A network flow IP formulation and exact/heuristics approaches for just-in-time scheduling problems on parallel machines without idle times

Rainer Xavier de Amorim[1], Rosiane de Freitas[1], Eduardo Uchoa[2]

[1]Programa de Pós-Graduação em Informática (PPGI)
Instituto de Computação - Universidade Federal do Amazonas (IComp - UFAM)
Manaus - Amazonas - Brasil
[2]Departamento de Engenharia de Produção - Universidade Federal Fluminense (UFF)
Niterói - Rio de Janeiro - Brasil
{rainer,rosiane}@icomp.ufam.edu.br, uchoa@producao.uff.br

ABSTRACT

This article presents a type of scheduling problem widely used in the industry, known as just-in-time (JIT) scheduling or earliness-tardiness scheduling problem, where independent jobs with arbitrary processing times and weights on single and parallel machines are considered. A time-indexed mathematical formulation based on network flow, without machine idle-time, for single and parallel machines is presented for the problem. Algorithmic strategies are developed involving local search and path-relinking techniques, single and multi-start global search approaches. The best results are obtained with Iterated Local Search with multi-start global search, with tests performed on Tanaka's single machine instances, achieving optimal solutions in most cases tested. Moreover, the methods presented are also suitable for dealing with multi-machine instances, where it is also possible to achieve optimal solutions, in most cases tested, in a reasonable execution time.

KEYWORDS: algorithms, earliness and tardiness penalties, integer programming, scheduling theory.
Main area: Combinatorial Optimization.

1. Introduction

The study of scheduling problems which considers simultaneously the penalties of earliness and tardiness in job execution (E/T scheduling) has been motivated by the adoption of the production concept without looseness (not before or after the due date), in industries of productive process, that is, products where production finishes at the exact moment it must be delivered to the customer, characterizing *Just-in-time* (JIT) scheduling process. This term came into use in Japan in the 1970's, in the automotive industry, where the objective was to find a system in which is possible to coordinate it production with a specific demand and the smallest tardiness toward possible, aiming the improvement of production process and manufacturing flow, elimination of inventory and wastes. The application of this system is very ample, involving the production of perishable products, for example, and involving any system in which the jobs must be finished as close as possible to the due date [Leung and Anderson (2004)].

In this paper we present some heuristic strategies with single-start and multi-start global search approaches, based on Local Search and Path-Relinking techniques to solve a scheduling problem on single and parallel machines to minimize the earliness and tardiness penalties of jobs, considering arbitrary processing times and weights. A time-indexed formulation based on network flow is also proposed for this E/T scheduling problem, without machine idle time, for single and parallel machines. Exact methods of implicit enumeration were applied, through CPLEX and UFFLP [Pessoa and Barboza (2011)] computational tools.

This paper is organized as follows: In Section 2, some preliminary definitions and notations are presented. In Section 2.1, some related works for E/T scheduling are commented on briefly. In Section 3, the strategies proposed in this article for E/T scheduling on single and parallel machines are detailed: *Iterated Local Search*, *Iterated Local Search* with *Path-Relinking* between local optimal solutions, *Genetic Algorithm* using *Local Search* with *Path-Relinking* between local optimal solutions and *Iterated Local Search* with *multi-start* global search. The results of the proposed strategies are presented in Section 4. Concluding remarks are made in the last section.

2. Our E/T Scheduling Problem

The scheduling problem addressed in this paper consists of the minimization of both earliness and tardiness penalties on identical parallel machines and n independent jobs J_j, $j = \{1,...,n\}$ with an arbitrary processing time p_j, a suggested time to finish - due date d_j, a completion time C_j and different earliness and tardiness weights, α_j and β_j, respectively, of *Job J* [Brucker (2006), Pinedo (2012)]. *Job J* is early if $C_j \leq d_j$; its earliness $E_j = max\{0, d_j - C_j\}$ and *Job J* is tardy if $C_j > d_j$; its tardiness $T_j = max\{0, C_j - d_j\}$. The completion time of a *Job J* is given by $C_j = t + p_j$, where t represents the exactly time that the job j starts its processing. The total amount of earliness and tardiness is represented by the following generalization of the objective function: $\sum_{j=1}^{n} E_j + \sum_{j=1}^{n} T_j$. This problem is referenced as $P||\sum \alpha_j E_j + \sum \beta_j T_j$ in *three-field notation* proposed by Graham et al. (1979).

Considering this problem, in this article we propose a time-indexed formulation with a network flow, based on the classical time-indexed formulation [Dyer and Wolsey (1990)], which is presented below, where machines idle time is not permitted and the machines cannot process more than one job at a time. All the jobs can be processed from time zero. Thus, the jobs should be processed in the interval $[0,T]$, where $T = \left\lfloor \frac{\sum_{j=1}^{n} p_j - p_{max}}{m} \right\rfloor + p_{max}$, and p_{max} is the maximum processing time of all jobs and m is the number of machines. The binary decision variables y_j^t indicate that job j starts at time t on some machine. The Weighted Earliness-Tardiness Scheduling Formulation (WETSF) can be formulated as follows:

$$\text{Min} \sum_{t=0}^{T} \sum_{j=1}^{n} f_j(C_j) y^t{}_j \tag{1}$$

$$\text{S. a.} \sum_{t=0}^{T} y^t{}_j = 1 \quad (j = 1,2,...,n) \tag{2}$$

$$\sum_{j=1}^{n} y_j^{t-p_j} - \sum_{j=0}^{n} y_j^t = 0 \quad (t = 1,...,T) \tag{3}$$

$$\sum_{j=1}^{n} y_j^0 = m \tag{4}$$

$$y^t{}_j \in \{0,1\} \quad (j = 1,2,...,n; t = 0,...,T) \tag{5}$$

The objective function $f_j(C_j)$ (1) in this formulation is based on the problem $P||\sum\alpha_j E_j + \sum\beta_j T_j$. Thus, the value of the objective function can be calculated as follows: $f_j(C_j) = \sum\alpha_j \cdot max\{0, d_j - C_j\} + \sum\beta_j \cdot max\{0, C_j - d_j\}$. The constraints (2) state that every job can be processed only once. The constraints (3) determine the sequence of jobs without idle time, based on network flow. The idle time is only permitted when all jobs finishes their processing, where the idle time is represented by the dummy job 0 (zero), which is used at the end of the sequence of jobs on each machine, this dummy job is directed to the maximum time of scheduling, as illustrated in Figure 1 (a).

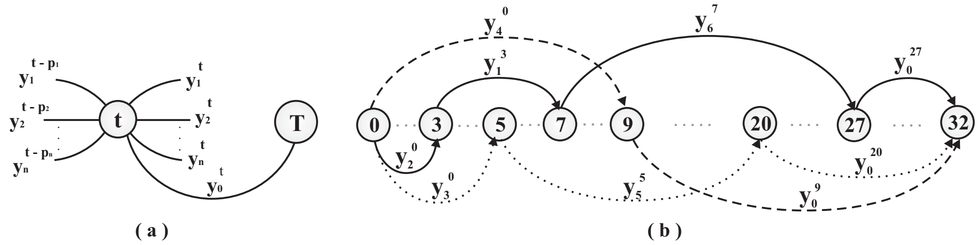
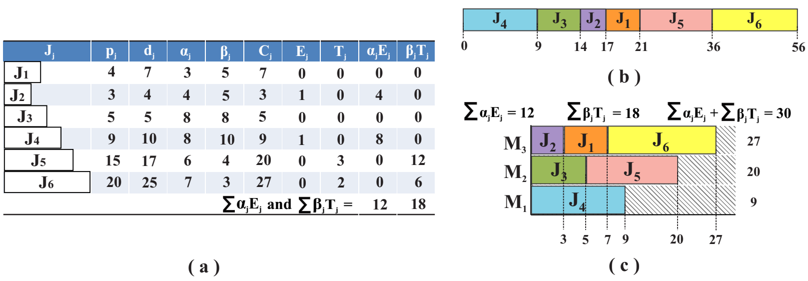


Figure 1. (a) Scheme of the constraints (3) for determining the sequence of jobs using network flow (b) Scheduling representation using network flow model for the classical formulation of the problem, for the solution presented in the Figure 2 (c).

