Cyclic scheduling of perishable products in parallel machine with release dates, due dates and deadlines

Nargess Shirvani* [1], Rubén Ruiz[2] and Shahram Shadrokh[1]

[1]Industrial Engineering Department, Sharif University of Technology, Tehran, Iran
[2]Grupo de Sistemas de Optimización Aplicada, Instituto Tecnológico de Informática, Ciudad Politécnica de la Innovación, Edifico 8G, Acc. B. Universitat Politècnica de València, Camino de Vera s/n, 46022 Valencia, Spain

Abstract

This paper deals with a realistic cyclic scheduling problem in the food industry environment in which parallel machines are considered to process perishable jobs with given release dates, due dates and deadlines. Jobs are subject to post-production shelf life limitation and must be delivered to retailers during the corresponding time window bounded by due dates and deadlines. Both early and tardy jobs are penalized by partial weighted earliness/tardiness functions and the overall problem is to provide a cyclic schedule of minimum cost. A mixed integer programming model is proposed and a heuristic solution beside an iterated greedy algorithm is developed to generate good and feasible solutions for the problem. The proposed MIP, heuristic and iterated greedy produce a series of solutions covering a wide range of cases from slow optimal solutions to quick and approximated schedules.

Keywords: Parallel machine scheduling, Perishable products, Partial weighted earliness/tardiness, Due date, Deadline, Release date, Iterated greedy algorithm

1 Introduction

The studied problem in this research is motivated by a real scheduling problem in the food industries. In food process control, safety of products has been one of the main objectives beside temporal and financial issues (Linko, 1998) and in the case of fresh products or highly perishable foods, final products are subject to deterioration through time. Hence, in most real cases, a limited post-production shelf life is considered, such that final products can be placed on supermarket shelves with a reasonable remaining shelf life. Moreover, some food products such as fresh foods or dairy products as subgroups of Fast Moving Consumer Goods (FMCG), have a quick turnover and need to be produced and distributed over a short period of hours, days or weeks. Therefore, the whole operations, due to limited post-production shelf life, should be carried out as

*Corresponding author. Tel: +98 912 762 3064
E-mail addresses: shirvani@mehr.sharif.ir (N. Shirvani), rruiz@eio.upv.es (R. Ruiz), shadrokh@sharif.edu (Sh. Shadrokh).

fast as possible, and due to FMCG's properties, the operations can be scheduled in a repetitive manner of a particular cycles in a relatively long horizon time. The production setting we consider is an identical parallel machines shop which is capable of producing different perishable goods. Manufacturer deals with fixed retailers' orders in each cycle. Orders (jobs) need to be processed on one of the machines for a known processing time. Related to each order there is a release date, a due date and a deadline. Early and tardy jobs are penalized and in order to achieve customer's satisfaction and a productive operation, the manufacturer has to schedule jobs as close as possible to their due dates.

Production scheduling in food industries has received significant research attention and there are plenty of case studies investigating particular subjects in this area. We refer the readers to Claassen and Van Beek (1993), Randhawa et al. (1994) and Tadei et al. (1995) in which aggregation of planning and scheduling of food industries is investigated. Moreover, Van Donk (2001), Soman et al. (2004) and Soman et al. (2007) have discussed about a combined production planning and inventory control framework in the food industry.

There is also an abundance of published researches considering machine scheduling subject to due dates or deadline constrains. Cheng and Gupta (1989) and Baker and Scudder (1990) provide a review on scheduling problems involving due dates and earliness/tardiness. A more recent survey has been also provided by Lauff and Werner (2004) considering multi machine problems with a common due date. Some papers deal with an interval called "due window" rather than due date. Anger et al. (1986) carried out the first due window study and Wan and Yen (2002), Arroyo et al. (2011) and Chen and Lee (2002) are instances of recent studies on this area. "Assignable due window" is another extension in the classic form in which the early and late due dates are treated as decision variables. We refer readers to Mosheiov and Sarig (2010) and Mor and Mosheiov (2012) as examples of this subject.

Release date of jobs has been widely taken into consideration, for example Bank and Werner (2001) discussed on parallel unrelated machines with a common due date and release dates. They presented various constructive and iterative heuristic algorithms for solving the problem. A single machine scheduling problem with release dates and due dates is also considered by Sourd (2006) in presence of sequence dependent setup times and costs. A large-scale neighborhood search is designed for solving the problem. Baptiste and Le Pape (2005) and Tercinet et al. (2004) also investigated a release date with deadline, respectively, in a single machine and multiprocessor scheduling. Huo et al. (2010) have also investigated a factory that manufactures perishable goods while considering a time window for a safe finish time of products. They suppose time windows to be disjoint and of the same size and the goal is to select a subset of jobs to produce such that maximize the total profit.

A parallel machine scheduling problem with due date to deadline window are studied by Kaplan and Rabadi (2012) while start times of the jobs are subject to ready times. Jobs are supposed to be completed before the due dates. Missing due dates is not preferred but allowed and a weighted tardiness cost will be incurred for the jobs. Our research can be considered as an extension of this paper.

In the current research we focus on the scheduling of perishable products on parallel machines. Each job has a due date, which is the preferred delivery date of the retailers and might be violated subject to a penalization as lateness penalty, and similar to Kaplan and Rabadi (2012) there is an strict deadline imposed by the retailer that should not be exceeded. Moreover, since the products are perishable, and producing them far in advance of the delivery time is not preferred, there are release dates as the earliest possible start times of the jobs. Unlike in Kaplan and Rabadi (2012), storing early products in the manufacturer sites also incurs in a job dependent holding cost.

As the main extension in our research we take into account the case of FMCG, and consider the cooperation of the manufacturer and the retailers for an extended period of time. In order to decrease changeover costs and increase reliability of the operations, different parties prefer to adopt a routine and repetitive working plan during short cycles like days, weeks or months. This type of problem, known as cyclic (periodic) scheduling, is an effective approach to deal with a set of jobs that should be iterated during a long horizon (Hanen and Munier, 1995). An abundance of researches have discussed the advantages of cyclic scheduling over static (non-cyclic) scheduling, we refer the readers to Levner et al. (2010) as a review on complexity of fundamental cyclic scheduling problems including the cyclic job shop, cyclic flowshop, and cyclic project scheduling problems. Šcha and Hanzálek (2008) and Trautmann and Schwindt (2009) are also samples of practical research in this subject.

We will consider the interaction of adjacent cycles into account and take the advantage of this extension to increase the ability and the manufacturer's flexibility to satisfy customers' orders. Compared to existing models, the studied problem in this paper, is more practical and to the best of our knowledge, a cyclic parallel machine scheduling with release dates, due dates and deadlines has not been investigated in the literature.

Since Kaplan and Rabadi (2012) demonstrated their studied problem to be NP-hard, this extension is also NP-hard. Therefore, apart from a mixed integer model we present heuristic and iterated greedy algorithms. The rest of this paper is organized as follows. In Section 2 we precisely describe the problem, the notation and the mathematical model. Section 3 is dedicated to the development of a heuristic algorithm. In Section 4 an iterated greedy method is presented. Section 5 is to illustrate the numerical experiments and the last section concludes the paper and suggests topics for future research.

2 Problem description and mathematical model

We consider a production scheduling problem with identical parallel machines capable of producing different perishable jobs over a cycle of length T. Manufacturer receives a set $J = \{1,\ 2, \dots,\ n\}$ of n different orders (jobs) from retailers that need to be processed on a set $M = \{1,\ 2, \dots,\ m\}$ of m identical machines without preemption. Each job $j \in J$ has a due date d_j, which is the preferred delivery date of the retailers, a release date r_j as an earliest start time of the job, and a deadline $\bar{d}_j$ which is the latest possible completion time of the job. The jobs are delivered to retailers during the corresponding time window bounded by the due date and the deadline. The retailers do not accept jobs after the deadline, while early jobs can be held on the production site. Jobs that are completed before their due dates are subject to a holding cost and are penalized by rate h_j. Jobs that are completed after their due dates are also subject to a

penalization by a rate of w_j as a lateness cost.
Machines are always available and at most one product can be processed on each machine at any moment. The goal of this research is to schedule jobs on machines, in order to minimize the total earliness and lateness costs while adhering to the release date and the deadline constraints.
Since in this case we consider a company that produces products with a quick turnover and establishes a long term relationship with retailers, it is supposed that the production orders come up iteratively through determined cycles such as weeks, 10 days periods or months. These manufacturers are usually interested in designing a routine production plan for consecutive cycles, while the interaction between adjacent cycles is taken into account.
A mixed integer programming (MIP) model is designed and provides a cyclic schedule for the problem. The parameters and the decision variables are now defined.

