New Heuristics for No-Wait Flowshops with a Linear Combination of Makespan and Maximum Lateness

Ruben Ruiz, Ali Allahverdi

New Heuristics for No-Wait Flowshops with a Linear Combination of Makespan and Maximum Lateness

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2007-IJPR-0872.R1
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	10-Mar-2008
Complete List of Authors:	Ruiz, Ruben; Universidad Politecnica de Valencia, DEIOAC Allahverdi, Ali; Kuwait University, Dept of Indl and Mgmt Systems Engineering
Keywords:	SCHEDULING, FLOW SHOP, FLOW SHOP SCHEDULING, MAKESPAN, META-HEURISTICS, DISPATCHING RULES, GENETIC ALGORITHMS
Keywords (user):	SCHEDULING, FLOW SHOP

New Heuristics for No-Wait Flowshops with a Linear Combination of Makespan and Maximum Lateness

Rubén Ruiz[1] and Ali Allahverdi[2]

[1]Instituto Tecnológico de Informática,
Universidad Politécnica de Valencia,
Camino de Vera S/N, 46021,
Valencia, Spain
Email: rruiz@eio.upv.es

[2]Department of Industrial and Management Systems Engineering,
College of Engineering and Petroleum,
Kuwait University, P.O. Box 5969, Safat, Kuwait,
Fax: (+965)481-6137
allahverdi@kuc01.kuniv.edu.kw

Abstract

In this work we study a flowshop scheduling problem in which jobs are not allowed to wait in-between machines, a situation commonly referred to as no-wait. The concerned criterion is to minimize a weighted sum of makespan and maximum lateness. A dominance relation for the case of three-machine is presented and evaluated by experimental designs. Several heuristics and local search methods are proposed for the general *m*-machine case. The local search methods are based on genetic algorithms and iterated greedy procedures. An extensive computational analysis is conducted where it is shown that the proposed methods outperform existing heuristics and metaheuristics in all tested scenarios by a considerable margin and under identical CPU times.

Keywords: No-wait flowshop, bicriteria, makespan, maximum lateness, dominance relation

[1] Corresponding author. Telephone number: +34 96 387 70 07, ext. 74946. Fax number: +34 96 387 74 99

1. Introduction

This paper considers the *m*-machine no-wait flowshop scheduling problem. In a no-wait flowshop problem, the operation of each job has to be processed without interruptions between consecutive machines. Applications of no-wait flowshops can be found in many industries. For example, in steel factories, the heated metal continuously goes through a sequence of operations before it is allowed to cool in order to prevent defects in the composition of the steel. A second example is a plastic product that requires a series of processes to immediately follow one another in order to prevent degradation. Similar situations arise in other process industries such as the chemical and pharmaceutical. Hall and Sriskandarajah (1996) provide a survey paper on the research and applications of the no-wait flowshop problems.

For the problem with the single criterion of makespan, or using the common three field notation, the problem $Fm/no-wait/C_{\max}$, Reddi and Ramamoorthy (1972) and Wismer (1972) were the first to address the problem. Bonney and Gundry (1976) and King and Spachis (1980) have later developed heuristics. Gangadharan and Rajendran (1993) and Rajendran (1994) have developed additional heuristics and showed that their heuristics outperform those of Bonney and Gundry (1976) and King and Spachis (1980). Aldowaisan and Allahverdi (2003) presented new heuristics, which were shown to outperform those of Gangadharan and Rajendran (1993) and Rajendran (1994). More recently, Grabowski and Pempera (2005) have presented complex local search algorithms for the same problem based on tabu search methods. Additionally, and as it is well known, the $Fm/no-wait/C_{\max}$ problem can be transformed into a Traveling Salesman Problem (TSP) (see for example Bagchi, Gupta and Sriskandarajah (2006)) and therefore, all the existing TSP literature also applies.

Unlike the makespan criterion, and to the best of our knowledge, there is no known heuristic when the objective is to minimize maximum lateness for the general *m*-machine case. Actually, the literature on the $Fm/no-wait/L_{\max}$ is rather scant. Dileepan (2004) presented a dominance relation for the two machine case. More recently, Wang and Cheng (2006) have also studied the two machine case and have proposed heuristics to solve the problem.

The bicriteria problem has also been scarcely investigated. Allahverdi and Aldowaisan (2004) addressed the *m*-machine no-wait flowshop with the objective of minimizing a weighted sum of makespan and maximum lateness. Allahverdi and Aldowaisan (2004) developed a

dominance relation for the two-machine case, and presented two hybrid heuristics for the case of *m*-machines based on simulated annealing and genetic algorithms. A related work is that of Allahverdi and Aldowaisan (2002) where a constructive heuristic along with some improvement methods were presented for the same problem but for the objectives of makespan and total completion time.

In this paper we address the same problem that Allahverdi and Aldowaisan (2004) studied. We provide a dominance relation for the case of three-machines. Moreover, we propose several dispatching rules as well as an adapted genetic algorithm and an iterated greedy method. As we will show, the proposed methods clearly outperform those of Allahverdi and Aldowaisan (2004) by a considerable margin under comparable conditions.

Problem description and formulation is given in the next section and a dominance relation is presented in Section 3. The proposed heuristics are described in Section 4 and computational analysis is carried out in Section 5. Finally, conclusions and future research are presented in Section 6.

2. Problem Description

In the *m*-machine no-wait flowshop scheduling problem, each job needs to be processed on machines 1, 2, …, *m* in that order such that the operation of each job has to be processed without interruptions between consecutive machines. Therefore, if necessary, the start of a job on a given machine might be delayed so that the completion of the operation coincides with the beginning of the operation on the following machine and the no-wait constraint is satisfied. All jobs are ready and available for processing at time zero. We assume that every job has a positive processing time on each machine. At any time, machines can process at most one job at a time and jobs can be processed on at most one machine. The problem is to find a schedule that minimizes a linear combination of makespan and maximum lateness, i.e., $\alpha C_{max}+(1-\alpha)L_{max}$. Simplifications of this problem have been proven to be NP-Hard. For example, the $F2/no-wait/L_{\max}$ was shown to be NP-Hard by Röck (1984a). Similarly, the $F3/no-wait/C_{\max}$ was also shown to be NP-Hard by Röck (1984b) (more precisely, the associated decision problem was shown to be NP-Complete). Therefore, the $Fm/no-wait/\alpha C_{\max}+(1-\alpha)L_{\max}$ problem is also NP-Hard.

In the following we present some necessary notation. Let:

$t_{i,k}$: processing time of job i on machine k (k=1, …, m)

d_i : due date of job i

$t_{[i,k]}$: processing time of the job in position i on machine k

$d_{[i]}$: due date of the job in position i

$C_{[i]}$: completion time of the job in position i

$L_{[i]}$: lateness of the job in position i

It can be shown that for the job in position j when m=3,

$$C_{[j]} = \sum_{i=1}^{j} (t_{[i,2]}-t_{[i-1,2]} - t_{[i-1,3]} + max\,(t_{[i,1]},\ t_{[i-1,2]}))^{+} + \sum_{i=1}^{j} t_{[i,3]} \quad (1)$$

where, $t_{[0,2]} = t_{[0,3]} = 0$. Notice that for simplicity $max(0,X)$ is noted as $(X)^{+}$.