The constraints (4) eliminate the occurrences of idle time at the beginning of the scheduling. The constraints (5) define the type of variables used, $0-1$ integer decision variables. An example of scheduling using the scheduling representation through a network flow is presented in Figure 1 (b), for the solution presented in Figure 2 (c), where each path is distinguished by a different arrow in the network presented in Figure 1 (b), which represents a scheduling on some machine.

An instance for this problem is composed by an integer n, that represents the number of jobs to be scheduled, and arrays of n integers for processing times p_j, due dates d_j and, for earliness and tardiness weights (α_j and β_j). Figure 2 (a) presents an instance example for the problem with 6 jobs, a solution for this instance is presented in Figure 2 (b), using the single sequence representation (used in the algorithm proposed in this article, which simplifies the treatment of the problem), when the algorithm needs to calculate the scheduling for parallel machines, the jobs are distributed in the machines (without idle time), as presented in Figure 2 (c).



J_j	p_j	d_j	α_j	β_j	C_j	E_j	T_j	$\alpha_j E_j$	$\beta_j T_j$
J_1	4	7	3	5	7	0	0	0	0
J_2	3	4	4	5	3	1	0	4	0
J_3	5	5	8	8	5	0	0	0	0
J_4	9	10	8	10	9	1	0	8	0
J_5	15	17	6	4	20	0	3	0	12
J_6	20	25	7	3	27	0	2	0	6
							$\sum\alpha_j E_j$ and $\sum\beta_j T_j$ =	12	18

Figure 2. (a) An instance for earliness-tardiness scheduling problem. (b) Single sequence representation for multi-machine schedules. (c) Scheduling representation for parallel machines.

2.1. Related Works

This section presents some related works regarding scheduling problems under earliness and tardiness penalties on single and identical parallel machines.

Baker and Scudder (1990) focus on problems involving earliness and tardiness scheduling with common due dates in their literature review. Classification and variations for scheduling problems involving common due dates can also be found in Gordon et al. (2002). Shabtay and Steiner (2012) present a literature review on *Just-in-Time* scheduling problems.

Scheduling problems with earliness and tardiness penalties with Genetic Algorithm as algorithmic strategy can be seen in Rym and M'Hallah (2007). The literature demonstrates some algorithmic strategies for the weighted tardiness scheduling problem which may be adapted for the weighted earliness-tardiness scheduling problem, such as the Genetic Algorithm proposed by Liu et al. (2005) in which genetic operators can be adapted for the problem considered in this article.

An exact approach which can be adapted for the weighted earliness-tardiness scheduling problem can be seen in Pessoa et al. (2010), where a robust exact algorithmic strategy for $P||\sum w_j T_j$ scheduling problem is proposed, and where a *Branch-Cut-and-Price* method was developed applying as primal heuristic, one proposed by Rodrigues et al. (2008), with an innovative feature where multi-machine schedules are represented by a single sequence, greatly simplifying the treatment of that problem (Figure 2 (b)). The single sequence is manipulated using an iterated local search over generalized pairwise interchange moves, improved with a suitable tie breaking criterion. Our proposed methods adopt the same data structure and an upgrade of that local search, using additional evolutionary strategies with better results. Figure 2 (c) shows the scheduling representation for identical parallel machines using the Gantt chart. For the weighted tardiness scheduling problem, considering parallel machines, Croce et al. (2012) present a heuristic where computational results are better than the results presented in the literature, for multi-machine instances.

A *Branch-and-Bound* algorithm for the weighted earliness-tardiness scheduling problem can be seen in Sourd and Kedad-Sidhoum (2003), Tanaka et al. (2003) and Rabadi et al. (2004).

3. Proposed Methods

This article proposes four algorithmic strategies, with single-start or multi-start global search optimization, and based on Local Search and Path-Relinking techniques [Glove and Kochenberger (2003)], whose goal is to achieve as good as possible solutions by analysing the convergence process and measuring the execution time performance and the quality of the solutions generated. The first method is an *Iterated Local Search* (ILS) [Rodrigues et al. (2008)], an iterative single-start global search method, where upon each iteration, the current solution is changed by a new better neighboring solution, using a single sequence representation for multi-machine schedules and a 2-opt neighborhood-based with smart GPI moves and tie-breaking criteria. The second algorithmic strategy proposed, ILS+PR method [Amorim et al. (2012)], involves *Iterated Local Search* with *Path-Relinking* (PR) technique, whose goal is to better explore the search space between local optimal solutions.

The third algorithmic strategy proposed combines *Genetic Algorithm* using *Local Search* (LS) with *Path-Relinking*. The GA+LS+PR method, whose goal is to better explore the search space between local optimal solutions, as ILS+PR method does, but now a population is considered, in order to converge more quickly and obtain better solutions, even for larger instances. The fourth and last algorithmic strategy proposed involves ILS with multi-start global search approach, the ILS Multi-start method, where the objective of this strategy is to check if ILS can better explore and achieve better solutions, for larger instances, considering a set of diversified solutions on each iteration. In the following subsections, the proposed methods are detailed.

3.1. Iterated Local Search - ILS

The first method proposed is the *Iterated Local Search* algorithm, which was used for the weighted earliness-tardiness parallel scheduling problem, $P||\sum \alpha_j E_j + \beta_j T_j$, based on the ILS method proposed by Rodrigues et al. (2008). This algorithm is based on the idea of representing the scheduling on multiple machines as a single sequence, which simplifies the treatment of the problem. Figure 2 (b) presents the correlation between a single sequence of jobs (which is also a permutation of the set of jobs) and its corresponding scheduling on parallel machines presented in Figure 2 (c). Algorithm 1 presents the general steps of the heuristic proposed by Rodrigues et al. (2008), where N represents the number of iterations, r represents the number of times that every permutation is generated and k represents the number of *Generalized Pairwise Interchange* (GPI) moves applied to the solution π.

Algorithm 1 Heuristic based on single sequence representation [Rodrigues et al. (2008)].

$i \leftarrow 1$;
$\pi^* \leftarrow$ a permutation following the *Earliest Due Date* (EDD) rule;
while $i < N$ **do**
    if i **is a multiple of** r **then**
        $\pi \leftarrow$ **a random permutation of jobs**;
    end if
    Apply GPI (Generalized Pairwise Interchange) moves in π, until no further improvement is possible;
    if $w(\pi) < w(\pi^*)$ **then**
        $\pi^* \leftarrow \pi$;
    end if
    Apply k GPI movements randomly chosen in π;
    $i \leftarrow i+1$.
end while

Considering n jobs and m machines, let π be a permutation of the n jobs. The algorithm begins its processing by an initial feasible solution generated by the *Earliest Due Date First* (EDD), and the cost of this initial schedule is stored as $w(\pi^*)$, the best value so far. The algorithm performs a local search on a given initial sequence of the n jobs over a neighbourhood defined by some *Generalized Pairwise Interchange* (GPI) moves: the swap of two jobs in the sequence π or the move of a job to another position (the job is removed from the sequence and then inserted in the other position, which corresponds to forward and backward insertion). The change in the sequence is accepted if the cost of the corresponding schedule is lower than the cost of the best current solution (in other words, $w(\pi) < w(\pi^*)$). If the solutions have the same cost, the algorithm runs a tie-breaking criterion defined by $b(\pi) = \sum d_{\pi j}.(n-j+1)$. If $b(\pi) < b(\pi^*)$, then π is considered to improve over π^*. Then, some random moves are applied in order to jump to another region of the search space with a potential better solution in the neighborhood.

This process is repeated during x iterations, where $x = k \cdot n \cdot m$, where k is a given constant. When the number of the current iteration is a multiple of a given constant r, a new random sequence is generated over which the local search will be made.

3.2. Iterated Local Search with Path-Relinking between local optimal solutions - ILS+PR

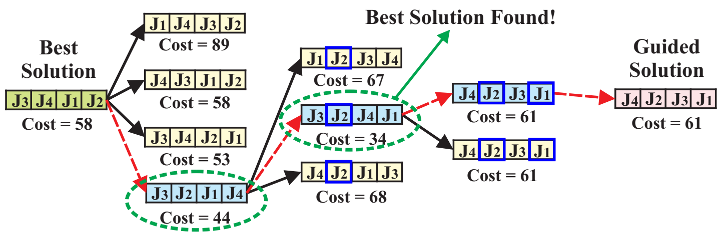
The second method applied to the E/T scheduling problem consists of a combination of the ILS algorithm and the *Path-Relinking* technique between local optimal solutions, which was originally proposed for diversification of the *scatter search algorithm* [Glover et al. (2000)]. Thus, it was possible to converge faster to better solutions than the ILS algorithm (without the Path-Relinking technique) and also to obtain better solutions for larger instances.