Parameters:
T : Cycle length
p_j : Processing time of job j
d_j : Due date of job j
$\bar{d}_j$: Deadline of job j
r_j : Release date of job j
h_j : Earliness penalty of job j
w_j : Tardiness penalty of job j
M : A large positive integer
F : A large positive integer as compensation for rejecting a job

Variables:
C_j : Completion time of job j
E_j : Earliness of job j
T_j : Tardiness of job j
C_i^d : Completion time of a dummy job on machine i

Binary variables:
x_{ijk} : 1 if job j precedes job k on machine i;
α_j : 1 if job j is considered as a tardy job;
β_j : 1 if $C_j - d_j \geq 0$;

$$\min Z = \sum_{j=1}^{n} (h_j E_j + w_j T_j) \tag{1}$$

$s.t.$

$$\sum_{i=1}^{m} \sum_{k=0,\, k\neq j}^{n} x_{ijk} \leq 1 \qquad j = 1, \ldots, n \tag{2}$$

$$\sum_{k=0,\, k\neq j}^{n} x_{ijk} = \sum_{k=0,\, k\neq j}^{n} x_{ikj} \qquad j = 1, \ldots, n \quad i = 1, \ldots, m \tag{3}$$

$$\sum_{j=1}^{n} x_{ij0} \leq 1 \qquad i = 1, \ldots, m \quad (4)$$

$$\sum_{k=1}^{n} x_{i0k} \leq 1 \qquad i = 1, \ldots, m \quad (5)$$

$$x_{ijk} + x_{ikj} \leq 1 \qquad i = 1, \ldots, m \; j, k = 1, \ldots, n \quad (6)$$

$$C_k \geq C_j + p_k - M\left(1 - x_{ijk}\right) \qquad i = 1, \ldots, m \; j, k = 1, \ldots, n \quad (7)$$

$$C_i^d \geq C_j - M\left(1 - x_{ij0}\right) \qquad i = 1, \ldots, m \quad j = 1, \ldots, n \quad (8)$$

$$C_j \geq C_i^d + p_j - T - M\left(1 - x_{i0j}\right) \qquad i = 1, \ldots, m \quad j = 1, \ldots, n \quad (9)$$

$$E_j \geq d_j - C_j + T\left(1 - \beta_j\right) - M\alpha_j \qquad j = 1, \ldots, n \quad (10)$$

$$T_j \geq C_j - d_j + T\beta_j - M\left(1 - \alpha_j\right) \qquad j = 1, \ldots, n \quad (11)$$

$$M\beta_j \geq d_j - C_j \qquad j = 1, \ldots, n \quad (12)$$

$$M(1 - \beta_j) \geq C_j - d_j \qquad j = 1, \ldots, n \quad (13)$$

$$E_j \leq d_j - r_j - p_j + M\left(1 - \sum_{i=1}^{m} \sum_{k=0,\, k \neq j}^{n} x_{ijk}\right) + M\alpha_j \qquad j = 1, \ldots, n \quad (14)$$

$$T_j \leq \bar{d}_j - d_j + M\left(1 - \sum_{i=1}^{m} \sum_{k=0,\, k \neq j}^{n} x_{ijk}\right) + M\left(1 - \alpha_j\right) \qquad j = 1, \ldots, n \quad (15)$$

$$E_j \geq \frac{F}{h_j + w_j}\left(1 - \sum_{i=1}^{m} \sum_{k=0,\, k \neq j}^{n} x_{ijk}\right) \qquad j = 1, \ldots, n \quad (16)$$

$$T_j \geq \frac{F}{h_j + w_j}\left(1 - \sum_{i=1}^{m} \sum_{k=0,\, k \neq j}^{n} x_{ijk}\right) \qquad j = 1, \ldots, n \quad (17)$$

$$E_j,\; T_j\; C_i^d \geq 0, \quad 0 \leq C_j < T \qquad j = 1, \ldots, n$$

$$x_{ijk},\; \alpha_j,\; \beta_j \in \{0, 1\} \qquad i = 1, \ldots, m \; j, k = 0, \ldots, n$$

The objective function (1) minimizes the total earliness and tardiness costs. In the original problem, rejecting orders is not allowed and therefore in some cases, limited machine capacity and strict deadlines might result in an infeasible problem. Here, similar to Kaplan and Rabadi (2012), a large integer number F determines the cost of rejecting a job and it must be considered big enough in order not to affect the optimal solution. In the model, the binary variable x_{ijk} determines sequence of the jobs on the machines. Eq. (2) insures that each job is assigned at most to one machine and precedes at most one job. A dummy job $j = 0$ with zero processing time is supposed to be processed first on all machines and in order to keep the cyclic property, the dummy job is also considered to succeed the last job on the machines. By considering the dummy job, if a job is assigned to a machine it must precede and succeed exactly one job, this constraint is supported by Eq. (3) to (5). It is possible a job not to be assigned to the machines and a machine does not work at all during the cycles. Constraints (6) guarantees that job j cannot precede and succeed the same job k. Constraints (7) ensures that there is a gap, at least, of length p_j between start time of job j and its successor and Eq. (8) and (9) are added to the model in order to adjust the completion time of the dummy jobs on each machine.

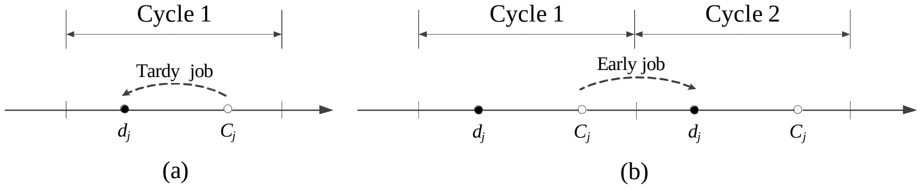

Figure 1: Early or tardy job by considering due dates in consequent cycles.

Due to the cyclic property of scheduling, a job at the same time can be early and tardy by considering the corresponding orders in consequent cycles. In other words, if a job is tardy (early) by considering due date of current cycle it can be early (tardy) for the same due date placed in the next (previous) cycle. This concept is depicted on Figure 1, where part (a) shows job j as tardy, due to the corresponding due date inside cycle 1. However, by considering due date of cycle 2, it can be an early job. Therefore, we first need to determine whether the job is early or tardy. Moreover, as it is shown in the following, for a tardy (early) job j in case due date and completion time belong to the same cycle, tardiness (earliness) is calculated as $C_j - d_j$ ($d_j - C_j$); while, in case job j satisfies an order of the previous (next) cycle the tardiness (earliness) is calculated by $(T + C_j) - d_j$ ($(T + d_j) - C_j$). Constraints (10) to (13) evaluate these requirements by using two binary variables α_j and β_j.

$$\mathrm{E}_j = \begin{cases} d_j - C_j & \text{if } C_j \leq d_j; \\ T + d_j - C_j & \text{if } C_j > d_j. \end{cases}$$

$$\mathrm{T}_j = \begin{cases} C_j - d_j & \text{if } C_j \geq d_j; \\ T + C_j - d_j & \text{if } C_j < d_j. \end{cases}$$

Due to release date and deadline limitations, constraints (14) and (15) ensure that earliness or tardiness of a processed job does not violate the maximum allowed earliness and tardiness. The two last inequalities adjust earliness and tardiness of a rejected job such that large compensation F inures in the objective function.

Each feasible solution of the problem includes a set R of the rejected jobs and a list C of the completion times in which completion times of the rejected jobs are set to the large number M. Then corresponding to each machine $i \in M$ there is a sequences S_i, of the jobs in increasing order of the completion times. Therefore a complete solution S, consists of $m + 2$ elements that can be shown by $S = \{C, R, S_1, S_2, \ldots, S_m\}$. Regarding to well known WSPT rule it can be easily verified that the properties below are satisfied in the optimal solution. Consider S as an optimal solution:

- For two consecutive early jobs j and k in S_i, if $d_j \geq C_k$ and $h_j/p_j > h_k/p_k$, j precedes k if and only if the release date of k (r_k) is greater than the start time of k.
- For two consecutive tardy jobs j and k in S_i, if $d_k \leq C_j - p_j$ and $w_j/p_j < w_k/p_k$, j precedes k if and only if the deadline of i ($\bar{d}_j$) is smaller than the completion time of k.