Therefore, the makespan, C_{max}, can be written as

$$C_{max} = \sum_{i=1}^{n} (t_{[i,2]}-t_{[i-1,2]} - t_{[i-1,3]} + max\,(t_{[i,1]},\ t_{[i-1,2]}))^{+} + \sum_{i=1}^{n} t_{[i,3]} \quad (2)$$

Given $C_{[j]}$, the lateness of the job in position j, $L_{[j]}$, can be written as

$$L_{[j]} = \sum_{i=1}^{j} (t_{[i,2]}-t_{[i-1,2]} - t_{[i-1,3]} + max\,(t_{[i,1]},\ t_{[i-1,2]}))^{+} + \sum_{i=1}^{j} t_{[i,3]} - d_{[j]} \quad (3)$$

Hence, maximum lateness, L_{max}, is expressed as

$$L_{max} = \max_{i=1,\ldots,n}\{ L_{[i]} \} \quad (4)$$

The objective is to minimize $\alpha C_{max}+(1-\alpha)L_{max}$. We assume that every job requires processing on all the m machines, i.e., $t_{j,k}>0$. Such an m-machine no-wait flowshop is necessarily a permutation flow shop. In other words, the order of jobs on all machines must be the same.

The above formulae for the three machine cases can be generalized to the general case of m-machines. It can be shown that (from a simplification of Allahverdi and Aldowaisan (2000) and Brown, McGarvey and Ventura (2004)) for the job in position i on machine m:

$$C_{[i]} = \sum_{l=1}^{i} C_{[l]}^{m} \quad (5)$$

where, $C_{[l]}^{m} = \max\left\{C_{[l]}^{m-1} - t_{[l-1,m]};0\right\} + t_{[l,m]}$

It should be noted that $C_{[l]}^{0} = t_{[0,k]} = 0$, l=1,…,n; k=1,…,m.

Once the completion times of every job in the sequence ($C_{[i]}$) are calculated, the lateness of the job in position i, $L_{[i]}$, can be written as

$$L_{[i]} = C_{[i]} - d_{[i]} \quad (6)$$

Therefore, the maximum lateness or $L_{\max}$, is calculated as in expression (4).

Once these calculations are carried out, the bicriteria objective function for a given sequence is straightforward to calculate and can be carried out in $O(nm)$ steps.

3. A Dominance Relation for Three-Machines

Dominance relations are common in the scheduling literature. They are mainly used either as part of heuristics or more commonly, in implicit enumeration techniques such as branch-and-bound algorithms. We provide a dominance relation in the following theorem for the considered problem.

Theorem: For a three-machine no-wait flowshop, there exists an optimal solution that minimizes both C_{max} and L_{max} where job j precedes job i if jobs i and j are adjacent and if the following three conditions are satisfied; (i) $t_{j,k} \leq t_{i,k}$ for k=1,2,3, (ii) $d_j \leq d_i$, and (iii) $t_{i,2}+max(t_{i,1}, t_{j,2}) \leq t_{j,2}+t_{j,3}$.

Proof: See the Appendix.

Corollary: For a three-machine no-wait flowshop, in a sequence where jobs i and j are adjacent, there exists an optimal solution for minimizing $\alpha C_{max}+(1-\alpha)L_{max}$ for any value of α where job j precedes job i if the following conditions are satisfied: (i) $t_{j,k} \leq t_{i,k}$ for k=1,2,3, (ii) $d_j \leq d_i$, and (iii) $t_{i,2}+max(t_{i,1}, t_{j,2}) \leq t_{j,2}+t_{j,3}$.

Proof: The proof directly follows from the previous theorem since if these conditions are satisfied then both C_{max} and L_{max} are simultaneously minimized, and hence, $\alpha C_{max}+(1-\alpha)L_{max}$ is also minimized for any value of α.

One important aspect is to asses how effective this dominance relation might be. Checking the dominance relation has a negligible cost but, what about the expected number of times that the dominance relation is going to be satisfied? In order to answer this question, we generate a large set of random instances. To the best of our knowledge, there is no common benchmark in

the literature for the considered problem. Therefore, in the following we define a common benchmark set that will be used both for testing the dominance relation and later on for the computational evaluation of the existing and proposed heuristics. The number of jobs n is tested at n={20, 40, 60, 80, 100} and the number of machines m at m={3, 5, 10, 15, 20}. Of course, the dominance relation will be tested only for m=3. The due dates are generated as usual in the literature with a uniform distribution that follows the formula [P(1-T-R/2),P(1-T+R/2)] where P is a lower bound on the makespan and T and R are the tardiness factor and relative range of the due dates, respectively. This method is extensively used in the literature and is due to Potts and Van Wassenhove (1982). We need a bound P on the makespan which is calculated as follows:

$$P = LB(C_{\max}) = \min_{i=1,\ldots,n}\left(\sum_{k=1}^{m-1} t_{i,k}\right) + \sum_{i=1}^{n} t_{i,m} \qquad (7)$$

Obviously this is a trivial bound on the makespan. Fortunately, we are not interested in tight lower bounds since they will tend to generate looser due dates (and henceforth, easy to solve problems). Once P is defined we have four different combinations of T and R, namely T={0.0, 0.6} and R={0.2, 0.6}. Lower values of T generate looser due dates whereas high values of T generate very tight due dates. As regards to R, lower values result in due dates of jobs being closer together and high values result in due dates being more scattered. Presumably, the "hardest" combination is T-R=0.6-0.2 since this results in due dates being uniformly distributed between 0.3P and 0.5P, and since P is not a very tight bound on the makespan, we can expect due dates to be really difficult to satisfy. Another important factor to consider is how processing times are distributed, therefore we uniformly generate them in the range [1,5], [1,10], [1,50], [1,100] and according to a Gaussian distribution of average 100 and standard deviation of 2. Overall we have five combinations of n, five of m, four of T-R and five combinations for the distribution of processing times. This results in 500 different combinations. For each combination we generate 10 different problems to a grand total of 5,000 instances. The full set of instances as well as the best solutions known for them are available for download from http://soa.iti.es.

For testing the dominance relation, we restrict to the subset where m=3 (1,000 instances). For every instance there are n·(n-1) possible pairs of jobs in a sequence. An important measure of the performance of this dominance rule is to count how many of these possible pairs actually

satisfy it. We compare the percentage of times that the dominance rule is satisfied among the $n\cdot(n\text{-}1)$ possible pairs (%Satisfied). Additionally, it is easy to see that every job pair that satisfies the dominance rule results in $(n\text{-}1)!$ solutions saved if used in a branch and bound algorithm. However, in this case, the same solution can be saved by several pairs of jobs satisfying the dominance rule, so counting the estimated number of nodes saved over the total $n!$ is not an easy task. A first experimental design in which all the instance characteristics are considered as factors is analyzed by means of the ANOVA technique. The response variable is %Satisfied for each instance. From the resulting analysis, we can conclude that the only significant factor is the distribution of processing times whose means plot and confidence intervals can bee seen in Figure 1.

[INSERT FIGURE 1 ABOUT HERE]

As can be seen, tightly distributed processing times result in more pairs of jobs satisfying the dominance relation. Overall, when the processing times are uniformly distributed between 1 and 5, there is an average of 1.22% pairs of jobs in the instances that satisfy the dominance relation. While this percentage might be small, one has to think that each pair saves $(n\text{-}1)!$ solutions at best. For example, for a problem with 50 jobs, 1.22% means about 30 pairs that satisfy the dominance relation and that can translate to a best scenario of about 60% of solutions saved in a branch and bound algorithm. Clearly, the usage of the proposed dominance relation for the three machine case in exact algorithms is a worthwhile venue for research. In the following we propose heuristic methods for the more general m-machine case.