The ILS+PR method begins its processing with two feasible solutions, these solutions are generated through a random permutation on single sequences following the EDD rule. Then, the algorithm performs a local search on each feasible solution. After the local search, two local optimal solutions are obtained, and the PR technique is applied in order to better explore the solution space between two local optimal solutions. We prove that there are better solutions in search space between two local optimal solutions.

In the proposed implementation, the method begins its execution from the best local optimal solution, of the iteration, to the worst local optimal solution, this strategy is known as *Backward Path Relinking*. The best solution found in the path is stored and, finally, the four solutions (two solutions obtained by local searches, one obtained by local searches, one solution obtained by Path-Relinking and the best global solution) are compared among themselves. The best solution of these four candidate solutions is considered the best global solution. In the Figure 3 (b), which considers the instance table Figure 3 (a) on two parallel machines, the strategy of the Path-Relinking technique is presented.

J_j	p_j	d_j	α_j	β_j
J_1	4	7	3	5
J_2	3	4	4	5
J_3	5	5	8	8
J_4	9	10	8	10

(a)



(b)

Figure 3. (a) An instance for the earliness-tardiness scheduling problem. (b) Backward Path Relinking strategy, where the solutions with dashed arrows represent the path of best solutions in the search between the initial best solution and guided solution.

In Path-Relinking, the best solution of the two initial (local optimal) solutions found is saved. This solution which will be called *path sequence*, will be changed to become equal to the guide solution. So, the method finds, for each position of the path sequence, a solution that, when changed, becomes equal to the guided solution, thus returning the best solution found. When the best move is found in the path sequence, the position where the value is equal to the position of the guided solution is marked as fixed and will not be changed. This procedure is repeated until all positions become fixed (n iterations). To help the Path-Relinking method to explore even more of the solution space, a perturbation of the solution is performed, on every 5 iterations, in order to always try to go to a new wide area of the solution space which can be explored by these procedures (two local searches and Path-Relinking). The Path-Relinking pseudocode is presented in Algorithm 2.

Algorithm 2 Path-Relinking(best solution, worst solution) [Amorim et al. (2012)].

```
path ← best solution;
guide ← worst solution;
count ← 1;
BestFoundCost ← ∞;
while count ≤ n do
    BestChangeCost ← ∞;
    while j ≤ n do
        if j is not fixed then
            k ← Position of guide_j in path;
            Swap path_j e path_k;
            if w(path) < BestChangeCost then
                BestJ ← j;
                BestK ← k;
                BestChangeCost ← w(caminho);
            end if
            Swap path_k e path_j;
        end if
    end while
    Swap path_BestJ and path_BestK;
    if w(path) < BestFoundCost then
        BestFound ← path;
        BestFoundCost ← w(path);
    end if
    Mark j as Fixed;
end while
return BestFound;
```

3.3. Genetic Algorithm using Local Search with Path-Relinking between local optimal solutions - GA+LS+PR

The third method proposed is the Genetic Algorithm (GA) using Local Search with Path-Relinking between local optimal solutions. This algorithm also uses single sequence representation [Rodrigues et al. (2008)] (Figure 2 (b)), which is treated as a chromosome for genetic operators (*Position-based Crossover* and *Mutation*). Each gene represents one job and on each iteration a new population is generated, when it is necessary to calculate the cost (fitness) of a chromosome (solution), the jobs are distributed on the machines and the value of weighted earliness-tardiness is calculated (Figure 2 (c)). The algorithm was based on genetic operators proposed by Liu et al. (2005) for the weighted tardiness scheduling problem on single machines ($1||\sum w_j T_j$).

A *Position-based Crossover* is used in proposed Genetic Algorithm, where the positions to be fixed are randomly chosen and kept unchanged in the offspring. The Figure 4 (a) shows a *Crossover* example, for the instance presented in the Figure 2 (a), where the parents (individuals A and B) were chosen by *Tournament Selection* strategy, and two offsprings (C and D) were generated by Crossover. A *Tournament Selection* is also used for the *Mutation* operator, but only one individual is selected to receive the *Mutation*, the genes, selected for exchange, are randomly chosen as can be observed in Figure 4 (b). The Genetic Algorithm proposed in this article has an extra step: for each offspring (child) solution generated by *Crossover*, a local search in the GPI neighbourhood is performed. This approach can reduce the time needed to reach an optimal or near-optimal solution in the algorithm (Figure 4 (c)).

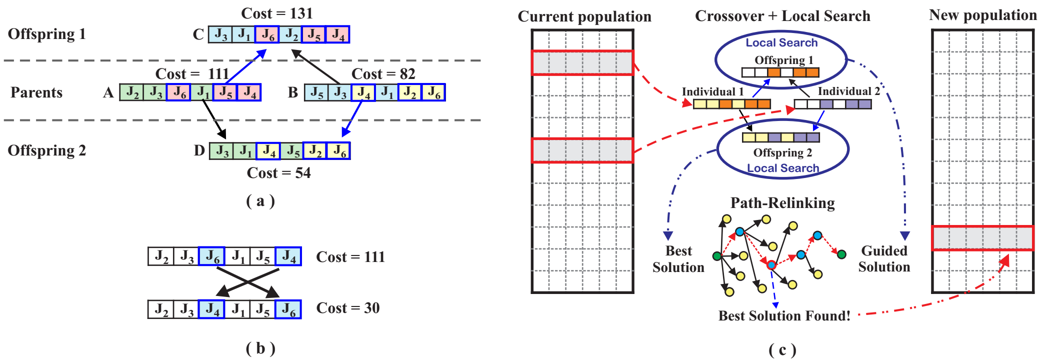


Figure 4. (a) Position-based crossover genetic operator (b) Mutation genetic operator and (c) Scheme of position based crossover followed by local search and path-relinking.

On each iteration of the Genetic Algorithm proposed, a new population with better and diversified solutions is generated to be used in the next iteration of the algorithm. Algorithm 3 presents the general steps of the proposed idea, where on each iteration the population is sorted from the best to the worst solution, but only the best solutions are copied to the new population of the algorithm, ensuring the *Elitism*. Next, a Tournament Selection is performed in order to select some solutions, from the current population, to apply genetic operators such as *Mutation*, *Crossover* and *Crossover* with Local Search. Finally, a Crossover is performed again in order to obtain two new offsprings (local optimal solutions) to be used by the Path-Relinking technique to explore the solution space between them, Figure 4 (c) presents the proposed idea, where the new solution obtained is kept in a new population.

If a best solution is found during those steps, the current best global solution of the algorithm is replaced with the new best solution. Algorithm 3 presents the proposed idea, where N is the number of generations, $\prod$ represent the current population, $\prod^*$ represents the new population which will be used in the next iteration of the algorithm, π and θ represent the solutions chosen from $\prod$ by *Tournament Selection* to be used as parents and π^* and θ^* represents the generated solutions to be copied to the new population $\prod^*$.

Algorithm 3 Hybrid genetic algorithm with local search using the single sequence approach by Rodrigues et al. (2008) and genetic operators by Liu et al. (2005).

$\prod \longleftarrow$ set of randomly generated permutations of jobs;
Calculate the fitness value of each permutation;
while $i < N$ **do**
  $\prod^* \longleftarrow$ best solutions from $\prod$ {Elitism};
  $\prod^* \longleftarrow$ *Mutation* on solution π;
  $\prod^* \longleftarrow$ *Crossover* using solutions π and θ;
  $\prod^* \longleftarrow$ *Crossover with Local Search* using solutions π and θ;
  π^* and $\theta^* \longleftarrow$ *Position-based Crossover* on π and θ {Apply Crossover using Local Search with Path-Relinking};
  for each offspring π^* and θ^* **do**
    Apply GPI moves until no improvement is possible;
  end for
  if $w(\pi^*) < w(\theta^*)$ **then**
    $\prod^* \longleftarrow$ **Path-Relinking**(π^*, θ^*);
  else
    $\prod^* \longleftarrow$ **Path-Relinking**(θ^*, π^*);
  end if
  $i \longleftarrow i+1$;
end while
Return the best solution found.

