively small increase in the CPU time. In terms of the overall solution quality, the best solution is given by the parameter setting of MUTPROB = 0.01 and POPSIZE = 100 for the ATC + GD + LDR base heuristic. Crauwels *et al*. (1998) also note that a genetic algorithm with the permutation representation performed so poorly compared to other local search heuristics, that they did not even report their results. This gives further credence to our observation that the neighborhood structure and encoding are far more important than the search mechanism.

5.2. *Algorithm effectiveness*

In the paper by Crauwels *et al*. (1998), the best results were obtained with tabu search. Their genetic algorithm with a binary encoding scheme was comparable and used less computation time than the other local search heuristics, but tabu search still gave the best overall results in terms of average and maximum deviation. The results reported in Crauwels *et al*. (1998) were quite good. Based on comparisons to optimal solutions for 40 and 50 job problems, there is little margin for improvement. Nevertheless, our computational results show that our best PSGA provided the largest *total match*, and the smallest *average* and *maximum deviation* (which also indicates its robustness) when compared to the best algorithm reported in Crauwels *et al*. (1998).

Another important feature of the PSGA is its running time behavior as a function of the number of jobs. In Crauwels *et al*. (1998), the local search heuristics were run on a HP 9000 - G50 computer. In their study, the genetic

Table 3. Computational Results

Base heuristic		*MUTPROB = 0.01*					
		POPSIZE = 50			*POPSIZE = 100*		
		ATC	*ATC + GD*	*ATC + GD + LDR*	*ATC*	*ATC + GD*	*ATC + GD + LDR*
$n = 40$	Total Match	72 (19)	81 (19)	122 (45)	79	86	125
	Ave. Dev.	0.31% (18.3)	0.26% (17.1)	0.00% (5.4)	0.18%	0.18%	0.00%
	Max. Dev.	9.4% (275)	8.1% (355)	0.20% (107)	8.3%	6.7%	0.0%
	Ave. Gen.	276.3	225.3	25.9	257.1	224.4	9.7
	CPU Time	31.4 (0.001)	13.3 (0.000)	14.7 (0.000)	62.9	26.3	29.3
$n = 50$	Total Match	53 (18)	57 (18)	119 (30)	59	66	121
	Ave. Dev.	0.52% (50.3)	0.43% (12.2)	0.01% (5.6)	0.23%	0.21%	0.00%
	Max. Dev.	10.7% (4200)	9.7% (182)	0.7% (128)	8.1%	8.1%	0.15%
	Ave. Gen.	373.5	274.3	54.5	388.6	238.9	33.8
	CPU Time	49.2 (0.002)	18.4 (0.000)	20.8 (0.000)	98.4	36.3	41.5
$n = 100$ (A)	Total Match	38 (18)	40 (18)	95 (25)	41	39	101
	Ave. Dev.	4.73% (41.5)	1.0% (39.5)	0.02% (10.7)	4.7%	0.13%	0.0%
	Max. Dev.	552% (2225)	82% (2225)	0.97% (162)	552%	2.7%	0.1%
	Ave. Gen.	617.2	557.8	211.1	591.6	565.9	148.2
	Number of Improvements	0	0	3	0	0	0
	CPU Time	186.4 (0.007)	52.3 (0.001)	60.8 (0.001)	368.3	103.9	120.8
$n = 100$ (B)	Total Match	38 (18)	40 (18)	94 (25)	41	39	103
	Ave. Dev.	4.73% (41.5)	1.00% (39.5)	0.02% (10.7)	4.70%	0.13%	0.00%
	Max. Dev.	552% (2225)	82% (2225)	0.97% (161.8)	552%	2.7%	0.1%
	Ave. Gen.	617.2	557.8	211.1	591.6	565.9	148.2
	Number of Improvements	0	0	0	0	0	0
	CPU Time	186.4 (0.007)	52.3 (0.001)	60.8 (0.001)	368.3	103.9	120.8
$n = 200$	Total Match	30 (21)	30 (21)	55 (24)	32	33	62
	Ave. Dev.	0.48% (16.2)	0.85% (15.5)	0.14% (12.0)	0.33%	0.40%	0.04%
	Max. Dev.	10.1% (154)	16.8% (157)	4.6% (142)	2.9%	7.8%	1.2%
	Ave. Gen.	689.6	709.3	429.3	692.9	685.4	418.9
	CPU Time	719 (0.026)	149.3 (0.004)	182 (0.003)	1414.5	297.4	362.1
				MUTPROB=0.05			
$n = 40$	Total Match	64	73	125	66	79	125
	Ave. Dev.	0.23%	0.06%	0.00%	0.11%	0.05%	0.00%
	Max. Dev.	9.38%	0.84%	0.00%	1.75%	0.56%	0.00%
	Ave. Gen.	389.12	244.31	5.97	415.73	277.18	4.78
	CPU Time	31.40	12.85	14.48	62.62	25.84	29.11
$n = 50$	Total Match	44	56	121	43	57	124
	Ave. Dev.	0.43%	0.25%	0.01%	0.39%	0.21%	0.00%
	Max. Dev.	8.09%	8.09%	1.27%	8.09%	8.09%	0.02%
	Ave. Gen.	495.30	405.3	34.15	510.15	365.57	42.11
	CPU Time	49.18	17.83	20.54	97.92	35.89	41.02
$n = 100$ (A)	Total Match	37	39	86	32	42	83
	Ave. Dev.	4.39%	0.35%	0.01%	2.79%	0.27%	0.02%
	Max. Dev.	455.56%	11.42%	0.22%	266.67%	2.11%	0.30%
	Ave. Gen.	602.32	618.26	208.38	601.14	604.62	233.98
	Number of Improvements	0	0	0	0	0	0
	CPU Time	185.83	50.46	59.17	370.13	101.54	118.97
$n = 100$ (B)	Total Match	37	39	86	32	42	83
	Ave. Dev.	4.39%	0.35%	0.01%	2.79%	0.27%	0.02%
	Max. Dev.	455.56%	11.42%	0.22%	266.67%	2.11%	0.30%
	Ave. Gen.	602.32	618.26	208.38	601.14	604.62	233.98
	Number of Improvements	0	0	0	0	0	0
	CPU Time	185.83	50.46	59.17	370.13	101.54	118.97