- For two consecutive jobs j and k in S_i, which start after d_j and finish before d_k, job j precedes job k.

3 The heuristic algorithm

In this section, a constructive heuristic algorithm is presented in which jobs are selected based on different priority rules and in a greedy way are scheduled at the best available position among all machines. The general framework of the algorithm is to select a due date as a central point and schedule the feasible jobs around it in a greedy manner such that no idle time occurs among the jobs scheduled in each machine. Then, if any job remains unscheduled, the next center point is chosen, and this procedure is iterated until no job remains to be scheduled.
As the first center point, our intention is to select the most occupied part of the cycle, where a relatively large number of jobs are available to be scheduled. Index ρ_t called "density factor" is proposed corresponding to time t based on Eq. 18, which evaluates how occupied is the area around the selected time. Where $\Delta_{d_j t}$ is calculated by Eq. 19 as the minimum cyclic distance between d_j and time t.

$$\rho_t = \sum_{j \in J} p_j \left(\frac{1}{T} + \Delta_{d_j t} \right)^{-1} \qquad 0 \leq t < T \tag{18}$$

$$\Delta_{\, t\, t'} = min\{|t - t'|, T - |t - t'|\} \tag{19}$$

By using Eq. 18, the due date with the largest density factor is selected as the central point of scheduling. The unscheduled jobs can then be processed to the left or to the right of this center point. Therefore, corresponding to each center point, there are two time frames on each machine, which determine the available times for scheduling.
Suppose at the first step, d^* is selected as the central due date. Since machines are available the whole cycle, the processing of the selected job might be started inside processing frames of length T to the right or to the left of d^*. So, as it is shown in Figure 2 (a) for each machine $i \in M$ the first selected job might be scheduled inside the interval bounded by the lower bound of the left processing frame L_i^L and the upper bound of the right processing frame R_i^U. Once a job is scheduled inside the left (right) processing frame, an scheduled frame is created at the middle of the scheduling zone and the upper bound (lower bound) of the left (right) processing frame L_i^U (R_i^L) must be updated. Moreover, due to the cyclic property, scheduling a job in the left (right) processing frame affects the maximum (minimum) available time of the right (left) processing frame and decreases the length of the frame as it is illustrated in Figure 2 part (b).

Once all the frames on different machines are updated, candidate jobs to be scheduled next must be determined. To do this, we determine $\omega : \left[\omega^L, \, \omega^U\right]$ as a common period to all scheduled frames on the machines and select candidate jobs j among the unscheduled jobs such that $\omega^L - p_j \leq d_j \leq \omega^U + p_j$. Then, a criterion is needed to rank the candidate jobs and to select one to be scheduled. Various criteria have been proposed in the literature of the job scheduling with earliness/tardiness penalties, in order to determine scheduling priority of the

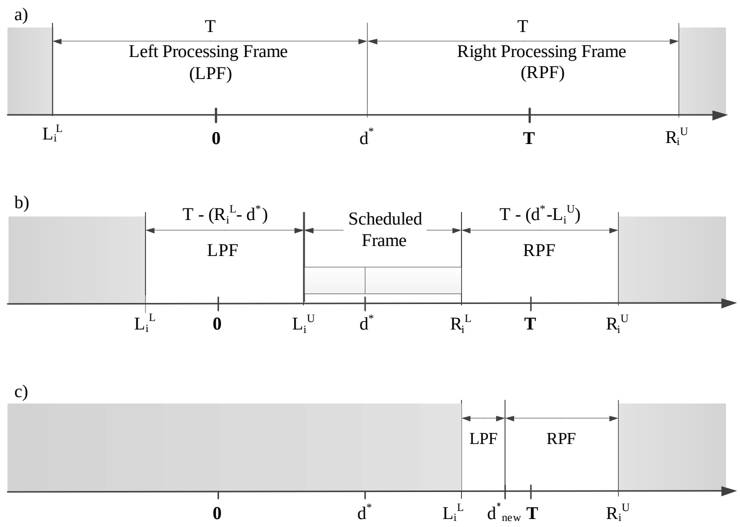

Figure 2: Processing and scheduling frames around the central due dates for machine i.
a) The initial central due date and the corresponding processing frames. b) Scheduling of feasible jobs around the central due date and updating the processing frames. c) Next selected central due date and new processing frames.

jobs. Bank and Werner (2001) have proposed and compared different criteria, in an unrelated parallel machine scheduling problem with common due dates, and concluded the superiority of the ranking based on the job's slack. Here, we consider *total slack* which is calculated as $s_j^t = \bar{d}_j - r_j - p_j$, beside some other factors which are listed below. These ranking criteria will be all evaluated, and the best one will be embedded in the heuristic algorithm.

- Nonincreasing/Nondecreasing total slack (TSDEC/TSINC)
- Nonincreasing/Nondecreasing due date (DDDEC/DDINC)
- Nonincreasing/Nondecreasing ratio $(w_j h_j)/p_j$ (WHDEC/WHINC)

The selected job is scheduled as close as possible to the central due date. Therefore, at each machine i there are two possibilities for each selected job j: placing in the left processing frame and $C_j = L_i^U$ or scheduling in the right processing frame and $C_j = R_i^L + p_j$. These alternatives must be checked to see if they meet the release date an deadline constraints. Furthermore, jobs are not allowed to exceed the processing frames' bounds of the machines. When all feasible alternatives are tested the one with minimum earliness/tardiness cost is selected to schedule job j, and in case there is no feasible alternative, the job will be rejected.
The whole procedure is iterated until no available job is left and when we encounter with an empty list, the next central due date must be selected among the remaining due dates by the selection criteria of minimum distance to the common schedule frame's bounds. As the next step, the processing frame, containing new central due date d^*_{new}, is considered as the scheduling zone and frames' bounds need to be updated. Part (c) of Figure 2 shows the new scheduling zone and the frames. The algorithm stops when no unscheduled job remains.

The pseudo code for the whole procedure is given in Algorithm 1, computational complexity of this algorithm is $O(n^2m)$.

Algorithm 1 Heuristic Algorithm

d^* : Central due date
F_j^*: Ratio related to job j based on the selected ranking criterion
U : Set of unscheduled jobs
R : Set of rejected jobs
A : Set of available jobs
M : Set of machines
k : Number of distinct due dates
D_r: r^{th} distinct due date
D : Set of distinct due dates
D^R: Set of remaining due dates
L_i^L (L_i^U): Lower (Upper) bound of the left processing frame at machine i
R_i^L (R_i^U): Lower (Upper) bound of the right processing frame at machine i
ω^L (ω^U): Lower (Upper) bound of the common scheduled frame

Set $U = \{1, 2, \ldots, n\}$, $M = \{1, 2, \ldots, m\}$, $D = \{1, 2, \ldots, k\}$.
Select d^* such that $\rho_{d^*} = \max_{r \in D} \{\rho_{D_r}\}$ and $\forall i \in M$ set $L_i^L = d^* - T$, $L_i^U = d^*$, $R_i^L = d^*$ and $R_i^U = d^* + T$
while U not empty **do**
  Set $\omega^L = \max_{i \in M} \{L_i^U\}$ and $\omega^U = \min_{i \in M} \{R_i^L\}$, $\forall j \in U$ if $\omega^L - p_j \leq d_j \leq \omega^U + p_j$ then add j to A
  while A not empty **do**
    Find $j \in A$ such that $F_j^* = \max_{k \in A} \{F_k^*\}$
    for $i = 1\, to\, m$ **do**
      Schedule j on machine i in two positions such that $C_j = L_i^U$ or $C_j = R_i^L + p_j$.
      Check the feasibility criteria for both cases ($C_j - p_j \geq \max\{r_j, L_i^U\}$, $C_j \leq \min\{\bar{d}_j, R_i^U\}$)
      Consider new feasible position as the best alternative in case provides a better solution than the best known alternative.
    end for
    If there is no feasible alternative for the selected job, consider j as a removed job and add it to R; Otherwise schedule j in the best known position.
    Remove j from U and update all frames' bounds
    Update A
  end while
  if U not empty **then**
    Set $D^R : \{D_r : r \in D, \exists\, j \in U : d_j = D_r\}$
    Update d^* among $D_i \in D^R$ such that D_i has the minimum distance to the common scheduled frame's bounds.
    update all frames' bounds and update A
  end if
end while

3.1 Local Improvement

Once a feasible solution $S = \{C, R, S_1, S_2, \ldots, S_m\}$ is obtained, two simple local searches are conducted to improve the quality of the solution. The first proposed improvement deals with idle time of each machine and a shifting of jobs which are processed just before or after the idle time in such a way that no other jobs are displaced. It is also straightforward to adopt a greedy style and in each machine choosing the shift of the job which provides maximum improvement. This procedure is repeated until no further improvement is possible. If in solution S no job is rejected, all n jobs are processed on machines and therefore at most n separated idle time might occur. Consequently a single step of this improvement has computational complexity of $O(n)$.