4. Heuristic and local search methods

4.1. Heuristic methods

Among the most widely used heuristic algorithms for due date based scheduling problems are dispatching rules. We make use of some of the most well known dispatching rules for the bicriteria no-wait flowshop problem considered in this paper. More precisely, the dispatching rules are:

1. Earliest Due Date (EDD). The sequence of jobs is obtained by sorting the jobs on increasing order of their due dates d_i.

2. Modified Due Date (MDD). In this rule, the next scheduled job is the one with the minimum modified due date $\max\{d_i, C_i(\sigma)\}$.
3. SLACK. At each step, the completion time of job *i*, when appended to the partial sequence σ, i.e. $C_i(\sigma)$, is calculated. The job with the minimum slack with regards to its due date or $d_i - C_i(\sigma)$ is appended to the sequence.
4. SLACK with Remaining Work (SLACKRW). It is a modification of the previous SLACK rule where the total processing time of each job *i* is also considered. As a result, the job with the minimum value in the following index is scheduled:

$$\frac{d_i - C_i(\sigma)}{\sum_{k=1}^{m} t_{i,k}}$$

5. NEH heuristic. This well known constructive heuristic, initially proposed by Nawaz, Enscore and Ham (1983) has been profusely used in the scheduling literature. Although it was originally devised for the regular flowshop with makespan objective, its structure allows for an easy adaptation to other problems and criteria. In the NEH, jobs are first sorted in non-increasing order of their total processing times. The first job in the ordered list is first scheduled. Then, at each step, the remaining jobs in the list are selected and inserted in all possible solutions of the partial sequence, keeping the best one for the next insertions. We tested several different initial orders for the considered bicriteria objective (a much more in depth study about different initial orders for the NEH method can be seen in Framinan, Leisten and Rajendran (2003)). The best results for most tested values of α resulted to be the EDD rule. Therefore, prior to the application of the NEH algorithm, jobs are initially ordered according to the EDD rule.

4.2. Local search methods

The performance of the previous dispatching rules cannot be expected to be competitive when compared against modern metaheuristics. In what follows we explain two competitive algorithms that we apply to the $Fm/no-wait/\alpha C_{\max} + (1-\alpha)L_{\max}$ problem obtaining very good results.

A recent new breed of fast genetic algorithms was presented in Ruiz and Allahverdi (2007). In that work, a total of four genetic algorithms were presented. Among the four, the

steady state version with local search yielded the best results. This algorithm is referred to as SGALS. Here we explain the basic workings of this method and of the special adaptations for the no-wait flowshop with the bicriteria objective. Given that no-wait flowshops are necessarily permutation flowshops, a non-delay schedule is best represented by a simple permutation of jobs. This representation is adopted for the individuals that form the population of the genetic algorithm. In order to match the characteristics of the bicriteria problem, all individuals in the initial population are generated at random except two. These two solutions are generated after applying the aforementioned EDD and NEH heuristics.

One of the salient characteristics of the proposed genetic algorithm is that selection is carried out on the basis of the direct bicriteria objective value. No mappings and/or fitness values are assigned to individuals. This keeps the algorithm simpler and at the same time speeds up selection and population upkeep. According to a given parameter called pressure%, a given percentage of the population is randomly screened, and among the screened individuals, the one with the best bicriteria objective value is selected as a parent. For selecting the second parent, the first one is not considered in the draw. Therefore, we ensure that two distinct parents are selected for crossover. This type of selection has been refereed to as *n-tournament* and results in both fast and simple genetic algorithms. After the extensive calibration and experimentation carried out in Ruiz and Allahverdi (2007), the population size was set to 50 individuals and the pressure parameter to 30%.

In the same study, the best crossover operator resulted to be the Two Point Order crossover which is a simple and at the same time fast operator since it always generates feasible offspring. This crossover operator is applied to both parents after selection with a probability of 0.3 to generate two feasible offspring. If the operator is not applied, direct copies of parents are generated as offspring. After the crossover, a simple insertion mutation operator is applied to each position in the chromosome with a probability of 0.02. This is, each position in the chromosome is randomly extracted and re-inserted in another random position with a low probability of 0.02.

After generating the new offspring with selection, crossover and mutation, these new individuals undergo an improvement phase. This improvement is carried out by local search. We improve each individual with a probability of 0.15 by means of a limited local search in which each job is removed from the sequence and tested in all the possible positions of the sequence

(i.e., n positions). The lowest bicriteria objective value dictates the final position for the job. The process stops when all jobs have been reinserted. This local search does not guarantee the final improved offspring to be local optima with respect to the insertion neighborhood but it is much faster.

The last important characteristic of the adapted SGALS algorithm is that it is a steady stage genetic algorithm. Once the two offspring have been finally generated and possibly improved by the curtailed local search procedure, they are considered for insertion in the population. They are only inserted if they satisfy two criteria; a) they are not already in the population (the same bicriteria objective value is not present in the population or if it is, the permutations are different) and b) they have a lower bicriteria objective value than the worst individual in the population. Only after these two criteria are satisfied, offspring are inserted in the population replacing the two worst individuals. Such a scheme has been also tested in successful GAs (see Ruiz, Maroto and Alcaraz (2006)) since it provides a fast, albeit diverse, generational scheme avoiding at the same time premature convergence.

The second local search method we present in this work is based on the recent Iterated Greedy (IG) algorithm proposed by Ruiz and Stützle (2007) for the regular flowshop problem with makespan criterion and later adapted by the same authors (see Ruiz and Stützle (2008)) to the sequence dependent setup times flowshop with makespan and total weighted tardiness objectives. According to those studies, the Iterated Greedy algorithm with local search (IGLS) is currently the best algorithm in those problems. Hence, it seems natural that we adapt the IGLS method to the scheduling problem tackled in this study.

As the name implies, IG algorithms start from a given initial solution and then iterate through four operators; destruction, construction, local search and acceptance criterion. These four operators are adapted to the bi-criteria problem as follows: First, we construct an initial solution by the modified NEH algorithm given in Section 4.1. Then we enter the main loop. In this loop, the incumbent solution is first partially deconstructed (destruction). To this end, a given number of jobs are randomly selected and removed from the sequence, leaving a partial sequence as a result. Afterwards, the removed jobs in the destruction phase are re-inserted in this partial sequence one at a time, starting from the job that was removed first. This phase is called construction and each job that was removed is tested in all possible positions of the partial

sequence, just like in the NEH or in the local search procedures outlined above. The construction phase ends when there are no more jobs to re-insert and thus the sequence is complete. After the incumbent sequence has been put through destruction and construction, a full local search procedure is applied. In this case, the local search algorithm outlined for the SGALS is repeated until no further improvements are obtained, i.e., the resulting sequence is a local optima with respect to the insertion neighborhood. Notice that this local search procedure is noticeably slower that the curtailed one. However, IGLS mainly relies on this local search to obtain good results.