Downloaded by [Bilkent University] at 23:39 12 November 2017

Table 3. (*Continued*)

Base heuristic		*MUTPROB = 0.05*					
		POPSIZE = 50			*POPSIZE = 100*		
		ATC	*ATC + GD*	*ATC + GD + LDR*	*ATC*	*ATC + GD*	*ATC + GD + LDR*
$n = 200$	Total Match	29	32	47	29	33	48
	Ave. Dev.	1.36%	0.78%	0.13%	1.16%	0.66%	0.07%
	Max. Dev.	10.05%	9.08%	1.90%	6.64%	6.58%	0.59%
	Ave. Gen.	390.53	611.95	429.07	426.60	628.56	433.7
	CPU Time	714.63	143.72	176.83	1420.54	287.54	351.71

algorithm was the fastest among the several local search heuristics tested. Since our runs were taken on a PC Pentium II, we could not directly compare the actual CPU time requirements. Instead, we normalized both of the algorithms (i.e., our PSGA algorithm with the ATC + GD + LDR heuristic, and GA(B,1) and GA(B,5) by Crauwels *et al*. (1998)) in terms of the average CPU requirements for problems of size $n = 40$. We then plotted CPU times as a function of the number of jobs in order to show how the run times scale up. These results appear in Fig. 1. It is important to note that Crauwels *et al*. (1998) did not test problems of size $n = 200$. The apparent gradual increase in running time as a function of problem size is another advantage of the proposed PSGA over the other local search heuristics in the literature.

In order to investigate the importance of using a GA to search problem space, we compare the effectiveness of a GA to a pure 'probabilistic search'. In probabilistic search we generate each element of the perturbation vector from a Uniform $(-\theta, \theta)$ distribution where the tuning parameter θ is set to one as in the PSGA's. This is equivalent to creating one very large first generation, and performing no genetic operations nor generating any subsequent generations. We used the ATC + GD + LDR as the base heuristic. We repeat the probabilistic generation of solutions 25 000 and 50 000 times and report the best solution found. The random search performed poorly relative to the genetic algorithm versions. This indicates the value of an evolutionary strategy for this problem. We conjecture that the population of perturbations serves as an effective memory structure that learns as the algorithm proceeds. Our general conclusions, supported by empirical evidence, are that PSGA's are successful primarily due to neighborhood quality induced by the base heuristic and encoding scheme, but that the GA does provide some benefit over and above that of a much simpler probabilistic search.