We also take advantage of the previous mentioned properties of optimum solution by performing an adjacent pairwise interchange. This improvement can be done by considering all feasible interchange of adjacent jobs on each $S_i \in S$. This procedure starts from the last job j in sequence S_i and compares it with the previous job, or if job j is placed at the first position it continues by the last job in S_i. In case the pairwise interchange improves the solution, the interchange is performed and job j then is compared to the previous job in improved S_i for further improvement. In worst case at each stage, the selected job must be compared with $n-1$ different jobs. Thus the complexity of a single stage of this improvement is $O(n)$. This procedure also is repeated until no further improvement is possible.

In order to perform a complete local search phase, a solution is first improved by applying the first proposed local search and once no improvement is possible the second local search is applied on the new improved solution.

4 Iterated greedy algorithm

Iterated greedy (IG) which was first introduced by Ruiz and Stützle (2007) for scheduling problems, is well known for its very simple principles and has exhibited far better performance than other more complex approaches in the literature. The IG is a constructive two-phase heuristic which starts from an initial solution and iteratively applies a greedy heuristic to improve it. The first phase, called destruction, randomly removes some solution components and then the second phase, called construction, reinserts the removed components into the solution in such a way that minimum possible cost is obtained at each stage. An acceptance criterion determines whether the current solution is replaced by the solution generated in the construction phase.

Ruiz and Stützle (2008) has reported the superiority of the IG for solving the sequence dependent setup times flowshop problem in comparison to many other solutions. Ying and Cheng (2010), Minella et al. (2011), and Kang et al. (2011) are also samples of recent extensions and applications of the IG heuristic. Inspired by these results, in this research an IG algorithm is designed for the problem under consideration. The following subsection describes the proposed IG algorithm in detail.

4.1 Destruction phase

In the first step, we start from a feasible solution $S = \{C, R, S_1, S_2, \ldots, S_m\}$, generated by the proposed heuristic algorithms. The destruction procedure choses r random different jobs in such a way that rejected jobs $j \in R$ have twice the chance of being selected. Selected jobs are removed from the initial solution S. The result is a partial solution $S^P = \left\{C^P, R^P, S_1^P, S_2^P, \ldots, S_m^P\right\}$ and a sequence of removed jobs π in the order of selection.

4.2 Construction phase

The construction phase considers the partial schedule S^P, and in r stages reinserts removed jobs in order of π to obtain a complete solution S'. At each stage the selected job j must be scheduled on each machine i at a feasible completion time t which is randomly selected in interval $\left[r_j + p_j, \bar{d}_j\right]$. Completion time of the other jobs in S_i^P are then updated. We start from the first job k in sequence S_i^P such that $C_k \geq t$. Due to the cyclic property if there is no job that satisfies the condition, the first job of sequence S_i^P is selected. This job is rescheduled on machine i as early as possible. In the same way, the remaining jobs are rescheduled in order of sequence S_i^P. In case for a selected job there is no feasible alternative it is considered as a rejected job.
Selecting a random completion time and rescheduling the jobs is repeated λ times on each machine. Hence, for each member of π, we generate $m \times \lambda$ alternatives that need to be evaluated. In each complete solution, jobs can be classified in three groups: early jobs E, tardy jobs T and rejected jobs R. In an evaluation step each removed job is penalized by large number F. Therefore, the evaluation function for solution S is:

$$f(S) = \sum_{j \in E} h_j E_j + \sum_{j \in T} w_j T_j + \sum_{j \in R} F \tag{20}$$

Once a removed job is inserted in S^P, it will be removed from π and then, the whole procedure is iterated until π is empty. At the end of construction phase, the local improvements explained in Subsection 3.1 are carried out to improve the candidate solution S'.

4.3 Acceptance criterion

After a complete iteration of a greedy algorithm it should be decided whether the new solution S' is accepted as an initial solution for the next iteration. Instead of considering a better objective value, similar to Ruiz and Stützle (2007) we consider a simple simulated annealing-like acceptance criterion with a constant temperature which is computed as follows, where T_{IG} is a parameter that needs to be adjusted and the quotient calculates the average of maximum possible earliness/tardiness of a job.

$$Temperature = 0.1 \times T_{IG} \times \frac{\sum_{j \in J} \left[h_j(d_j - r_j - p_j) + w_j(\bar{d}_j - d_j)\right]}{2 \times n} \tag{21}$$

5 Experimental results and computational analysis

Comprehensive numerical experiments are conducted for testing and comparing the efficiency of algorithms and quality of solutions. Various instances are generated randomly in which cycle length (T), job number (n) and machine number (m) are considered as the main parameters that determine size of the

Table 1: Main parameters of the random instances.

Parameter	Description	Number of levels	Values
Small instances			
T	Cycle time	3	5, 7, 10
n	Number of jobs	3	6, 8, 10
μ	Ratio of n to m	2	5, 10
Large instances			
T	Cycle time	3	7, 10, 30
n	Number of jobs	4	10, 30, 60, 100
μ	Ratio of n to m	4	5, 10, 20, 30

instances. Three levels $\{7, 10, 30\}$ for T and four levels $\{10, 30, 60, 100\}$ for n are considered here. Since we are discussing a case of perishable products with a quick turnover, like fresh foods and dairy products, a maximum cycle of 10 days is realistic. In the other hand, the maximum number of 60 jobs, as different retailers orders during a short cycle, is large enough. However, larger T and n equal to 30 and 100, respectively, are considered to evaluate the efficiency of the solutions in larger instances. similar to Kaplan and Rabadi (2012), the number of machines is considered as fraction of n and is calculated by $m = \left\lceil \frac{n}{\mu} \right\rceil$. μ is considered to vary from low value 5 to high value 30. Beside the above mentioned instances, small size instances are designed for evaluating the proposed methods in comparison with the optimum solution provided by solving MIP model. Different levels of the main parameters are demonstrated in Table 1. In order to generate a random instance, after selecting levels of all main parameters, μ determines the number of the machines. Job related parameters (p_j, h_j, w_j, d_j, r_j, $\bar{d}_j$) are generated then as follows: processing time of each job p_j is determined from a uniform distribution $U\,[0.1,\ 1.5]$. Earliness costs h_j and tardiness costs w_j are also independently generated based on a uniform distribution $U\,[1,\ 5]$. Due dates are also uniformly selected between 1 to T. Release dates and deadlines are generated such that for each job j, $d_j - r_j \in U\,[1,\ T]$ and $\bar{d}_j - d_j \in U\,[1, T]$.
All the combinations of the main parameters are considered for generating random instances and 10 instances are generated in each group, resulting in 180 small instances and 480 large instances in total. All instances with the best solutions known are available at `http://soa.iti.es`. The MIP model is solved via ILOG-IBM CPLEX 12.4 and all heuristic methods are implemented in $C\#$ 4.0. All methods are run on a cluster of 30 blade severs each one with two Intel XEON 5254 processors running at 2.5 GHz with 16 GB of RAM memory. Each processor has four cores and the experiments are carried out in virtualized Windows XP machines, each one with two virtualized processors and 2 GB of RAM memory.

5.1 Calibration of the heuristic algorithm

The first comparative analysis is dedicated to calibrate the proposed heuristic algorithm and the ranking criteria discussed in Section 3. For the calibration

Table 2: Comparative analysis of the ranking criteria in the heuristic algorithm including density factor (ρHA).