The last step in this IGLS algorithm is to decide if the sequence obtained after the local search should replace the incumbent sequence for the next iteration. We use the same approach as in Ruiz and Stützle (2007) which consists in accepting the new sequence according to a fixed temperature simulated annealing criterion, in which worse solutions have a small probability of being accepted. The proposed IGLS only needs a single parameter which is the number of jobs that are removed from the sequence and later re-inserted in the destruction phase. Some short experiments indicate that the original value of four given by Ruiz and Stützle (2007) is a good choice.

As a result, we propose five heuristics, one genetic algorithm with local search and an iterated greedy method. In total there are seven algorithms that we apply to the $Fm/no-wait/\alpha C_{\max}+(1-\alpha)L_{\max}$ problem.

5. Computational Evaluation

As mentioned in Section 2, the only existing work that deals with the $Fm/no-wait/\alpha C_{\max}+(1-\alpha)L_{\max}$ problem considered in this paper is that of Allahverdi and Aldowaisan (2004). Other related papers are Allahverdi and Aldowaisan (2002), where the $Fm/no-wait/\alpha C_{\max}+(1-\alpha)\sum C_i$ problem is addressed or Ruiz and Allahverdi (2007) where a single criterion no-wait version with setups ($Fm/no-wait,s_{ij}/L_{\max}$) is studied. For the computational evaluation, we will compare the proposed heuristics and local search methods against some of these algorithms. The hybrid genetic algorithm and hybrid simulated annealing of Allahverdi and Aldowaisan (2004) will be referred to as HG and HSA, respectively, following the original names given by the authors. The heuristic proposed by Allahverdi and Aldowaisan (2002) will be called PAAH.

All algorithms are coded in Delphi 2006 and share most functions and all data structures. We test every algorithm in a Pentium IV PC/AT computer running at 3.2 GHz with 2 GBytes of RAM memory and Windows XP professional operating system. From the 5,000 instances introduced in Section 3, we focus on those in which the processing times are uniformly distributed in the range [1,100] as it is usual in the literature. This is because a heuristic performing well for a wide rage of processing times is expected to perform well for other narrower ranges. Similarly, we skip instances where m=3. Therefore, the benchmark test comprises 800 instances with different values of T, R, n and m. We measure the average percentage deviation over the best solution known or:

$$AVRPD = \left(\frac{\text{Alg}_{\text{sol}} - \text{Best}_{\text{sol}}}{\text{Best}_{\text{sol}}} \right) \times 100 \quad (8)$$

Where Alg_{sol} is the solution for any of the tested algorithms for a given instance and Best_{sol} is the best solution known for that instance. As mentioned, the best solutions can be obtained from http://soa.iti.es.

All algorithms are run five independent times. This is necessary in the case of the stochastic methods such as HG, HSA, SGALS and IGLS in order to better asses the quality of solutions. In the case of all other tested methods (EDD, MDD, NEH, PAAH, SLACK and SLACKRW) this is not necessary since the solution is deterministic. However, we also run five replicates in order to better estimate running times by averaging the results. We run all algorithms and replicates against the 800 instances six different times in which we vary the value of α (the relative weight assigned to C_{max}) in the bicriteria objective function $\alpha C_{max}+(1-\alpha)L_{max}$. The values tested are α={0.0, 0.2, 0.4, 0.6, 0.8, 1.0}. Notice that when α=0.0 (α=1.0), the considered bicriteria problem reduces to L_{max} (C_{max}).

An important issue as regards to the computational evaluation is the stopping criterion for the metaheuristic methods. The existing algorithms HSA and HG have a specific stopping criterion that was set by Allahverdi and Aldowaisan (2004) according to their own results. We choose to run these algorithms with this original or "natural" stopping criterion and will refer to them as HSA_N and HG_N, respectively.

Apart from this natural stopping criterion, and considering that all algorithms have been coded in the same language, sharing most functions and being run on the same computer, a much better stopping criterion is a maximum elapsed CPU time. By doing so, all algorithms are run

under the same computational effort on the same computer and therefore, the *AVRPD* values are more directly comparable. We modify the HG and HSA algorithms so that they stop at a specific elapsed CPU time. Both proposed methods IGLS and SGALS also use this stopping criterion which is set to $n\cdot(m/2)\cdot 60$ milliseconds. As we can see, as the size of the instance grows, more CPU time is allowed.

In summary, we have 800 instances, six values for α, 12 algorithms (EDD, HG, HG_N, HSA, HSA_N, IGLS, MDD, NEH, PAAH, SGALS, SLACK, SLACKRW) and five replicates for each algorithm, instance and value of α. This results in a total of 288,000 data points.

Let us first analyze the results for all instances grouped by T and R. Table 1 and Table 2 show the results for all values of α. Bold and italic underlined values indicate the best and the worst average results, respectively.

[INSERT TABLE 1 ABOUT HERE]
[INSERT TABLE 2 ABOUT HERE]

As can be seen from the results, there seems to be a trend for lower *AVRPD* values as α increases. Recall that the case of α=0.0 reduces to the single criterion L_{max} whereas the case of α=1.0 reduces to C_{max}. For α=0.0, the average results of all methods are significantly larger than for larger values of α. This does not mean that instances are harder in this case, rather the contrary. For many instances, the optimum values of L_{max} are small, in one case even 0 and for a few instances even negative. The way the *AVRPD* values are calculated tends to generate very large *AVRPD* values when the optimum solution for a given instance is close to zero, and thus the larger overall *AVRPD* values for α=0.0. This together with the fact that the *AVRPD* values depend on the best solutions known (and there is no clear idea about how far these best solutions might be from the optimum ones) makes the interpretation of the effect of α difficult. What seems clear though is that the dispatching rules EDD, MDD, SLACK and SLACKRW yield solutions that are, as expected, considerably worse than those of the other methods. Judging average values alone, it seems that SLACK and SLACKRW are the worst and best performers, respectively. PAAH performs better than NEH in most situations. This outcome is certainly remarkable, since NEH has been regarded as the best constructive heuristic for many flowshop related scheduling problems. Afterwards, as average performance is concerned, we find the two

previous existing algorithms from Allahverdi and Aldowaisan (2004) with the natural and elapsed CPU time stopping criterion, namely HG, HG_N, HSA and HSA_N with similar performances throughout the tests. HSA and HSA_N yield near identical results. However, HG and HG_N show different results, which favor HG across the board. This indicates that probably the original stopping criterion for HG_N does not result in enough time. Lastly, it is clear that the two proposed IGLS and SGALS algorithms are the best performers with a slight advantage for IGLS although not for every value of α.

As regards to the effect of T and R, it can be seen that for most methods and values of α, an increase in R results in worse solutions whereas an increase in T results in better *AVRPD* values. The effect of R is contrary to what was expected but the effect of T was definitely envisioned. Of course, this trend is very strong for low values of α and, as one expects, weak for high values of α.

Of additional interest are the results when grouped by n and m values. Such a grouping for α=0.6 is presented in Table 3.

[INSERT TABLE 3 ABOUT HERE]

Overall, all methods result in higher *AVRPD* values as both n and m increase. Although for large values of n the trend is less clear. More or less the same situation is found for other values of α, but we do not present them here for the sake of limited space.

Now we analyze the CPU times consumed by all methods. These times are not affected by T, R or α, as they mainly depend on the number of jobs n and to a lesser extent on the number of machines m. These CPU times are shown in Table 4 for α=0.6.