3.4. ILS Multi-start

The fourth and last method proposed involves a multi-start global search method with Local Search as improvement procedure. The difference between this algorithm and the ILS algorithm (presented in the Section 3.1) is that this method considers a set of initial solutions from the solution space, which are randomly generated by a permutation of jobs. On every iteration of the Algorithm two steps are considered: the first step is dedicated to build new solutions by applying perturbations on some randomly chosen solutions of the set in order to explore the widest range of the solution space, the second step is dedicated to improving the solutions through a Local Search on a few randomly chosen solutions in the solutions set. Through this approach it is possible to achieve the best solutions in a reasonable execution time, better than the other three methods proposed. Algorithm 4 presents the proposed strategy, where x and y are constants, N is the number of iterations and $\prod$ is the set of solutions used on each iteration.

Algorithm 4 ILS Multi-start.

```
1: i ← 1;
2: ∏ ⟵ a set of randomly generated solutions;
3: Calculate the objective function value of every solution in the set;
4: while i < N do
5:     ∏ ⟵ random permutation on x randomly selected solutions;
6:     ∏ ⟵ GPI moves on y randomly selected solutions until no improvement is possible;
7:     i ← i+1;
8: end while
9: Return the best solution found.
```

4. Computational Experiments

The single-start methods were tested with the following configuration: $30nm$ iterations for the ILS method and $25nm$ iterations for the ILS+PR method. The ILS multi-start method was tested with a set of $12n$ solutions and 60 iterations, where n is the number of jobs and m is the number of machines.

The proposed Genetic Algorithm was tested with $2nm$ generations and with a population size of $2nm$, the iterations/generations was divided in two parts: in the first part ($1/3$ of iterations), the new generated populations are based on 5% of Elitism, 50% of Mutation, 40% of Crossover and 15% of Crossover with Local Search. The objective of the first part is to obtain the widest range of possible solutions, increasing the diversity of individuals. In the second part ($2/3$ of iterations), the new generated populations are based on 10% of Elitism, 30% of Mutation, 10% of Crossover, 20% of Crossover with Local Search and 30% of Crossover using Local Search with Path-Relinking. This second configuration permits finding optimal or near-optimal solutions in the population created by the $1/3$ iterations. In the next $2/3$ iterations, the algorithm is dedicated to finding better solutions and generating new populations with increasingly better solutions, but nothing prevents optimal solutions to be found in the first part.

Instances from literature were used in order to test the performance of the proposed algorithms, where single machine instances for the problem $1||\sum\alpha_jE_j + \sum\beta_jT_j$, proposed by Tanaka (2012) [1] were used. The results are presented in Table 1, achieving all optimal solutions, except for 300 jobs. These instances had to be adapted in order to be tested on identical parallel machines ($P \mid\mid \sum\alpha_jE_j + \sum\beta_jT_j$), this adaptation was made by dividing the due dates by the number of machines. Computational experiments were also made on these adapted instances and reported in Tables 2, 3 and 4, where is possible to observe that the GA+LS+PR and ILS Multi-start algorithms proved to be the best methods proposed, achieving optimal or near-optimal solutions in most of the cases tested, even for bigger instances.

The optimal solutions presented for multi-machines was obtained by using the *Branch-and-Cut* algorithm from the IBM/ILOG CPLEX 12.4 solver, where our proposed formulation was implemented by using the UFFLP library [Pessoa and Barboza (2011)] with C++ where, given an instance for the problem, it is possible to generate the mathematical model to be executed in CPLEX. The following columns present the best solution obtained (*Best Solution*), the average of three executions for every instance (*Average Solution*), the quantity of iterations (*Iterations*), the best times in seconds to reach the best solutions presented (*Best Time*) and the *Total Time* in seconds of the algorithm. In all four Tables, the proposed methods are identified by the number of their respective

[1] http://turbine.kuee.kyoto-u.ac.jp/∼tanaka/SiPS/index.html

sections. Thus, the ILS method is identified by 3.1, the ILS+PR method is identified by 3.2, the GA+LS+PR method is identified by 3.3 and finally, the ILS Multi-start method is identified by 3.4.

Table 1. Algorithm Results for 40, 50, 100, 150, 200 and 300 jobs on single machine.