T	n		TSDEC	TSINC	DDDEC	DDINC	WHDEC	WHINC
		BR%	60	100	67	87	87	80
	10	BS%	40	40	33	20	53	27
		AD%	22	8	25	22	10	20
		BR%	40	100	60	60	60	80
7	30	BS%	0	47	7	0	27	27
		AD%	68	5	48	46	26	32
		BR%	47	93	53	67	47	73
	60	BS%	13	33	20	7	20	7
		AD%	40	4	23	21	28	21
		BR%	87	100	100	100	100	100
	10	BS%	40	53	67	33	53	47
		AD%	155	3	2	14	10	49
		BR%	87	100	93	87	87	100
10	30	BS%	33	7	20	0	33	20
		AD%	18	16	15	38	18	17
		BR%	93	100	93	93	93	93
	60	BS%	13	7	47	0	20	13
		AD%	15	15	7	30	6	15
		BR%	69	99	78	82	79	88
Total		BS%	23	31	32	10	34	23
		AD%	30	9	20	28	16	18

we employ a different random benchmark to avoid overfilling and biased results. The instances are generated according to Table 1, by considering 10, 30 and 20 as high level of T, n and μ, respectively. All the combinations are considered and in each group 5 instances are generated randomly. We perform the heuristic algorithm by applying the proposed criteria to solve the random instances. Furthermore, in order to evaluate the effect of density factor ρ, we consider two different version of the heuristic algorithm: The first, as it is explained in section 3 includes density factor and is called ρHA and the second selects the central point randomly and is called RHA. The local search is also performed in all cases. Therefore, in total twelve candidate algorithms must be evaluated. A summarized results of the first six alternatives related to ρHA are presented in Table 2. Similar results are obtained while we conduct the same experiments via RHA.

This table shows percentage of times that each criterion generates the best known solution (BS) for each instance, percentage of times that each criterion provides a solution with minimum job rejection number (BR) and average deviation of results, in comparison with the best known solution (AD). As it is shown, in all the rows TSINC generates the highest number of solutions with the minimum rejected jobs and in most of the groups this criterion provides relatively better solutions.

In order to evaluate the outputs, Eq. 20 is used to calculate the objective values, and the large number F is independently set for each instance, such that each rejected job be penalized by the largest cost, obtained in any solution for the same instance. The obtained objective values are transfered to relative

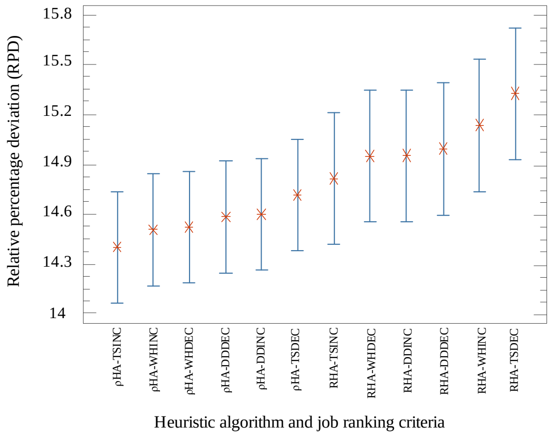

Figure 3: Means and Tukey's Honest Significant Differences (HSD) intervals (95% confidence level) of relative percentage deviation from the best known solutions for the heuristic algorithm.

percentage deviation (RPD) applying:

$$RPD = \frac{Some_{sol} - Best_{Sol}}{Best_{Sol}} \times 100 \tag{22}$$

where $Some_{sol}$ is the objective function of a solution on an instance and $Best_{sol}$ is the lowest objective value obtained in all solutions under experiment. We analyze the results by using a multi-factor analysis of variance (ANOVA) technique in which T, n and μ are considered as independent factors. As a preliminary investigation, we need to check the three main hypothesis of ANOVA that are normality, homogeneity of variance and independence of residuals. Graphical and numerical methods are known as two main groups of tools for assessing normality. Here we use a typical graphical test called Quantile–Quantile plot which checks how the residuals fit a theoretical normal distribution. The residuals are clearly homogeneous and independent while the plot depicts a strong tailed normal distribution, which is not a major problem based on the results of Rasch and Guiard (2004) and Basso et al. (2007).

The results of ANOVA indicate that all independent factors that determine instance size, are very significant. These results also demonstrate that using density factor can make a statistically significant difference, while different criteria do not provide significant differences in the response variable. For determining the best algorithm among twelve available alternatives we refer to the means plot shown in Figure 3. This plot illustrates the average of the relative percentage deviation and corresponding means and Tukey's Honest Significant Differences (HSD) intervals at the 95% confidence level. According to the plot, different criteria show the same behavior in both heuristic algorithms, while in total the ρHA reveals better performance. This plot also shows that TSINC provides better solution, however there is no statistically significant different among all the criteria at a 95% confidence level. Therefore without any significant priority between criteria, ρHA-TSINC is the selected heuristic algorithm in the rest of the experiments.

5.2 Adjusting parameters of IG algorithm

An experiment is carried out to tune the parameters of the iterated greedy algorithm which starts from an initial solution generated by the selected heuristic algorithm. IG includes 3 parameters: number of destructed jobs (r), number of iteration for reinserting each destructed job (λ) and the parameter using in calculating the temperature (T_{IG}). We consider three levels $\{1/10n, 1/8n, 1/5n\}$ for r, three levels $\{3, 5, 7\}$ for λ and five levels $\{0, 0.1, 0.3, 0.6, 1\}$ for T_{IG}.
The calibration is carried out based on the Design of Experiments (DOE) approach and a full factorial design is employed. By considering all the combinations of above mentioned parameters, 45 different treatments must be analyzed. For the experiment, the random calibration instances are used as in Section 5.1. For each instance, a time limitation of $T \times n \times m$ milliseconds is considered as the stopping criterion. The experiment was analyzed by the ANOVA technique, where beside non-controllable factors related to the instance size, r, λ and T_{IG} are considered as the controllable factors and the RPD is the response variable. The results indicate that all factors related to instance size result in statistically significant differences. Also, the different levels of r, λ and T_{IG} provide significant differences in the response variable. Means plots are used again, to determine the best level of each parameter. Figure 4 illustrates different RPD levels of r, where we can see that increasing r results in statistically worse algorithms, therefore level $r = 1/10 \times n$ is selected for the number of destructed jobs. Based on Figure 5 it seems that levels 0, 0.1 or 0.3, for T_{IG}, statistically provide the same RPD, therefore without any priority we select $T_{IG} = 0.3$. Figure 6 depicts the means plot for λ in which decreasing λ results in statistically better algorithms. Hence, level $\lambda = 3$ is selected as the best level of λ.

5.3 Experimental evaluation

In this section, a comparative computational experiment is conducted to evaluate the selected heuristic method and the calibrated iterated greedy algorithm. We consider the proposed heuristic algorithm in two different versions, where the former version does not include a local search, referred to as SHA, while the latter one uses local search and is denoted as HALS. The iterated greedy algorithm is also considered in different forms. In the first one a simple IG, denoted as SIG, is considered such that starts from a naive solution of rejecting all jobs and does not include a local search phase. In the second IG algorithm, referred to as HAIG, a solution generated by the selected heuristic algorithm is considered as an initial solution while no local search is used. The last variant, denoted by HAIGLS starts from a solution generated by the selected heuristic algorithm and includes the local search phase.
In the first experiment, the set of 180 small test instances are tested to evaluate the deviation of the proposed algorithms in comparison with the optimum solutions. ILOG-IBM CPLEX 12.4 is used to solve the MIP model of each instance such that the best current solution is considered as the final solution, in case the optimal solution is not obtained after the maximum CPU time which is set to three hours. In the experiment a few number of instances reached the time limit of three hours and there is also an instance in which an out of memory error was found. Similar to the other tests, a cluster of 30 blade severs each one

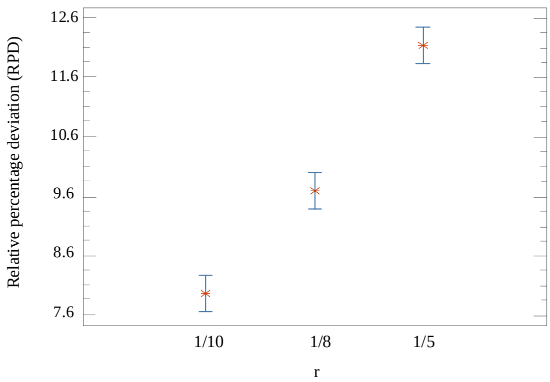

Figure 4: Means and Tukey's Honest Significant Differences (HSD) intervals (95% confidence level) of relative percentage deviation from the best known solutions for the number of destructed jobs.