[INSERT TABLE 4 ABOUT HERE]

Dispatching rules are extremely fast even for the largest 100×20 instances as they need a negligible CPU time. As a matter of fact, the elapsed times are on the limit of what can be reliably measured (les than 0.001 seconds). The adaptation of the NEH method is also very fast, needing an average of 0.09 seconds for the largest instances. PAAH, is considerably slower than NEH, needing a bit less than two seconds for the largest instances. As expected, IGLS and

SGALS use the same times since these two methods stop by a given elapsed CPU time given by the expression $n \cdot (m/2) \cdot 60$ milliseconds. Notice that this time results in 30 seconds for the largest instances which is still a short CPU time. It is important to mention that both HSA and HG are expected to have the same time. However, in these two methods, and as specified by Allahverdi and Aldowaisan (2004), a final local search step is applied the final solution, therefore, HSA and HG times are slightly larger than those of IGLS and SGALS. Comparing HSA and HG with the elapsed CPU time stopping criterion against the HSA_N and HG_N versions with Allahverdi and Aldowaisan (2004) stopping criterion, we see that HSA_N and HG_N are much faster. In the case of HSA_N this is good since we observed in Table 3 that there is no difference in performance between HSA and HSA_N. However, for HG, the additional CPU time results in better solutions. It is worth noting that HG_N and HSA_N times are comparable to those of PAAH.

All previous results are based on average values. While there is a large number of data points (and therefore the observed averages are likely to approach those averages at the population level), a more comprehensive statistical analysis must be carried out. We analyze a complete statistical experiment in which all factors affecting the instances; T, R, n, m and α are considered along with a last factor that controls the algorithm. The response variable in the experiment is the *AVRPD*. The dispatching rules EDD, MDD, SLACK and SLACKRW had to be analyzed separately from the other methods due to much larger *AVRPD* values. All studied factors resulted to be statistically significant at a 95% confidence level for α values larger than 0.0. All dispatching rules presented the same statistical performance for α=0.0. Figure 2 shows a means plot of the *AVRPD* values for the interaction between the dispatching rules and α.

[INSERT FIGURE 2 ABOUT HERE]

All tested dispatching rules work with due dates, and therefore, this plot confirms the previous findings that increasing α values (closer to C_{max} objective) result in lower *AVRPD* values. There are no statistically significant differences for many α levels. However, we can see that MDD is worse than EDD and SLACKRW for α=0.2 but better than all others for α=1.0.

A similar analysis can be carried out for the remaining methods. A means plot of the *AVRPD* values across all other factors is shown in Figure 3.

[INSERT FIGURE 3 ABOUT HERE]

Three groups can be observed. First we have the NEH and PAAH algorithms. Clearly, PAAH yields better solutions in all scenarios but as we have shown earlier, at a much higher CPU time. On the other hand, HSA and HG algorithms have better performance with both stopping criteria. At this stage it is important to note that both HSA_N and HG_N are statistically better than PAAH and even faster on average. Although HG is shown as statistically equivalent to HG_N, differences can be found for specific values of α (not shown). Lastly, it is clear that IGLS and SGALS outperform all other methods, including HSA and HG by a considerably margin. Taking into account that the CPU times for IGLS and SGLS are slightly smaller than those of HSA and HG, we can safely conclude that IGLS and SGALS are preferable for all scenarios. Although not shown here, the performances for different values of α, T, R and other factors are similar and IGLS and SGALS are consistently better in all situations. A comparison of these two algorithms is shown in Figure 4.

[INSERT FIGURE 4 ABOUT HERE]

As we can see, there are no differences for all values of α except for α=0.0. This is an unexpected result since IGLS was specifically designed for the C_{max} objective. Some differences can be found between these two algorithms when studying other factors. Although not shown, IGLS shows better results than SGALS for n=80 across all other factors. The same can be said for R=0.6, T=0.0 and m=5. Therefore, we can conclude that IGLS performs better than SGALS in many cases and statistically equivalent in other cases. Moreover, the coding of IGLS is simpler, and therefore, it is preferable to SGALS.

6. Concluding Remarks

In this paper, the m-machine no-wait scheduling problem is addressed to minimize a linear combination of makespan and maximum lateness performance measures. A dominance relation is developed for the case of three-machines. Moreover, several heuristics are proposed and compared with the existing heuristics for the problem. The computational experiments and

thorough statistical analyses indicate that the proposed heuristics significantly outperform the existing ones for all scenarios considered; different linear combinations of the objectives, tardiness factors, ranges of due dates and number of jobs and machines. Since the comparisons have been done in the same computer with identical stopping criterion, and the codes of all tested algorithms share most functions, it can be safely stated that the computational evaluation is fair and therefore our proposed methods outperform the existing ones.

There are usually two approaches to solve the problem addressed in this paper. One is use an implicit enumeration technique such as a branch-and-bound algorithm by which problems with a limited number of jobs can be solved optimally. The other is to use heuristics to solve the problem, where an optimal solution may not be obtained but problems with much larger number of jobs and machines can be easily solved. We have chosen the latter approach to address the problem since we could solve larger realistic size problems, and more importantly, we were able to compare our proposed heuristics with the existing ones in the literature. Furthermore, we also established a dominance relation for the case m=3. Dominance relations are very helpful when used in implicit enumeration techniques. Therefore, a possible research area is to construct a branch-and-bound algorithm for this problem by utilizing the dominance relation established in this paper or to develop a Lagrangian relaxation approach similar to the one used by Augusto et al. (2008).

We have assumed that setup times are included in the processing times. This assumption is perfectly valid for some environments. However, the assumption is not realistic for some other manufacturing environments, e.g., Yu et al. (2007), Hendizadeh et al. (2007), Pessan et al. (2008), and Chandrasekaran et al. (2007). The significance of considering setup times as separate is addressed by Allahverdi and Soroush (2008), and a recent survey of scheduling with separate setup times is given by Allahverdi et al. (2008). Therefore, one possible research area is to address the problem where setup times are explicitly treated as separate from processing times. Another possible extension is to consider the problem with respect to other objective functions such as job waiting time variance, e.g., Li et al., (2007) or an objective function taking into account early and tardy penalties, e.g., Valente, (2007). It would be also interesting to consider a hybrid flowshop, e.g., Ben Hmida et al. (2007).

Further work needs to be conducted in order to deal the bi-criteria problem in a more explicit way by building methods capable of obtaining a full set of non-dominated solutions in the bi-criteria space (a posteriori approach) which would eliminate the need of fixing α values.

REFERENCES

Aldowaisan, T. and Allahverdi, A. (2003). "New heuristics for no-wait flowshops to minimize makespan", Computers & Operations Research, 30 (8):1219-1231.

Allahverdi, A. and Aldowaisan, T. (2000). "No-wait and separate setup three-machine flowshop with total completion time criterion", International Transactions in Operational Research, 7 (3):245-264.

Allahverdi, A. and Aldowaisan, T. (2002). "No-wait flowshops with bicriteria of makespan and total completion time", Journal of the Operational Research Society, 53 (9):1004-1015.

Allahverdi, A. and Aldowaisan, T. (2004). "No-wait flowshops with bicriteria of makespan and maximum lateness", European Journal of Operational Research, 152 (1):132-147.

Allahverdi, A., Ng, C.T., Cheng, T.C.E., and Kovalyov, M.Y. (2008) "A survey of scheduling problems with setup times or costs", *European Journal of Operational Research* , Vol. 187, 985-1032.

Allahverdi, A., Soroush, H.M. (2008) "The significance of reducing setup times/setup costs", *European Journal of Operational Research,* Vol. 187, 978-984.