J_i	Inst.	Tanaka	Best Solution				Average Solution				Iterations/Generations				Best Time				Total Time			
			3.1	3.2	3.3	3.4	3.1	3.2	3.3	3.4	3.1	3.2	3.3	3.4	3.1	3.2	3.3	3.4	3.1	3.2	3.3	3.4
	1	54640	54640	54640	54640	54640	54640.0	54640.0	54640.0	54640.0	7	1	1	1	0.0	0.0	0.0	0.0	3.0	5.7	18.3	10.8
	11	29924	29924	29924	29924	29924	29924.0	29924.0	29924.0	29924.0	1	3	1	1	0.0	0.0	0.0	0.0	2.7	5.4	7.9	12.4
	21	77819	77819	77819	77819	77819	77819.0	77819.0	77819.0	77819.0	3	1	1	1	0.0	0.0	0.0	0.0	2.3	4.2	11.2	12.0
	31	23291	23291	23291	23291	23291	23291.0	23291.0	23291.0	23291.0	1	1	1	1	0.0	0.0	0.0	0.0	3.3	5.9	12.2	11.2
	41	59093	59093	59093	59093	59093	59093.0	59093.0	59093.0	59093.0	3	1	1	1	0.0	0.0	0.0	0.0	3.1	6.1	13.2	10.6
	51	47667	47667	47667	47667	47667	47667.0	47667.0	47667.0	47667.0	2	6	1	1	0.0	0.1	0.0	0.1	3.1	6.2	8.2	11.4
40	61	21737	21737	21737	21737	21737	21737.0	21737.0	21737.0	21737.0	11	1	1	1	0.0	0.0	0.0	0.0	3.6	6.9	14.0	12.3
	71	90521	90521	90521	90521	90521	90521.0	90521.0	90521.0	90521.0	1	1	1	1	0.0	0.0	0.0	0.0	3.0	5.6	18.5	10.3
	81	12552	12552	12552	12552	12552	12552.0	12552.0	12575.0	12552.0	92	32	1	1	0.3	0.2	0.0	0.2	3.4	6.6	12.6	12.7
	91	48311	48311	48311	48311	48311	48311.0	48311.0	48311.0	48311.0	1	1	1	1	0.0	0.0	0.0	0.0	3.4	6.6	6.6	11.5
	101	85961	85961	85961	85961	85961	85961.0	85961.0	85961.0	85961.0	4	5	1	1	0.1	0.1	0.0	0.1	3.8	7.2	6.6	11.3
	111	32374	32374	32374	32374	32374	32374.0	32374.0	32377.3	32374.0	5	2	1	1	0.0	0.0	0.0	0.0	3.5	6.9	6.7	11.9
	121	122538	122538	122538	122538	122538	122538.0	122538.0	122540.3	122538.0	40	9	1	1	0.1	0.1	0.0	0.1	3.3	6.5	6.7	11.2
	1	130548	130548	130548	130548	130548	130548.0	130548.0	130570.3	130548.0	11	15	1	1	0.1	0.2	0.1	0.0	6.5	12.7	17.6	23.6
	11	54167	54167	54167	54167	54167	54167.0	54167.0	54167.0	54167.0	3	19	1	1	0.0	0.2	0.0	0.2	0.0	12.2	17.7	23.9
	21	214555	214555	214555	214555	214555	214555.0	214555.0	214555.0	214555.0	5	2	1	1	0.0	0.0	0.0	0.0	5.9	11.2	16.6	21.4
	31	37565	37565	37565	37565	37565	37565.0	37565.0	37573.7	37565.0	5	21	1	1	0.0	0.3	0.1	0.0	7.5	14.4	18.7	25.6
	41	71949	71949	71949	71949	71949	71949.0	71949.0	71949.0	71949.0	4	1	1	1	0.0	0.0	0.0	0.0	6.8	12.9	18.2	24.1
	51	56208	56208	56208	56208	56208	56208.0	56208.0	56208.0	56208.0	1	1	1	1	0.0	0.0	0.0	0.0	8.6	15.8	20.2	31.7
50	61	34640	34640	34640	34660	34640	34640.0	34640.0	34676.7	34640.0	121	88	1	1	0.7	1.2	0.1	0.6	8.4	16.3	19.8	28.0
	71	150978	150978	150978	150978	150978	150978.0	150978.0	150978.0	150978.0	36	16	1	1	0.2	0.2	0.0	0.0	7.0	14.1	17.6	23.9
	81	17433	17433	17433	17433	17433	17433.0	17433.0	17433.0	17433.0	13	13	1	1	0.1	0.2	0.0	0.0	8.8	16.7	22.1	28.8
	91	89715	89715	89715	89715	89715	89715.0	89715.0	89715.0	89715.0	31	8	1	1	0.2	0.1	0.0	0.0	8.3	16.1	18.8	26.6
	101	90880	90880	90880	90880	90880	90880.0	90880.0	90883.7	90880.0	38	4	1	1	0.2	0.1	0.1	0.1	8.5	15.8	18.8	28.5
	111	30170	30170	30170	30170	30170	30170.0	30170.0	30170.0	30170.0	8	8	1	1	0.0	0.1	0.1	0.0	8.1	15.8	19.7	27.9
	121	79007	79007	79007	79007	79007	79007.0	79007.0	79007.0	79007.0	2	1	1	1	0.0	0.0	0.0	0.0	7.1	13.8	18.1	25.1
	1	405122	405122	405122	405122	405122	405122.0	405122.0	405122.0	405122.0	707	41	1	1	24.7	3.7	1.7	11.0	107.6	201.5	535.1	372.7
	11	252623	252623	252623	252623	252623	252633.3	252623.0	252637.7	252623.0	1858	321	1	1	66.1	26.2	0.7	2.5	109.1	204.9	528.9	360.5
	21	898927	898927	898927	898927	898927	898927.0	898927.0	898927.0	898927.0	1	1	1	1	0.1	0.1	0.1	0.1	85.6	151.6	481.4	307.6
	31	193480	193480	193480	193480	193480	193480.0	193480.0	193525.3	193480.0	69	50	2	1	3.2	4.8	5.0	3.6	124.5	227.7	568.3	370.9
	41	463554	463554	463554	463554	463554	463554.0	463554.0	463556.3	463554.0	593	56	1	1	21.5	5.8	0.3	1.7	107.8	226.7	534.2	355.2
	51	444991	444991	444991	444991	444991	444991.0	444991.0	444991.0	444991.0	2	15	1	1	0.1	1.5	0.5	0.4	123.5	233.1	549.0	388.9
100	61	102185	102185	102185	102185	102185	102185.0	102185.0	102187.0	102185.0	568	61	2	1	23.8	6.4	4.7	11.7	128.7	245.9	602.9	404.4
	71	640932	640932	640932	640932	640932	640932.0	640932.0	640932.0	640932.0	342	72	1	1	11.8	5.7	0.1	1.1	101.9	197.4	523.7	392.4
	81	47629	47629	47629	47629	47629	47629.0	47629.0	47680.0	47629.0	252	24	2	1	11.1	2.3	5.3	3.7	125.8	255.9	630.2	467.8
	91	249743	249743	249743	249743	249743	249743.0	249743.0	249743.0	249743.0	672	374	2	1	24.8	32.4	4.1	3.2	112.4	219.3	554.6	391.0
	101	356397	356397	356397	356397	356397	356397.0	356397.0	356397.0	356397.0	15	4	1	1	0.7	0.5	0.2	0.3	129.4	239.9	550.2	373.3
	111	161642	161653	161680	161691	**161642**	161669.7	161682.0	161691.0	161674.7	109	883	4	8	5.2	97.2	10.6	91.7	147.3	274.8	639.0	460.3
	121	471761	471768	**471761**	**471761**	**471761**	471768.0	471761.0	471766.0	471761.0	2863	825	3	1	151.4	74.9	8.0	4.6	158.8	230.4	528.5	428.8
	1	855097	855097	855097	855097	855097	855097.0	855097.0	855097.0	855097.0	56	27	2	1	12.9	7.8	31.6	31.5	822.4	1012.9	4245.7	2475.5
	31	378921	378921	378921	378921	378921	378924.3	378924.3	378921.0	378921.0	2065	1012	2	2	406.7	293.1	28.2	151.8	867.1	1100.6	4376.9	2684.3
150	61	235255	235358	235275	235752	**235255**	235451.0	235291.7	235752.0	235255.0	1230	1397	4	6	273.3	460.9	60.1	535.1	997.3	1237.0	4572.4	3129.7
	91	416842	416842	416842	416842	416842	416842.0	416842.0	416842.0	416842.0	77	65	3	1	17.9	21.2	45.9	36.4	973.1	1224.3	4503.5	2949.6
	121	629821	629821	629821	629821	629821	629821.0	629821.0	629821.0	629821.0	44	39	1	1	8.9	10.8	3.8	12.0	892.7	1075.5	4448.2	2742.7
200	1	1508466	1508473	1508473	1508674	**1508466**	1508473.0	1508475.6	1508674.0	1508479.0	1863	2959	4	3	1463.6	5184.7	153.5	646.6	4709.0	8764.2	17461.8	7468.9
300	1	3184308	3184669	3184707	3185010	3184598	3184883.8	3184781.2	3185010.0	3184606.0	1386	20	6	8	4045	127.6	2811.9	10995.2	26655.7	45057.7	135907.7	43990.6

5. Concluding Remarks

This work considered the earliness-tardiness scheduling problem on single and identical parallel machines, with arbitrary processing times and independent weighted jobs ($P \,||\, \sum \alpha_j E_j + \sum \beta_j T_j$ in 3-field notation). We proposed four algorithmic strategies involving Local Search and Path-Relinking techniques, with single-start and multi-start global search optimization approaches. A comparative empirical analysis showed that ILS Multi-start global optimization was the best method proposed, achieving all optimal solutions for single machine Tanaka´s instances (obtained by his IP exact method).

An integer mathematical formulation, based on the classical time-indexed formulation and the network flow model for parallel machines without idle time, is also presented for the problem. The proposed algorithms were able to achieve optimal solutions for single machine Tanaka (2012) instances. The tests were also performed on multi-machine instances, achieving good solutions in a reasonable execution time in most cases tested, but there is no available benchmark in the literature for the scheduling problem with earliness-tardiness on parallel machines. Therefore, we tested our proposed formulation by running it at IBM/ILOG CPLEX in order to compare the results with the proposed methods in this article, they achieved optimal solutions in most cases tested.

Acknowledgments

This work is partially supported by the Brazilian agencies: CNPq ("Conselho Nacional de Desenvolvimento Científico e Tecnológico"), and CAPES ("Coordenação de Aperfeiçoamento de Pessoal de Nível Superior").

Table 2. Algorithm Results for 40, 50, 100, 150 **and** 200 **jobs on** 2 **machines.**