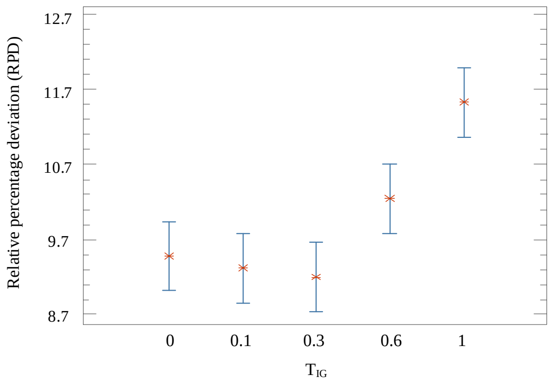

Figure 5: Means and Tukey's Honest Significant Differences (HSD) intervals (95% confidence level) of relative percentage deviation from the best known solutions for the temperature.

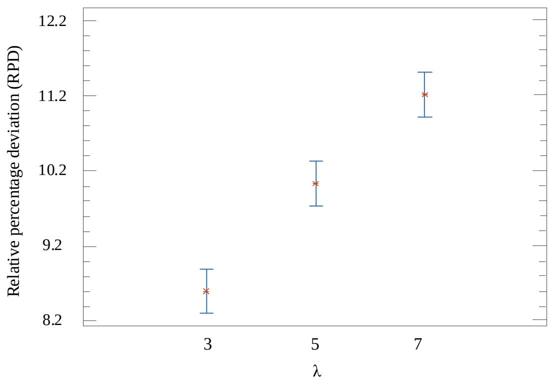

Figure 6: Means and Tukey's Honest Significant Differences (HSD) intervals (95% confidence level) of relative percentage deviation from the best known solutions for the iteration of reinserting the destructed jobs.

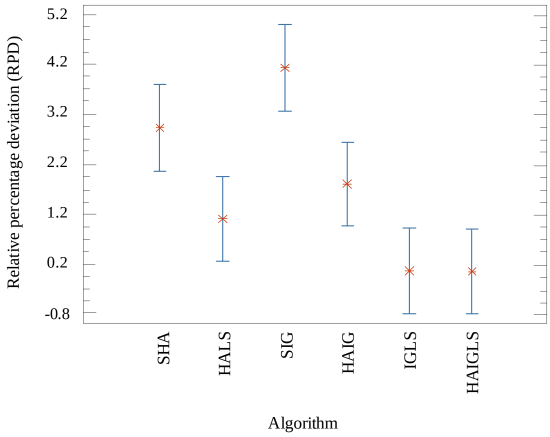

Figure 7: Means and Tukey's Honest Significant Differences (HSD) intervals (95% confidence level) of relative percentage deviation from the best known solutions for the algorithms over set of the small instances.

with two Intel XEON 5254 processors running at 2.5 GHz with 16 GB of RAM memory is used in the current experiment.
Table 3 summarizes the results for all proposed methods in which the RPD measure is calculated over the optimal value in case CPLEX provides the optimal solution. The heuristic algorithms do not depend on the CPU time while for iterated greedy-based algorithms a maximum elapsed CPU time, considering problem size, is set as stopping criteria. Here $(T \times n \times m) \times \tau$ milliseconds is considered as the stopping criterion where τ is tested at three values of $\{30, 60, 90\}$. In table 3 the results for different values of τ are separated by dashes. Based on the results, all the IG-based algorithms dominate the heuristic methods, in instances with the cycle length of 5; while for larger cycle lengths, heuristic methods outperform the simple form of IG algorithm (SIG). Generally the best solutions are provided by the IGLS and HAIGLS, where the local search phase is applied besides the iterated greedy algorithm.
Similar to the previous experiments, an analysis of variance (ANOVA) is used in order to verify if the observed differences in the performance of the tested methods are statistically significant. Figure 7 depicts the corresponding means plot with Tukey's HSD intervals at the 95% confidence level. In the plot SIG shows the worst performance however at a 95% confidence level it is not significantly different from SHA. HAIG in average provides better solution in comparison with simplest IG and it confirms that starting from a better solution might improve the results; while there is no statistically significant difference between HAIG and the two heuristic methods. In general all three algorithms inclduing the local search phase perform better than others. From the plot we can see that combination of IG and local search provides the same outputs and initial solution of the algorithm does not statistically affect the results.

The next experiments are carried out over the 480 large instances. Here also a maximum CPU time limitation of $(T \times n \times m) \times \tau$ milliseconds is considered and τ is set to $\{30, 60, 90\}$. The results, for different combinations of T and n, are summarized in Table 4 in which in most of the rows all IG-based algorithms

Table 3: Average percentage deviation from the optimum solutions for the small instances with the stopping criteria set to $(T \times n \times m) \times \tau$ milliseconds maximum CPU time for iterated greedy-based algorithms.

T . n .m	SHA	HALS	SIG	HAIG	IGLS	HAIGLS
5 . 6 . 2	162.7	143.8	123.9 - 86.5 - 103.2	45.5 - 47.1 - 43.8	5.0 - 3.9 - 3.9	4.1 - 3.9 - 3.9
5 . 6 . 1	110.9	69.3	23.5 - 18.5 - 17.1	23.1 - 30.5 - 22.6	4.3 - 4.3 - 6.0	2.3 - 6.0 - 4.2
5 . 8 . 2	66.6	49.7	54.4 - 50.3 - 53.3	24.6 - 23.8 - 31.4	6.2 - 4.5 - 1.9	3.0 - 2.8 - 2.4
5 . 8 . 1	99.2	69.6	40.1 - 36.8 - 24.2	24.1 - 17.4 - 36.5	13.0 - 10.1 - 8.1	8.3 - 5.3 - 9.8
5 . 10 . 2	114.3	71.8	68.4 - 62.2 - 55.8	51.1 - 48.0 - 50.9	11.9 - 10.7 - 10.1	8.8 - 13.1 - 6.7
5 . 10 . 1	74.4	66.6	7.1 - 13.8 - 10.2	13.2 - 6.1 - 7.1	10.8 - 5.9 - 4.1	6.4 - 5.8 - 6.5
7 . 6 . 2	807.8	353.8	452.1 - 419.4 - 410.4	284.3 - 313.0 - 303.3	1.3 - 0.0 - 0.5	1.4 - 0.0 - 0.0
7 . 6 . 1	149.7	34.2	223.3 - 213.4 - 206.5	105.2 - 104.3 - 102.7	1.3 - 1.1 - 1.1	2.1 - 1.9 - 0.4
7 . 8 . 2	94.1	57.4	408.3 - 363.2 - 364.7	81.4 - 81.2 - 80.4	1.2 - 2.7 - 1.1	1.7 - 1.1 - 1.9
7 . 8 . 1	81.0	54.9	52.5 - 50.6 - 56.4	42.7 - 43.4 - 45.1	5.6 - 6.9 - 5.5	1.8 - 5.8 - 5.8
7 . 10 . 2	163.3	49.7	361.3 - 287.8 - 227.6	100.7 - 89.2 - 118.3	6.3 - 5.5 - 1.4	1.9 - 1.7 - 0.5
7 . 10 . 1	108.6	85.9	39.6 - 40.4 - 40.9	43.4 - 27.3 - 32.9	6.9 - 7.4 - 6.7	6.2 - 6.6 - 5.1
10 . 6 . 2	34.6	34.6	32.8 - 31.1 - 28.0	18.3 - 11.8 - 11.1	0.0 - 0.0 - 0.0	0.0 - 0.0 - 0.0
10 . 6 . 1	1487.7	64.3	3455.3 - 3421.9 - 3483.8	1390.3 - 1405.8 - 1410.4	0.0 - 0.0 - 0.0	0.0 - 0.0 - 0.0
10 . 8 . 2	479.4	207.3	1028.6 - 945.8 - 921.2	451.7 - 386.3 - 432.4	0.0 - 0.0 - 0.0	0.0 - 0.0 - 0.0
10 . 8 . 1	175.9	63.2	442.9 - 744.7 - 378.4	118.9 - 127.9 - 117.2	2.0 - 1.5 - 1.6	2.1 - 1.5 - 1.5
10 . 10 . 2	220.6	166.6	424.3 - 539.6 - 386.1	213.7 - 185.1 - 199.8	0.0 - 0.5 - 4.0	1.8 - 1.8 - 1.8
10 . 10 . 1	119.4	72.3	178.2 - 180.9 - 184.2	86.7 - 107.0 - 61.4	1.6 - 1.3 - 1.4	3.1 - 1.1 - 1.2
Average	262.6	97.2	430.9 - 437.7 - 405.4	181.6 - 178.3 - 180.8	4.1 - 3.5 - 3.1	2.9 - 3.1 - 2.8

In iterated greedy - based algorithms x - y - z shows the average percentage deviations for the setting of τ to 30, 60 and 90, respectively.