Augusto, V., Xie, X., Perdomo, V., (2008). " Operating theatre scheduling using Lagrangian relaxation", European Journal of Industrial Engineering, 2(2), 172-189.

Bagchi, T. P., Gupta, J. N. D. and Sriskandarajah, C. (2006). "A review of TSP based approaches for flowshop scheduling", European Journal of Operational Research, 169 (3):816-854.

Ben Hmida, A., Huguet, M.J., Lopez, P., Haouari, M. (2007). "Climbing depth-bounded discrepancy search for solving hybrid flow shop problems", European Journal of Industrial Engineering, 1(2): 223-240.

Bonney, M. C. and Gundry, S. W. (1976). "Solutions to the constrained flowshop sequencing problem", Operational Research Quarterly, 27 (4):869-883.

Brown, S. I., McGarvey, R. and Ventura, J. A. (2004). "Total flowtime and makespan for a no-wait m-machine flowshop with set-up times separated", Journal of the Operational Research Society, 55 (6):614-621.

Chandrasekaran, C., Rajendran, C., Krishnaiah Chetty, O.V., Hanumanna, D. (2007). " Metaheuristics for solving economic lot scheduling problems (ELSP) using time-varying lot-sizes approach", European Journal of Industrial Engineering, 1(2): 152-181.

Dileepan, P. (2004). "A note on minimizing maximum lateness in a two-machine no-wait flowshop", Computers & Operations Research, 31 (12):2111-2115.

Framinan, J. M., Leisten, R. and Rajendran, C. (2003). "Different initial sequences for the heuristic of Nawaz, Enscore and Ham to minimize makespan, idletime or flowtime in the static permutation flowshop sequencing problem", International Journal of Production Research, 41 (1):121-148.

Gangadharan, R. and Rajendran, C. (1993). "Heuristic algorithms for scheduling in the no-wait flowshop", International Journal of Production Economics, 32 (3):285-290.

Grabowski, J. and Pempera, J. (2005). "Some local search algorithms for no-wait flow-shop problem with makespan criterion", Computers & Operations Research, 32 (8):2197-2212.

Hall, N. G. and Sriskandarajah, C. (1996). "A survey of machine scheduling problems with blocking and no-wait in process", Operations Research, 44 (3):510-525.

Hendizadeh, S.H., ElMekkawy, T.Y., Wang, G.G. (2007). "Bi-criteria scheduling of a flowshop manufacturing cell with sequence dependent setup times", European Journal of Industrial Engineering, 1(4), 391-413.

King, J. R. and Spachis, A. S. (1980). "Heuristics for flowshop scheduling", International Journal of Production Research, 18 (3):345-357.

Li, X., Ye, N., Xu, X., Sawhey, R. (2007). "Influencing factors of job waiting time variance on a single machine", European Journal of Industrial Engineering, 1(1): 56-73.

Nawaz, M., Enscore, Jr. E. E. and Ham, I. (1983). "A heuristic algorithm for the m machine, n job flowshop sequencing problem", Omega-International Journal of Management Science, 11 (1):91-95.

Pessan, C., Bouquard, J.L., Neron, E. (2008). " An unrelated parallel machines model for an industrial production resetting problem", European Journal of Industrial Engineering, 2(2), 153-171.

Potts, C. N. and Van Wassenhove, L. N. (1982). "A decomposition algorithm for the single machine total tardiness problem", Operations Research Letters, 1 (5):177-181.

Rajendran, C. (1994). "A no-wait flowshop scheduling heuristic to minimize makespan", Journal of the Operational Research Society, 45 (4):472-478.

Reddi, S. S. and Ramamoorthy, C. V. (1972). "On the flowshop sequencing problem with no-wait in process", Operational Research Quarterly, 23 (3):323-331.

Röck, H. (1984a). "Some new results in flow shop scheduling", Mathematical Methods of Operations Research (ZOR), 28 (1):1-16.

Röck, H. (1984b). "The three-machine no-wait flow shop is NP-Complete", Journal of the ACM, 31 (2):336-345.

Ruiz, R. and Allahverdi, A. (2007). "No-wait flowshop with separate setup times to minimize maximum lateness", International Journal of Flexible Manufacturing Technology, 35(5-6):551-565.

Ruiz, R., Maroto, C. and Alcaraz, J. (2006). "Two new robust genetic algorithms for the flowshop scheduling problem", Omega-International Journal of Management Science, 34 (5):461-476.

Ruiz, R. and Stützle, T. (2007). "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem", European Journal of Operational Research, 177(3):2033-2049.

Ruiz, R. and Stützle, T. (2008). "An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives", European Journal of Operational Research, 187(3): 1143-1159.

Valente, J.M.S. (2007). "Heuristics for the single machine scheduling problem with early and quadratic tardy penalties", European Journal of Industrial Engineering, 1(4), 431-448.

Wang, X. L. and Cheng, T. C. E. (2006). "A heuristic approach for two-machine no-wait flowshop scheduling with due dates and class setups", Computers & Operations Research, 33 (5):1326-1344.

Wismer, D. A. (1972). "Solution of the flowshop scheduling problem with no intermediate queues", Operations Research, 20 (3):689-697.

Yu, X., Ram, B., Jiang, X. (2007). "Parameter setting in a bio-inspired model for dynamic flexible job shop scheduling with sequence-dependent setups", *European Journal of Industrial Engineering,* 1(2): 182-199

Appendix (Proof of the Theorem)

Consider two job sequences π_1 and π_2 such that π_1 has job i in an arbitrary position τ and job j in position $\tau+1$. The sequence π_2 is exactly the same as π_1 except that job j is in position τ and job i in position $\tau+1$.

The makespan for these two sequences can be written as:

$$C_{max}(\pi_1) = \sum_{r=1}^{\tau-1} (t_{[r,2]}-t_{[r-1,2]} - t_{[r-1,3]} + max\,(t_{[r,1]},\ t_{[r-1,2]}))^+ + \sum_{r=1}^{\tau-1} t_{[r,3]}$$
$$+ (t_{i,2} - t_{[\tau-1,2]} - t_{[\tau-1,3]} + max\,(t_{i,1},\ t_{[\tau-1,2]}))^+$$
$$+ (t_{j,2} - t_{i,2} - t_{i,3} + max\,(t_{j,1},\ t_{i,2}))^+$$
$$+ (t_{[\tau+2,2]} - t_{j,2} - t_{j,3} + max\,(t_{[\tau+2,1]},\ t_{j,2})\} + t_{i,3} + t_{j,3} + t_{[\tau+2,3]}$$
$$+ \sum_{r=\tau+3}^{n} (t_{[r,2]}-t_{[r-1,2]} - t_{[r-1,3]} + max\,(t_{[r,1]},\ t_{[r-1,2]}))^+ + \sum_{r=\tau+3}^{n} t_{[r,3]},$$

and

$$C_{max}(\pi_2) = \sum_{r=1}^{\tau-1} (t_{[r,2]}-t_{[r-1,2]} - t_{[r-1,3]} + max\,(t_{[r,1]},\ t_{[r-1,2]}))^+ + \sum_{r=1}^{\tau-1} t_{[r,3]}$$
$$+ (t_{j,2} - t_{[\tau-1,2]} - t_{[\tau-1,3]} + max\,(t_{j,1},\ t_{[\tau-1,2]}))^+$$
$$+ (t_{i,2} - t_{j,2} - t_{j,3} + max\,(t_{i,1},\ t_{j,2}))^+$$
$$+ (t_{[\tau+2,2]} - t_{i,2} - t_{i,3} + max\,(t_{[\tau+2,1]},\ t_{i,2}))^+ + t_{j,3} + t_{i,3} + t_{[\tau+2,3]}$$
$$+ \sum_{r=\tau+3}^{n} (t_{[r,2]}-t_{[r-1,2]} - t_{[r-1,3]} + max\,(t_{[r,1]},\ t_{[r-1,2]}))^+ + \sum_{r=\tau+3}^{n} t_{[r,3]}$$

where $t_{[0,2]} = t_{[0,3]} = 0$.