J_i	Inst.	WETSF	Best Solution				Average Solution				Iterations/Generations				Best Time				Total Time				
			3.1	3.2	3.3	3.4	3.1	3.2	3.3	3.4	3.1	3.2	3.3	3.4	3.1	3.2	3.3	3.4	3.1	3.2	3.3	3.4	WETSF
40	1	26063	**26063**	**26063**	**26063**	**26063**	26063.0	26063.0	26064.0	26063.0	288	182	4	4	1.1	1.4	0.8	1.6	8.4	15.4	33.3	13.4	1302.87
	11	15451	**15451**	**15451**	**15451**	**15451**	15451.0	15451.0	15451.0	15451.0	12	8	1	1	0.1	0.1	0.1	0.1	7.5	13.9	31.5	13.3	157.71
	21	41054	**41054**	**41054**	**41054**	**41054**	41054.0	41054.0	41054.0	41054.0	2	4	1	1	0.0	0.0	0.0	0.0	7.1	13.1	33.6	13.2	3.85
	31	11679	**11679**	**11679**	**11679**	**11679**	11679.0	11679.0	11679.0	11679.0	3	2	1	1	0.0	0.0	0.0	0.0	8.1	14.9	32.8	13.1	17.61
	41	31678	**31678**	**31678**	**31678**	**31678**	31678.0	31678.0	31678.0	31678.0	5	2	1	1	0.0	0.0	0.0	0.0	7.6	13.6	32.1	12.7	10.19
	51	20534	21709	21709	21709	21709	21709.0	21709.0	21709.0	21709.0	20	3	1	1	0.1	0.0	0.0	0.3	8.6	16.3	31.5	13.3	9834.40
	61	12472	**12472**	**12472**	**12472**	**12472**	12472.0	12472.0	12478.0	12472.0	683	278	2	2	2.6	2.4	0.4	0.8	9.1	16.4	33.7	14.1	341.65
	71	47952	**47952**	**47952**	**47952**	**47952**	47952.0	47952.0	47952.0	47952.0	7	1	1	1	0.0	0.0	0.0	0.1	7.8	14.5	34.8	12.9	3.76
	81	5278	**5278**	**5278**	**5278**	**5278**	5278.0	5278.0	5278.7	5278.0	516	3	3	1	2.0	0.0	0.5	0.1	9.1	17.0	37.4	16.2	388.32
	91	26309	**26309**	**26309**	**26309**	**26309**	26309.0	26309.0	26309.0	26309.0	53	17	1	1	0.2	0.1	0.1	0.3	8.7	16.8	34.7	13.9	16.23
	101	–	40300	40300	40300	40300	40300.0	40300.0	40305.0	40300.0	263	34	1	1	1.0	0.3	0.3	0.5	8.8	16.9	32.4	13.7	–
	111	18493	**18493**	**18493**	**18493**	**18493**	18493.0	18493.0	18493.0	18493.0	28	5	1	1	0.1	0.0	0.1	0.1	8.8	17.9	34.3	14.0	93.03
	121	64584	**64584**	**64584**	**64584**	**64584**	64584.0	64584.0	64584.0	64584.0	96	23	1	1	0.3	0.2	0.1	0.1	8.2	15.1	33.2	13.6	13.95
50	1	62985	62989	**62985**	**62985**	**62985**	62990.3	62985.0	62989.7	62985.3	856	840	3	5	5.9	12.3	1.5	4.1	20.4	37.0	94.3	32.7	743.36
	11	27871	**27871**	**27871**	**27871**	**27871**	27871.0	27871.0	27871.0	27871.0	55	39	2	1	0.4	0.6	0.7	0.5	19.5	36.6	93.3	30.4	853.47
	21	111069	**111069**	**111069**	**111069**	**111069**	111069.0	111069.0	111069.0	111069.0	46	1	1	1	0.2	0.0	0.0	0.4	16.3	30.2	87.0	27.2	19.41
	31	18266	**18266**	**18266**	**18266**	**18266**	18266.0	18266.0	18266.0	18266.0	79	292	2	1	0.6	6.7	1.0	0.5	21.4	57.1	99.9	40.2	1966.51
	41	38491	**38491**	**38491**	**38491**	**38491**	38491.0	38491.0	38523.0	38491.0	416	23	2	1	2.4	0.4	0.6	0.4	17.1	47.5	97.7	30.3	131.37
	51	–	26152	26152	26152	26152	26152.0	26152.0	26152.0	26152.0	2044	1634	3	2	15.4	37.6	1.6	2.6	22.7	54.4	105.1	34.8	–
	61	17456	**17456**	**17456**	**17456**	**17456**	17456.0	17456.0	17480.0	17456.0	559	132	2	2	4.0	2.6	0.8	2.2	20.8	45.5	100.2	32.3	94.8
	71	78612	**78612**	**78612**	**78612**	**78612**	78612.0	78612.0	78612.0	78612.0	13	67	1	1	0.1	1.1	0.1	0.0	18.5	32.5	90.4	28.9	14.1
	81	–	7903	7903	7903	7903	7903.0	7903.0	7903.0	7903.0	92	24	1	1	0.7	0.4	0.3	0.4	23.0	42.2	109.8	33.6	–
	91	47513	**47513**	**47513**	**47513**	**47513**	47513.0	47513.0	47515.7	47513.0	913	9	2	2	7.3	0.2	0.8	1.7	20.5	37.4	98.4	31.8	662.2
	101	40623	42973	42973	42973	42973	42973.0	42973.0	42973.0	42973.0	429	71	2	3	3.3	1.2	1.2	2.9	23.4	40.1	98.6	34.1	9916.9
	111	17012	**17012**	**17012**	**17012**	**17012**	17012.0	17012.0	17012.0	17012.0	799	69	2	2	6.0	1.0	1.2	1.7	22.1	37.4	116.8	35.2	7424.6
	121	41989	**41989**	**41989**	**41989**	**41989**	41989.0	41989.0	41989.0	41989.0	3	14	1	1	0.0	0.2	0.1	0.1	21.9	36.7	96.4	29.5	547.5
100	1	–	198288	198291	198282	**198281**	198289.7	198292.3	198302.7	198281.7	4056	1233	9	7	223.4	143.0	63.7	102.1	330.6	575.7	2959.7	658.1	–
	11	–	127682	127676	**127666**	**127666**	127682.0	127678.3	127685.0	127666.0	1338	1187	6	4	66.9	123.6	42.0	63.3	299.4	562.1	2957.8	464.9	–
	21	–	457865	457865	457865	457865	457865.0	457865.0	457865.0	457865.0	18	29	1	1	0.8	2.5	1.5	0.5	240.4	417.9	2592.1	397.8	–
	31	–	95053	95051	**95038**	**95038**	95088.7	95051.0	95076.3	95038.0	5699	4805	8	6	357.2	607.7	73.1	107.0	376.2	632.0	3811.4	636.6	–
	41	–	238416	238416	238416	238416	238416.0	238416.0	238416.0	238416.0	4817	265	1		250.0	28.6	3.1	2	320.4	531.1	2953.4	487.0	–
	51	–	215500	**215498**	**215498**	**215498**	215501.3	215501.3	215498.0	215498.0	85	4614	5	4	6.1	595.4	36.4	69.2	383.3	645.5	2902.9	502.7	–
	61	–	52740	**52739**	52740	52740	52755.0	52749.0	52740.0	52740.0	467	4748	2	3	29.2	563.1	19.1	45.3	367.1	593.1	3090.3	491.4	–
	71	–	327358	327358	327358	327358	327360.7	327359.0	327359.0	327358.7	1568	2196	3	6	87.8	233.7	20.2	79.8	361.2	531.7	3311.1	531.1	–
	81	–	22740	22755	**22720**	**22720**	22773.3	22761.3	22730.7	22720.0	3653	1177	6	8	230.1	164.3	54.7	141.9	382.6	670.7	3393.8	616.8	–
	91	–	130168	130169	**130165**	**130165**	130172.0	130169.0	130165.0	130165.0	3482	979	5	8	204.3	118.6	37.9	115.8	347.9	677.0	2921.0	534.2	–
	101	–	172816	172815	**172796**	**172796**	172816.7	172819.7	172796.0	172799.7	5740	4330	6	7	344.1	524.6	48.0	107.6	359.5	604.8	3527.3	541.9	–
	111	–	85172	85164	**85160**	**85160**	85172.0	85164.7	85181.3	85161.3	2529	1068	7	8	163.7	137.6	71.6	138.0	389.1	642.7	3440.4	586.1	–
	121	–	242250	242250	**242234**	**242234**	242253.0	242252.7	242234.3	242234.0	4413	1988	4	5	253.3	228.8	29.5	75.1	334.3	576.5	2844.1	489.8	–
150	1	–	421150	421145	**421141**	**421141**	421150.0	421150.0	421141.0	421141.0	6738	4246	5	4	3160.5	4256.9	226.4	501.8	4215.8	7532.5	33043.1	13272.4	–
200	1	–	746289	746246	**746190**	**746190**	746296.3	746246.0	746190.0	746190.0	2680	999	13	14	3164.7	2694.7	1464.3	6215.5	14189.9	26843.6	45281.7	17394.9	–

References

Amorim, R. X. d., Dias, B. R. C., and Rodrigues, R. d. F. (2012). ILS com reconexão de caminhos entre ótimos locais para um problema clássico de escalonamento com antecipação e atraso. *Anais do XVI CLAIO - Congreso Latino-iberoamericano de Investigación Operativa / XLIV SBPO - Simpósio Brasileiro de Pesquisa Operacional*, 1:1–12.