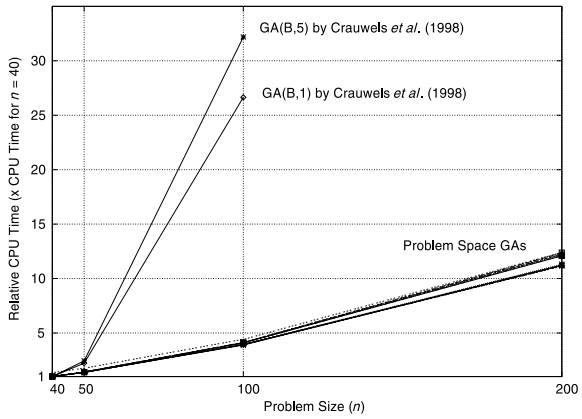

Fig. 1. CPU time comparison.

Acknowledgements

The authors would like to thank Prof. C.N. Potts and Prof. H.A.J. Crauwels for sharing their test results with us. This work was supported in part by the NATO Collaborative Research Grant CRG–971489 and NSF Grant DMI 9809479.

References

Akturk, M.S. and Yildirim, M.B. (1998) A new lower bounding scheme for the total weighted tardiness problem. *Computers and Operations Research*, **25**(4), 265–278.

Crauwels, H.A.J., Potts, C.N. and Van Wassenhove, L.N. (1998) Local search heuristics for the single machine total weighted tardiness scheduling. *INFORMS Journal on Computing*, **10**(3), 341–350.

Emmons, H. (1969) One machine sequencing to minimize certain functions of job tardiness. *Operations Research*, **17**, 701–715.

Lawler, E.L. (1977) A 'Pseudopolynomial' algorithm for sequencing jobs to minimize total tardiness. *Annals of Discrete Mathematics*, **1**, 331–342.

Morton, T.E. and Pentico, D.W. (1993) *Heuristic Scheduling Systems with Applications to Production Systems and Project Management*, Wiley, New York, NY.

Potts, C.N. and Van Wassenhove, L.N. (1985) A branch and bound algorithm for total weighted tardiness problem. *Operations Research*, **33**, 363–377.

Potts, C.N. and Van Wassenhove, L.N. (1991) Single machine tardiness sequencing heuristics. *IIE Transactions*, **23**, 346–354.

Rinnooy Kan, A.H.G., Lageweg, B.J. and Lenstra, J.K. (1975) Minimizing total costs in one-machine scheduling. *Operations Research*, **23**, 908–927.

Downloaded by [Bilkent University] at 23:39 12 November 2017

Storer, R.H., Wu, S.D. and Vaccari, R. (1992) New search spaces for sequencing problems with application to job shop scheduling. *Management Science*, **38**(10), 1495–1509.

Storer, R.H., Wu, S.D. and Vaccari, R. (1995) Local search in problem and heuristic space for job shop scheduling. *ORSA Journal on Computing*, **7**(4), 453–467.

Biographies

Selcuk Avci is a research staff member in the CASTLE Laboratory in the Department of Operations Research and Financial Engineering at Princeton University. He received his B.S. in Mechanical Engineering from the Middle East Technical University, Turkey, M.S. in Industrial Engineering from the Bilkent University, Turkey, and Ph.D. in Industrial Engineering from Lehigh University. His current research interests include production planning and scheduling, transportation planning and logistics, inventory theory and modern optimization heuristics.

M. Selim Akturk is an Associate Professor of Industrial Engineering at Bilkent University, Turkey. He holds a Ph.D. in Industrial Engineering from Lehigh University, USA and B.S.I.E. and M.S.I.E. degrees from the Middle East Technical University, Turkey. His current research interests include hierarchical planning of large scale systems, production scheduling, cellular manufacturing systems, and advanced manufacturing technologies. He is a senior member of IIE and member of INFORMS.

Robert H. Storer is Professor of Industrial and Systems Engineering, Co-Director of the Manufacturing Logistics Institute, and Co-Director of the Integrated Business and Engineering Honors Program at Lehigh University. He received his B.S in Industrial and Operations Engineering from the University of Michigan in 1979, and M.S. and Ph.D. degrees in Industrial and Systems Engineering from the Georgia Institute of Technology in 1982 and 1986 respectively. His interests lie in operations research and applied statistics with particular interest in heuristic optimization, scheduling and logistics.

Contributed by the Scheduling Department

Downloaded by [Bilkent University] at 23:39 12 November 2017