Since both sequences have the same jobs in all positions except τ and $\tau+1$, it follows from the above two equations that

$$C_{max}(\pi_2) - C_{max}(\pi_1) = (t_{j,2} - t_{[\tau-1,2]} - t_{[\tau-1,3]} + max\,(t_{j,1},\ t_{[\tau-1,2]}))^+$$
$$+(t_{i,2} - t_{j,2} - t_{j,3} + max\,(t_{i,1},\ t_{j,2}))^+$$
$$+ (t_{[\tau+2,2]} - t_{i,2} - t_{i,3} + max\,(t_{[\tau+2,1]},\ t_{i,2}))^+$$
$$- (t_{i,2} - t_{[\tau-1,2]} - t_{[\tau-1,3]} + max\,(t_{i,1},\ t_{[\tau-1,2]}))^+$$
$$- (t_{j,2} - t_{i,2} - t_{i,3} + max\,(t_{j,1},\ t_{i,2}))^+$$

$$- (t_{[\tau+2,2]} - t_{j,2} - t_{j,3} + max\ (t_{[\tau+2,1]},\ t_{j,2}))^+ \quad (9)$$

Observe that $(t_{i,2} - t_{j,2} - t_{j,3} + max\ (t_{i,1},\ t_{j,2}))^+ = 0$ since $t_{i,2}+max(t_{i,1},\ t_{j,2}) \leq t_{j,2}+t_{j,3}$, Therefore, equation (9) reduces to

$$\begin{aligned} C_{max}(\pi_2) - C_{max}(\pi_1) \quad &= \quad (t_{j,2} - t_{[\tau-1,2]} - t_{[\tau-1,3]} + max\ (t_{j,1},\ t_{[\tau-1,2]}))^+ \\ &+ (t_{[\tau+2,2]} - t_{i,2} - t_{i,3} + max\ (t_{[\tau+2,1]},\ t_{i,2}))^+ \\ &- (t_{i,2} - t_{[\tau-1,2]} - t_{[\tau-1,3]} + max\ (t_{i,1},\ t_{[\tau-1,2]}))^+ \\ &- (t_{j,2} - t_{i,2} - t_{i,3} + max\ (t_{j,1},\ t_{i,2}))^+ \\ &- (t_{[\tau+2,2]} - t_{j,2} - t_{j,3} + max\ (t_{[\tau+2,1]},\ t_{j,2}))^+ \\ &= (t_{j,2} - t_{[\tau-1,2]} - t_{[\tau-1,3]} + max\ (t_{j,1},\ t_{[\tau-1,2]}))^+ \\ &+ (t_{[\tau+2,2]} - t_{i,3} + (t_{[\tau+2,1]} - t_{i,2})^+)^+ \\ &- (t_{i,2} - t_{[\tau-1,2]} - t_{[\tau-1,3]} + max\ (t_{i,1},\ t_{[\tau-1,2]}))^+ \\ &- (t_{j,2} - t_{i,2} - t_{i,3} + max\ (t_{j,1},\ t_{i,2}))^+ \\ &- (t_{[\tau+2,2]} - t_{j,3} + (t_{[\tau+2,1]} - t_{j,2})^+)^+ \end{aligned}$$

Since $t_{j,k} \leq t_{i,k}$ for k=1,2,3, it follows from the above equation that $C_{max}(\pi_2) \leq C_{max}(\pi_1)$.

The lateness of the jobs in positions τ and τ+1 for the two sequences are given as:

$$L_{[\tau]}(\pi_1) = \sum_{r=1}^{\tau-1} (t_{[r,2]} - t_{[r-1,2]} - t_{[r-1,3]} + max\ (t_{[r,1]},\ t_{[r-1,2]}))^+ + \sum_{r=1}^{\tau-1} t_{[r,3]}$$

$$+ (t_{i,2} - t_{[\tau-1,2]} - t_{[\tau-1,3]} + max\ (t_{i,1},\ t_{[\tau-1,2]}))^+ + t_{i,3} - d_i, \quad (10)$$

$$L_{[\tau]}(\pi_2) = \sum_{r=1}^{\tau-1} (t_{[r,2]} - t_{[r-1,2]} - t_{[r-1,3]} + max\ (t_{[r,1]},\ t_{[r-1,2]}))^+ + \sum_{r=1}^{\tau-1} t_{[r,3]}$$

$$+ (t_{j,2} - t_{[\tau-1,2]} - t_{[\tau-1,3]} + max\ (t_{j,1},\ t_{[\tau-1,2]}))^+ + t_{j,3} - d_j, \quad (11)$$

$$L_{[\tau+1]}(\pi_1) = \sum_{r=1}^{\tau-1} (t_{[r,2]} - t_{[r-1,2]} - t_{[r-1,3]} + max\ (t_{[r,1]},\ t_{[r-1,2]}))^+ + \sum_{r=1}^{\tau-1} t_{[r,3]}$$

$$+ (t_{i,2} - t_{[\tau-1,2]} - t_{[\tau-1,3]} + max\ (t_{i,1},\ t_{[\tau-1,2]}))^+ + t_{i,3}$$

$$+ (t_{j,2} - t_{i,2} - t_{i,3} + max\ (t_{j,1},\ t_{i,2}))^+ + t_{j,3} - d_j, \quad (12)$$

$$L_{[\tau+1]}(\pi_2) = \sum_{r=1}^{\tau-1} (t_{[r,2]}-t_{[r-1,2]} - t_{[r-1,3]} + max\,(t_{[r,1]},\ t_{[r-1,2]}))^+ + \sum_{r=1}^{\tau-1} t_{[r,3]}$$

$$+ (t_{j,2}-t_{[\tau-1,2]} - t_{[\tau-1,3]} + max\,(t_{j,1},\ t_{[\tau-1,2]}))^+ + t_{j,3}$$

$$+ (t_{i,2}-t_{j,2} - t_{j,3} + max\,(t_{i,1},\ t_{j,2}))^+ + t_{i,3} - d_i, \quad (13)$$