Table 4: Average percentage deviation from the optimum solutions for the large instances with the stopping criteria set to $(T \times n \times m) \times \tau$ milliseconds maximum CPU time for iterated greedy-based algorithms.

T . n	SHA	HALS	SIG	HAIG	IGLS	HAIGLS
7 . 10	194.8 - 0.8	149.7 - 1.2	206.6 - 194.6 - 222.3	82.2 - 89.5 - 83.3	47.6 - 30.5 - 24.8	10.5 - 14.1 - 16.6
7 . 30	247.9 - 0.0	213.0 - 0.0	93.4 - 80.9 - 71.9	76.7 - 65.5 - 69.3	20.5 - 16.3 - 15.4	14.9 - 6.5 - 8.4
7 . 60	163.4 - 1.2	144.9 - 1.6	70.5 - 63.7 - 63.1	54.6 - 47.0 - 46.8	16.8 - 14.3 - 9.0	11.8 - 11.8 - 9.2
7 . 100	97.7 - 2.3	86.1 - 3.5	81.6 - 76.8 - 76.1	34.2 - 32.3 - 32.4	21.6 - 14.8 - 10.2	15.0 - 15.7 - 9.8
10 . 10	448.1 - 0.0	324.7 - 0.0	577.4 - 623.7 - 526.0	270.6 - 286.4 - 256.7	8.2 - 21.2 - 21.4	6.7 - 7.7 - 6.8
10 . 30	250.9 - 0.4	205.1 - 0.8	313.1 - 299.0 - 256.2	140.9 - 134.3 - 128.7	18.4 - 11.1 - 12.0	15.5 - 14.1 - 9.0
10 . 60	269.0 - 1.6	215.2 - 2.0	270.3 - 254.9 - 236.1	129.1 - 124.9 - 127.1	22.0 - 13.7 - 8.8	17.2 - 9.9 - 8.8
10 . 100	167.3 - 2.0	115.7 - 3.1	235.3 - 229.5 - 227.2	92.4 - 92.3 - 90.0	24.2 - 13.5 - 11.3	13.9 - 8.0 - 9.3
30 . 10	998.4 - 0.0	801.6 - 0.4	13692.5 - 12216.6 - 12589.6	586.0 - 577.5 - 744.5	2.9 - 2.3 - 2.7	0.1 - 0.0 - 0.0
30 . 30	5010.1 - 0.4	3110.0 - 1.2	51976.7 - 38511.9 - 32710.5	4444.6 - 4346.7 - 4371.1	28.2 - 19.6 - 16.2	43.8 - 8.3 - 6.3
30 . 60	2630.9 - 1.2	1762.7 - 2.3	18567.4 - 15083.1 - 14685.0	2120.0 - 2043.8 - 2058.9	110.1 - 65.7 - 36.6	88.6 - 25.6 - 27.8
30 . 100	1526.8 - 2.3	1180.6 - 5.5	7596.7 - 7353.7 - 7152.9	937.8 - 877.5 - 923.7	111.4 - 97.8 - 77.0	42.8 - 12.8 - 19.1
Average	1000.4 - 1.0	692.5 - 1.8	7806.8 - 6249.0 - 5734.7	747.4 - 726.5 - 744.4	36.0 - 26.7 - 20.4	23.4 - 11.2 - 10.9

In heuristic methods x - y shows the average relative percentage deviation of instances in each group and the average CPU time in millisecond; In iterated greedy - based algorithms x - y - z shows the average percentage deviations for the setting of τ to 30, 60 and 90, respectively.

except SIG outperform the heuristic methods and similar to the previous experiment IGLS and HAIGLS result in better performance and increasing the instance size raises the gap between the performance levels. Here in most of the rows HAIGLS outperform IGLS. Moreover, in these algorithms, local search affects the quality of solutions and in average decreases the percentage deviation. In IG-based algorithms, better performance of HAIG compared to SIG confirms that starting from a better solution improves the quality of solutions significantly. However, making a comparison between IGLS and HAIGLS, reveals that in presence of local search phase, the initial solution is not so much important. The table also shows that both heuristic algorithms are time efficient.
A means plot illustrated in Figure 8 also confirms the significant difference between SIG and the other methods. The rest of the algorithms, after removing SIG, can be compared better in Figure 9. From the plot it can be seen that a local search phase is likely to decrease the average percentage deviation of solutions in heuristic methods, however there is not significant difference between SHA and HALS. Due to the same mean and Tukey's HSD intervals for HALS and HAIG it can be concluded that at the 95% confidence level combination of heuristic method with local search and IG algorithm results in statistically same outputs. This plot also confirms that combination of iterated greedy and local search provides the best solutions and IGLS and HAIGLS are statistically different from the rest of the methods. In this case different initial solutions do not provide statistically significant differences in RPD and IGLS and HAIGLS generate the same solutions at the 95% confidence level.

The last analysis is dedicated to parameter τ which adjusts the stopping criterion in the IG-based algorithms. Here also an analysis of variance (ANOVA) is applied by focusing on the interaction between τ and the algorithms. The results can be seen in Figure 10. For SIG we can observe that increasing the parameter τ improves the value of RPD while it is not able to make a significant difference in any case. For the rest of the algorithms the three intervals are totally equivalent. Therefore, it can be concluded that all the algorithms have converged applying the proposed stopping criteria.

6 Conclusions

This paper studies a cyclic parallel machines scheduling problem in the food industry environment in which the manufacturer deals with the fixed retailers' orders with given due dates in each cycle. Products have to be delivered to the retailers during a time window bounded by due dates and deadlines with a time dependent cost as a lateness penalty. Retailers do not accept products after the deadline. However, early products can be stored at the production site with a product dependent holding cost, as a weighted earliness penalty. Since products are highly perishable, storage in the production site has a job dependent time limitation and therefore a release date depicts the earliest possible start time of the jobs by considering the due date and post-production shelf life limitation. The problem is to provide a cyclic schedule of all the jobs on the parallel machines such that the orders are delivered to customers in due date to deadline

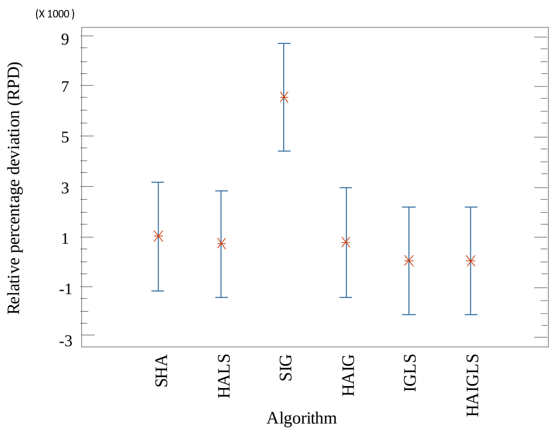

Figure 8: Means and Tukey's Honest Significant Differences (HSD) intervals (95% confidence level) of relative percentage deviation from the best known solutions for the algorithms over set of the large instances.

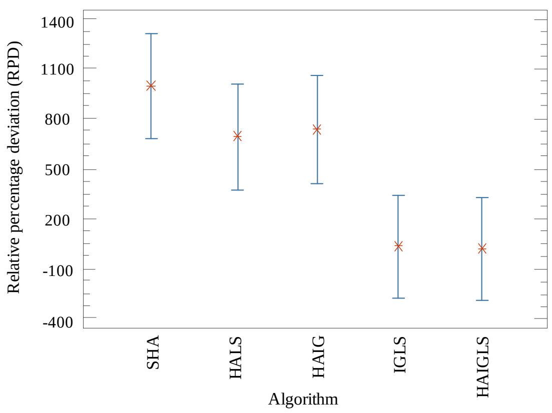

Figure 9: Means and Tukey's Honest Significant Differences (HSD) intervals (95% confidence level) of relative percentage deviation from the best known solutions for the algorithms over set of the large instances, after removing SIG.