Baker, K. R. and Scudder, G. D. (1990). Sequencing with earliness and tardiness penalties: a review. *Oper. Res.*, 38:22–36.

Brucker, P. (2006). Scheduling algorithms. *Springer Publishing Company, Incorporated*, 5a ed.:1–104.

Croce, F. D., Garaix, T., and Grosso, A. (2012). Iterated local search and very large neighborhoods for the parallel-machines total tardiness problem. *Computers & Operations Research*, 39:1213–1217.

Dyer, M. E. and Wolsey, L. A. (1990). Formulating the single machine sequencing problem with release dates as a mixed integer program. *Discrete Appl. Math.*, 26(2-3):255–270.

Glove, F. and Kochenberger, G. e. (2003). *Handbook of Metaheuristics*. Kluwer Academic Publishers.

Glover, F., Laguna, M., and Martí, R. (2000). Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 39:653–684.

Gordon, V., Proth, J.-M., and Chu, C. (2002). A survey of the state-of-the-art of common due date assignment and scheduling research. *European Journal of Operational Research*, 139:1–25.

Graham, R. L., Lawler, E. L., Lenstra, J. K., and Kan, A. H. G. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5:287–326.

Leung, J. Y.-T. and Anderson, J. H. (2004). *Handbook of scheduling: algorithms, models and performance analysis*. Chapman and Hall/CRC.

Liu, N., Abdelrahman, M., and Ramaswamy, S. (2005). A Genetic Algorithm for Single Machine Total Weighted Tardiness Scheduling Problem. *International Journal of Intelligent Control and Systems*, 10:218–225.

Pessoa, A., Uchoa, E., de Aragão, M. P., and Rodrigues, R. (2010). Exact algorithm over an arc-time-indexed formulation for parallel machine scheduling problems. *Mathematical Programming Computation*, 2:259–290.

Table 3. Algorithm Results for 40, 50 **and** 100 **jobs on** 4 **machines.**

J_i	Inst.	WETSF	Best Solution				Average Solution				Iterations/Generations				Best Time				Total Time				
			3.1	3.2	3.3	3.4	3.1	3.2	3.3	3.4	3.1	3.2	3.3	3.4	3.1	3.2	3.3	3.4	3.1	3.2	3.3	3.4	WETSF
40	1	11985	**11985**	**11985**	**11985**	**11985**	11987.7	11985.0	11986.7	11986.7	478	60	2	2	6.1	1.6	0.9	6.2	60.9	97.4	187.6	63.7	126.67
	11	8206	**8206**	**8206**	**8206**	**8206**	8206.0	8206.0	8206.0	8206.0	22	3	1	1	0.2	0.1	0.1	0.1	32.7	61.3	212.3	40.1	5.89
	21	22793	**22793**	**22793**	**22793**	**22793**	22793.0	22793.0	22793.0	22793.0	14	4	1	1	0.2	0.2	0.0	0.1	33.6	62.5	183.8	42.9	1.20
	31	5950	**5950**	**5950**	**5950**	**5950**	5950.0	5950.0	5950.0	5950.0	18	9	1	1	0.2	0.2	0.1		35.7	64.0	233.4	42.3	4.20
	41	18020	**18020**	**18020**	**18020**	**18020**	18020.0	18020.0	18020.0	18020.0	40	2	1	1	0.3	0.0	0.0	0.1	33.5	59.8	192.3	42.3	5.34
	51	8360	9104	9104	9104	9104	9104.0	9104.0	9104.0	9104.0	38	53	1	1	0.4	0.9	0.1	0.8	39.9	72.4	200.1	30.7	306.67
	61	7714	**7714**	**7714**	**7714**	**7714**	7714.0	7714.0	7714.0	7714.0	42	5	1	1	0.5	0.2	0.2	0.8	37.9	69.7	196.3	30.3	9.36
	71	26740	**26740**	**26740**	**26740**	**26740**	26740.0	26740.0	26740.0	26740.0	43	1	1	1	0.3	0.0	0.1	0.1	33.1	59.6	178.4	25.8	0.44
	81	2181	**2181**	**2181**	**2181**	**2181**	2181.0	2181.0	2181.0	2181.0	385	859	2	2	3.2	15.7	0.9	1.9	40.3	71.9	209.6	30.5	3.51
	91	15678	**15678**	**15678**	**15678**	**15678**	15678.0	15678.0	15685.0	15678.0	210	144	1	1	1.8	2.7	0.4	0.3	39.7	69.8	198.4	31.0	6.87
	101	8782	17819	17819	17819	17819	17819.0	17819.0	17819.0	17819.0	365	160	2	2	3.0	2.9	1.2	1.4	39.0	69.4	193.4	29.3	844.72
	111	11505	**11505**	**11505**	**11505**	**11505**	11505.0	11505.0	11505.0	11505.0	23	9	1	2	0.3	0.2	0.7	1.3	38.8	67.9	195.4	28.7	6.82
	121	35783	**35783**	**35783**	**35783**	**35783**	35783.0	35783.0	35783.0	35783.0	44	8	1	1	0.4	0.2	0.2	0.0	36.3	64.0	225.2	35.5	3.28
50	1	29222	29243	29241	29229	29229	29246.3	29243.7	29230.7	29230.0	2312	3193	7	5	37.1	108.7	7.7	10.5	96.4	170.4	548.8	81.4	229237.49
	11	14828	**14828**	**14828**	**14828**	**14828**	14828.0	14828.0	14828.0	14828.0	270	167	2	1	4.1	5.0	1.8	1.6	84.8	151.3	474.1	67.5	43.77
	21	59441	**59441**	**59441**	**59441**	**59441**	59441.0	59441.0	59441.0	59441.0	30	16	1	1	0.4	0.6	0.2	0.1	80.9	145.1	419.5	62.0	2.25
	31	8709	**8709**	**8709**	**8709**	**8709**	8709.0	8709.0	8709.0	8709.0	4846	750	6	4	81.6	27.1	6.4	9.2	99.8	179.1	506.8	76.7	66.79
	41	22009	**22009**	**22009**	**22009**	**22009**	22009.0	22009.0	22009.0	22009.0	90	12	1	1	1.4	0.4	0.1	0.1	82.9	148.7	474.9	72.5	18.39
	51	–	11377	11377	11380	11377	11377.0	11378.0	11380.0	11377.0	908	738	5	2	16.7	28.4	6.1	5.9	107.4	191.0	559.6	77.5	–
	61	9376	**9376**	**9376**	**9376**	**9376**	9376.0	9376.0	9376.0	9376.0	69	25	2	2	1.3	0.9	2.3	3.5	97.9	167.6	586.0	75.1	15.76
	71	42660	**42660**	**42660**	**42660**	**42660**	42660.0	42660.0	42660.0	42660.0	17	3	1	1	0.3	0.1	0.6	0.1	85.0	145.5	560.7	69.8	8.99
	81	3835	**3835**	**3835**	**3835**	**3835**	3835.0	3835.0	3835.0	3835.0	1577	1253	3	2	27.1	45.5	4.3	5.9	101.9	180.1	631.1	78.7	592.86
	91	26659	**26659**	**26659**	26661	**26659**	26659.0	26659.0	26661.0	26659.0	992	170	3	2	16.3	5.5	5.1	2.8	91.6	161.6	576.4	70.5	29.99
	101	10669	19234	19205	19205	19205	19234.0	19213.7	19205.0	19211.0	1380	1268	3	4	24.0	45.8	4.7	10.7	104.4	181.2	609.7	79.0	959.31
	111	10694	**10694**	**10694**	10714	**10694**	10694.0	10694.0	10714.0	10694.0	5235	957	2	2	92.4	35.1	2.9	4.0	101.3	181.5	645.0	80.1	88.97
	121	23884	**23884**	**23884**	23886	23885	23884.7	23884.0	23886.0	23885.7	4241	2115	3	6	74.8	68.9	4.1	10.0	106.4	163.6	591.4	74.6	40.79
100	1	–	95013	95027	94963	**94951**	95013.0	95027.0	94969.7	94961.0	1574	1972	14	21	270.8	673.7	336.0	1520.1	1971.5	3407.5	20336.3	4685.8	–
	11	–	65779	65756	**65719**	**65719**	65779.0	65771.3	65719.0	65719.0	2599	9252	7	9	369.0	2739.0	168.8	314.3	1668.9	2956.3	16975.2	1282.1	–
	21	–	**237398**	**237398**	237399	237399	237398.0	237398.3	237399.0	237399.0	85	2006	2	1	9.4	464.9	31.4	17.2	1275.2	2296.5	19024.0	1115.4	–
	31	–	45797	45738	45656	**45644**	45798.3	45738.0	45656.0	45692.3	10431	4110	17	16	1717.3	1409.5	473.9	1149.6	1949.3	3396.2	19761.5	1974.6	–
	41	–	126409	**126408**	**126408**	**126408**	126409.0	126408.0	126408.0	126408.0	8829	8079	1	4	1080.5	2150.7	15.7	146.8	1447.4	2661.6	21013.6	1225.3	–
	51	–	100649	100643	100585	**100581**	100662.7	100648.3	100585.0	100582.3	2754	6069	16	16	429.4	1984.5	424.4	965.0	1861.0	3262.1	20247.0	1874.9	–
	61	–	**28273**	28293	28289	28274	28296.7	28295.7	28289.0	28277.0	6907	6885	9	6	1049.6	2234.9	232.3	276.7	1820.4	3246.7	19940.2	1499.1	–
	71	–	170705	170704	170700	**170696**	170715.0	170705.7	170700.0	170699.0	8168	304	13	15	980.5	79.8	405.2	730.3	1442.4	2602.5	19954.7	1494.0	–
	81	–	10571	10590	**10484**	10494	10598.7	10604.7	10484.0	10504.7	10843	1934	9	19	1798.2	681.7	329.9	1375.7	2000.8	3502.5	28502.3	2010.6	–
	91	–	70781	70770	**70755**	**70755**	70781.0	70774.0	70755.0	70755.0	6290	2019	7	5	884.9	600.0	308.9	193.8	1688.0	2994.3	19157.5	1374.5	–
	101	–	81206	81186	81086	**81082**	81221.0	81187.3	81086.0	81085.3	7869	2690	12	15	1161.5	827.5	328.6	908.6	1781.7	3088.2	18936.0	1659.1	–
	111	–	47416	47372	47324	**47323**	47416.0	47372.0	47324.0	47328.7	1007	3767	11	20	165.9	1314.2	324.8	1743.2	2001.5	3495.8	47324.0	2449.3	–
	121	–	127706	127747	127700	**127699**	127722.7	127747.0	127700.0	127699.0	11978	7660	14	13	1681.7	2265.0	315.7	688.2	1684.5	2958.5	17600.1	1530.2	–