It follows from equations (11) and (12) that

$$L_{[\tau]}(\pi_2)-L_{[\tau+1]}(\pi_1) = (t_{j,2}-t_{[\tau-1,2]} - t_{[\tau-1,3]} + max\,(t_{j,1},\ t_{[\tau-1,2]}))^+$$

$$- (t_{i,2}-t_{[\tau-1,2]} - t_{[\tau-1,3]} + max\,(t_{i,1},\ t_{[\tau-1,2]}))^+ - t_{i,3}$$

$$- (t_{j,2}-t_{i,2} - t_{i,3} + max\,(t_{j,1},\ t_{i,2}))^+$$

Now it follows from the last equation and the fact that $t_{j,k} \le t_{i,k}$ for k=1,2,3

$$L_{[\tau]}(\pi_2) \le L_{[\tau+1]}(\pi_1). \quad (14)$$

Taking the difference between equations (12) and (13) yields

$$L_{[\tau+1]}(\pi_2)-L_{[\tau+1]}(\pi_1) = (t_{j,2}-t_{[\tau-1,2]} - t_{[\tau-1,3]} + max\,(t_{j,1},\ t_{[\tau-1,2]}))^+$$

$$+ (t_{i,2}-t_{j,2} - t_{j,3} + max\,(t_{i,1},\ t_{j,2}))^+$$

$$- (t_{i,2}-t_{[\tau-1,2]} - t_{[\tau-1,3]} + max\,(t_{i,1},\ t_{[\tau-1,2]}))^+$$

$$- (t_{j,2}-t_{i,2} - t_{i,3} + max\,(t_{j,1},\ t_{i,2}))^+$$

$$+ d_j - d_i$$

But $(t_{i,2}-t_{j,2} - t_{j,3} + max\,(t_{i,1},\ t_{j,2}))^+ = (t_{j,2}-t_{i,2} - t_{i,3} + max\,(t_{j,1},\ t_{i,2}))^+ = 0$ since $t_{i,2}+max(t_{i,1}, t_{j,2}) \le t_{j,2}+t_{j,3}$, and $t_{j,2}+max(t_{j,1}, t_{i,2}) \le t_{i,2}+t_{i,3}$, Moreover, $t_{j,k} \le t_{i,k}$ for k=1,2,3 and $d_j \le d_i$, hence,

$$L_{[\tau+1]}(\pi_2) \le L_{[\tau+1]}(\pi_1) \quad (15)$$

Therefore. from equations (14) and (15)

$$max\{ L_{[\tau]}(\pi_2), L_{[\tau+1]}(\pi_2)\} \le max\{ L_{[\tau]}(\pi_1), L_{[\tau+1]}(\pi_1)\} \quad (16)$$

Now, for $L_{[r]}$ where $r = \tau + 2,...,n$,

$$L_{[r]}(\pi_1) = \sum_{r=1}^{\tau-1} \{ t_{[r,2]} - t_{[r-1,2]} - t_{[r-1,3]} + max\,(t_{[r,1]},\ t_{[r-1,2]})\} + \sum_{r=1}^{\tau-1} t_{[r,3]}$$

$$+ (t_{i,2} - t_{[\tau-1,2]} - t_{[\tau-1,3]} + max\,(t_{i,1},\ t_{[\tau-1,2]}))^+ + t_{i,3}$$

$$+ (t_{j,2} - t_{i,2} - t_{i,3} + max\,(t_{j,1},\ t_{i,2}))^+ + t_{j,3}$$

$$+ (t_{[\tau+2,2]} - t_{j,2} - t_{j,3} + max\,(t_{[\tau+2,1]},\ t_{j,2}))^+ + t_{[\tau+2,3]}$$

$$+ \sum_{p=\tau+3}^{r} (t_{[p,2]} - t_{[p-1,2]} - t_{[p-1,3]} + max\,(t_{[p,1]},\ t_{[p-1,2]}))^+$$

$$+ \sum_{p=\tau+3}^{r} t_{[p,3]} - d_{[r]},$$

and

$$L_{[r]}(\pi_2) = \sum_{r=1}^{\tau-1} (t_{[r,2]} - t_{[r-1,2]} - t_{[r-1,3]} + max\,(t_{[r,1]},\ t_{[r-1,2]}))^+ + \sum_{r=1}^{\tau-1} t_{[r,3]}$$

$$+ (t_{j,2} - t_{[\tau-1,2]} - t_{[\tau-1,3]} + max\,(t_{j,1},\ t_{[\tau-1,2]}))^+ + t_{j,3}$$

$$+ (t_{i,2} - t_{j,2} - t_{j,3} + max\,(t_{i,1},\ t_{j,2}))^+ + t_{i,3}$$

$$+ (t_{[\tau+2,2]} - t_{i,2} - t_{i,3} + max\,(t_{[\tau+2,1]},\ t_{i,2}))^+ + t_{[\tau+2,3]}$$

$$+ \sum_{p=\tau+3}^{r} (t_{[p,2]} - t_{[p-1,2]} - t_{[p-1,3]} + max\,(t_{[p,1]},\ t_{[p-1,2]}))^+$$

$$+ \sum_{p=\tau+3}^{r} t_{[p,3]} - d_{[r]},$$

where $\sum_{p=\tau+3}^{\tau+2} (.) = 0.$

From the last two equations.

$$L_{[r]}(\pi_2) - L_{[r]}(\pi_1) = (t_{j,2} - t_{[\tau-1,2]} - t_{[\tau-1,3]} + max\,(t_{j,1},\ t_{[\tau-1,2]}))^+$$

$$+ (t_{i,2} - t_{j,2} - t_{j,3} + max\,(t_{i,1},\ t_{j,2}))^+$$

$$+ (t_{[\tau+2,2]} - t_{i,2} - t_{i,3} + max\,(t_{[\tau+2,1]},\ t_{i,2}))^{+}$$

$$- (t_{i,2} - t_{[\tau-1,2]} - t_{[\tau-1,3]} + max\,(t_{i,1},\ t_{[\tau-1,2]}))^{+}$$

$$- (t_{j,2} - t_{i,2} - t_{i,3} + max\,(t_{j,1},\ t_{i,2}))^{+}$$

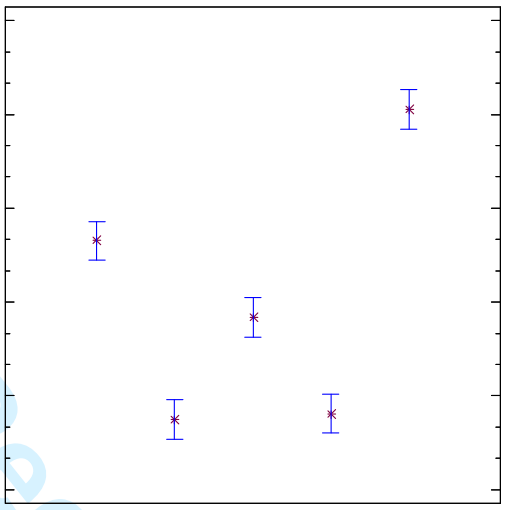
$$- (t_{[\tau+2,2]} - t_{j,2} - t_{j,3} + max\,(t_{[\tau+2,1]},\ t_{j,2}))^{+}$$

The right hand side of the above equation is equal to the right hand side of equation (9), hence

$$L_{[r]}(\pi_2) \leq L_{[r]}(\pi_1) \qquad (17)$$

for $r = \tau + 2, \ldots, n$. Needles to say $L_{[r]}(\pi_2) \leq L_{[r]}(\pi_1)$, $r = 1,2, \ldots, \tau-1$ since both sequences have the same jobs in these positions. Therefore, from equations (16) and (17), $L_{max}(\pi_2) \leq L_{max}(\pi_1)$.

FIGURES



```
ERROR: invalidaccess
OFFENDING COMMAND: --filter--

STACK:

/LZWDecode
-filestream-
[400 0 0 -195 0 195 ]
true
195
400
-savelevel-
```