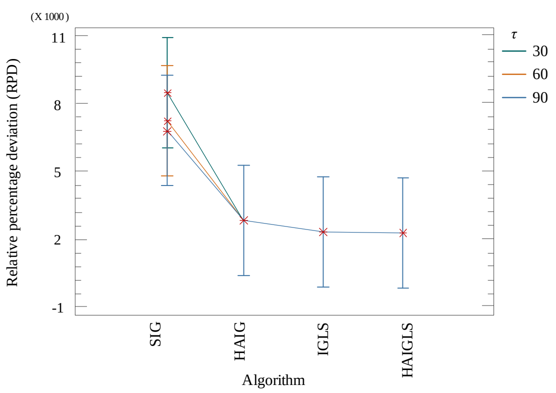

Figure 10: Means and Tukey's Honest Significant Differences (HSD) intervals (95% confidence level) of relative percentage deviation from the best known solutions for the interaction between τ and the IG-based algorithms over set of the large instances.

windows at the minimum possible earliness and tardiness costs.

A mixed integer programming model has been designed for the problem and since the problem is NP-Hard, a heuristic algorithm (HA) is developed to generate feasible solutions for the problem. Moreover, an Iterated greedy (IG) algorithm has been proposed to improve the quality of the solutions.

We have conducted the experimental design analysis to adjust the best heuristic solution and also to tune the parameters of the IG algorithm. The selected HA has been tested in comparison with IG algorithm and the results demonstrate that IG is more likely to outperform the heuristic approach. Different versions of IG and HA are tested in order to evaluate the effect of local search and experiments verify that carrying out the local search provides better solutions. IG algorithm also was tested in different variants which start from different quality solutions. The results showed that the simple IG which starts from a good initial solution, performs very well and generates solutions with less earliness and tardiness costs; while in the IG algorithm with local search phase the effect of initial solution is insignificant. According to the experiments the combination of IG and local search shows the best performance and greatly outperforms the other methods.

Extending the problem by adding setup times and setup costs, can be considered in future research. In addition, we can consider distribution planning beside production scheduling to coordinate a two stage supply chain of perishable products.

Acknowledgments

Rubén Ruiz is partially supported by the Spanish Ministry of Science and Innovation, under the projects "SMPA - Advanced Parallel Multiobjective Sequencing: Practical and Theoretical Advances" with reference DPI2008-03511/DPI and "RESULT - Realistic Extended Scheduling Using Light Techniques" with reference DPI2012-36243-C02-01 and by the Small and Medium Industry of the Generalitat Valenciana (IMPIVA) and by the European Union through the European Regional Development Fund (FEDER) inside the R+D program "Ayudas dirigidas a Institutos Tecnológicos de la Red IMPIVA" during the year 2012, with project number IMDEEA/2012/143.

References

F.D. Anger, C.Y. Lee, and L.A. Martin-Vega. Single machine scheduling with tight windows. Research Report 86-16, Department of Industrial and Systems Engineering, University of Florida, 1986.

J.E.C Arroyo, R.S. Ottoni, and A. Oliveira Paiva. Multi-objective variable neighborhood search algorithms for a single machine scheduling problem with distinct due windows. *Electronic Notes in Theoretical Computer Science*, 281: 5–19, 2011.

K.R. Baker and G.D. Scudder. Sequencing with earliness and tardiness penalties: a review. *Operations Research*, 38(1):22–36, 1990.

J. Bank and F. Werner. Heuristic algorithms for unrelated parallel machine scheduling with a common due date, release dates, and linear earliness and tardiness penalties. *Mathematical and Computer Modelling*, 33(4):363–383, 2001.

P. Baptiste and C. Le Pape. Scheduling a single machine to minimize a regular objective function under setup constraints. *Discrete Optimization*, 2(1):83–99, 2005.

D. Basso, M. Chiarandini, and L. Salmaso. Synchronized permutation tests in replicated i× j designs. *Journal of Statistical Planning and Inference*, 137(8): 2564–2578, 2007.

Z.L. Chen and C.Y. Lee. Parallel machine scheduling with a common due window. *European Journal of Operational Research*, 136(3):512–527, 2002.

T.C.E. Cheng and M.C. Gupta. Survey of scheduling research involving due date determination decisions. *European Journal of Operational Research*, 38 (2):156–166, 1989.

G.D.H. Claassen and P. Van Beek. Planning and scheduling packaging lines in food industry. *European Journal of Operational Research*, 70(2):150–158, 1993.

C. Hanen and A. Munier. A study of the cyclic scheduling problem on parallel processors. *Discrete Applied Mathematics*, 57(2):167–192, 1995.

Y. Huo, J.Y.T. Leung, and X. Wang. Integrated production and delivery scheduling with disjoint windows. *Discrete Applied Mathematics*, 158(8):921–931, 2010.

Q. Kang, H. He, and H. Song. Task assignment in heterogeneous computing systems using an effective iterated greedy algorithm. *Journal of Systems and Software*, 84(6):985–992, 2011.

S. Kaplan and G. Rabadi. Exact and heuristic algorithms for the aerial refueling parallel machine scheduling problem with due date-to-deadline window and ready times. *Computers & Industrial Engineering*, 62(1):276–285, 2012.

V. Lauff and F. Werner. Scheduling with common due date, earliness and tardiness penalties for multimachine problems: A survey. *Mathematical and Computer Modelling*, 40(5-6):637–655, 2004.

E. Levner, V. Kats, D. Alcaide López de Pablo, and T.C.E. Cheng. Complexity of cyclic scheduling problems: A state-of-the-art survey. *Computers & Industrial Engineering*, 59(2):352–361, 2010.

S. Linko. Expert systems–what can they do for the food industry? *Trends in Food Science & Technology*, 9(1):3–12, 1998.

G. Minella, R. Ruiz, and M. Ciavotta. Restarted iterated pareto greedy algorithm for multi-objective flowshop scheduling problems. *Computers & Operations Research*, 38(11):1521–1533, 2011.

B. Mor and G. Mosheiov. Scheduling a maintenance activity and due-window assignment based on common flow allowance. *International Journal of Production Economics*, 135(1):222–230, 2012.

G. Mosheiov and A. Sarig. Scheduling identical jobs and due-window on uniform machines. *European Journal of Operational Research*, 201(3):712–719, 2010.

S.U. Randhawa, C. Juwono, and S. Burhanuddin. Scheduling in multistage flowshop systems: An application in the food processing industry. *Industrial Management & Data Systems*, 94(5):16–24, 1994.

D. Rasch and V. Guiard. The robustness of parametric statistical methods. *Psychology Science*, 46(2):175–208, 2004.

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

R. Ruiz and T. Stützle. 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, 2008.

P. Šcha and Z. Hanzálek. Deadline constrained cyclic scheduling on pipelined dedicated processors considering multiprocessor tasks and changeover times. *Mathematical and Computer Modelling*, 47(9):925–942, 2008.

C.A. Soman, D.P. Van Donk, and G.J.C. Gaalman. Combined make-to-order and make-to-stock in a food production system. *International Journal of Production Economics*, 90(2):223–235, 2004.

C.A. Soman, D.P. Van Donk, and G.J.C. Gaalman. Capacitated planning and scheduling for combined make-to-order and make-to-stock production in the food industry: An illustrative case study. *International Journal of Production Economics*, 108(1-2):191–199, 2007.

F. Sourd. Dynasearch for the earliness–tardiness scheduling problem with release dates and setup constraints. *Operations Research Letters*, 34(5):591–598, 2006.

R. Tadei, M. Trubian, J.L. Avendano, F. Della Croce, and G. Menga. Aggregate planning and scheduling in the food industry: A case study. *European Journal of Operational Research*, 87(3):564–573, 1995.

F. Tercinet, C. Lente, and E. Néron. Mixed satisfiability tests for multiprocessor scheduling with release dates and deadlines. *Operations Research Letters*, 32 (4):326–330, 2004.

N. Trautmann and C. Schwindt. A cyclic approach to large-scale short-term planning in chemical batch production. *Journal of Scheduling*, 12(6):595–606, 2009.

D.P. Van Donk. Make to stock or make to order: The decoupling point in the food processing industries. *International Journal of Production Economics*, 69(3):297–306, 2001.

G. Wan and B.P.C. Yen. Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties. *European Journal of Operational Research*, 142(2):271–281, 2002.

K.C. Ying and H.M. Cheng. Dynamic parallel machine scheduling with sequence-dependent setup times using an iterated greedy heuristic. *Expert Systems with Applications*, 37(4):2848–2852, 2010.