Table 4. Algorithm Results for 40, 50 **and** 100 **jobs on** 10 **machines.**

J_i	Inst.	WETSF	Best Solution				Average Solution				Iterations/Generations				Best Time				Total Time				
			3.1	3.2	3.3	3.4	3.1	3.2	3.3	3.4	3.1	3.2	3.3	3.4	3.1	3.2	3.3	3.4	3.1	3.2	3.3	3.4	WETSF
40	1	3988	3988	3988	3988	3988	3988.0	3988.0	3988.0	3988.0	49	4	2	1	0.7	0.1	4.5	0.5	195.9	333.0	4229.0	47.5	0.14
	31	3558	3558	3558	3558	3558	3558.0	3558.0	3558.0	3558.0	8	7	1	1	0.1	0.2	4.9	0.0	200.1	337.4	5082.1	47.3	0.38
	61	5985	5985	5985	5985	5985	5985.0	5985.0	5985.0	5985.0	4	1	1	1	0.1	0.0	5.9	0.0	194.3	325.1	4749.9	47.5	0.40
	91	9818	9818	9818	9818	9818	9818.0	9818.0	9818.0	9818.0	41	76	1	1	0.6	2.5	8.6	0.0	191.0	320.4	4849.7	47.5	0.89
	121	18818	18818	18818	18818	18818	18818.0	18818.0	18818.0	18818.0	74	2	1	1	1.3	0.1	4.0	0.0	186.8	312.0	4991.4	44.9	0.67
50	1	9006	9171	9168	9154	9154	9171.0	9168.0	9157.3	9157.3	2216	4660	5	8	77.6	334.3	96.6	30.0	529.9	894.7	15211.1	145.9	17.01
	31	3839	**3839**	**3839**	**3839**	**3839**	3839.0	3839.0	3839.0	3839.0	189	341	3	1	6.5	23.8	53.8	2.9	524.3	881.7	11387.9	125.6	1.49
	61	5856	**5856**	**5856**	**5856**	**5856**	5856.0	5856.0	5856.0	5856.0	29	36	1	1	0.9	2.3	0.2	0.2	480.1	800.3	7480.1	115.1	1.60
	91	14527	**14527**	**14527**	**14527**	**14527**	14527.0	14527.0	14527.0	14527.0	89	22	1	1	3.2	1.5	5.7	1.3	512.7	857.2	9463.2	144.0	1.23
	121	13346	**13346**	**13346**	**13346**	**13346**	13346.0	13346.0	13346.0	13346.0	390	5	2	1	13.1	0.3	10.0	0.1	495.4	829.5	7646.5	119.6	0.85
100	1	31489	33362	33344	33278	33280	33365.3	33354.0	33278.0	33289.3	14449	18239	839	40	5026.6	12986.5	97462.8	7434.5	10440.7	17909.0	123686.2	19265.8	246.5
	31	17012	17127	17150	17022	17021	17149.7	17154.3	17022.0	17043.7	18965	7374	20	21	6388.0	5023.4	2499.6	4079.2	10146.1	17047.1	128010.8	9060.8	1724.9
	61	14634	14691	14684	14661	14663	14698.7	14684.0	14661.0	14664.0	26049	11799	4	13	8782.6	7984.5	636.4	1770.1	10143.7	16924.9	128529.3	3529.7	510.5
	91	35784	35813	35800	35788	35791	35816.0	35805.7	35788.0	35791.0	8320	855	15	12	2724.2	562.0	1857.4	1047.1	9877.4	16591.3	151380.8	3303.5	134.1
	121	59347	59389	59389	**59347**	59352	59393.7	59398.3	59347.0	59353.3	10665	4974	15	21	3377.5	3102.4	1992.0	3155.6	9494.8	15617.3	132072.4	6964.1	239.8

Pessoa, A. A. and Barboza, E. U. (2011). UFFLP - an easy API for mixed, integer and linear programming. Available at: http://www.gapso.com.br/ufflp.

Pinedo, M. L. (2012). Scheduling: Theory, algorithms, and systems. *Springer Publishing Company, Incorporated*, 4a ed.:1–104.

Rabadi, G., Mollaghasemi, M., and Anagnostopoulos, G. C. (2004). A branch-and-bound algorithm for the early/tardy machine scheduling problem with a common due-date and sequence-dependent setup time. *Comput. Oper. Res.*, 31:1727–1751.

Rodrigues, R., Pessoa, A., Uchoa, E., and de Aragão, M. P. (2008). Heuristic algorithm for the parallel machine total weighted tardiness scheduling problem. *Relatórios de pesquisa em engenharia de produção*, 8:1–12.

Rym and M'Hallah (2007). Minimizing total earliness and tardiness on a single machine using a hybrid heuristic. *Computers and Operations Research*, 34(10):3126–3142.

Shabtay, D. and Steiner, G. (2012). Scheduling to maximize the number of just-in-time jobs: A survey. *Springer Optimization and Its Applications*, 60:3–20.

Sourd, F. and Kedad-Sidhoum, S. (2003). The one-machine problem with earliness and tardiness penalties. *Journal of Scheduling*, 6:533–549.

Tanaka, S. (2012). An exact algorithm for the single-machine earliness-tardiness scheduling problem. *Springer Optimization and Its Applications*, 60:21–40.

Tanaka, S., Sasaki, T., and Araki, M. (2003). A branch-and-bound algorithm for the single-machine weighted earliness-tardiness scheduling problem with job independent weights. *Systems, Man and Cybernetics*, 2:1571–1577.