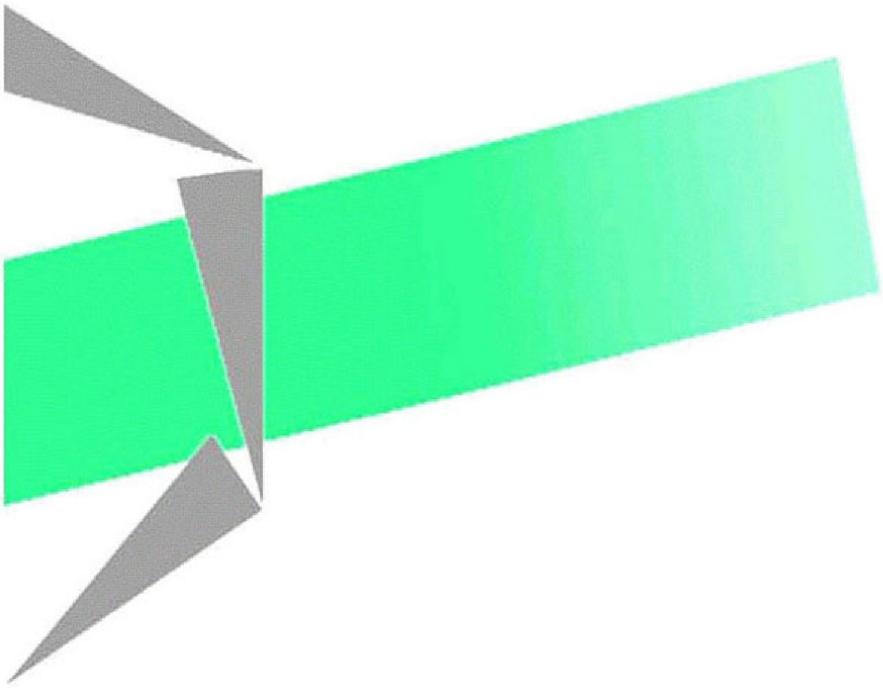
Les cahiers
Leibniz



Schedulling of washing operations in a hospital sterilization service for minimizing the mean predisinfection excess time of medical devices

Onur Ozturk, Maria Di Mascolo, Marie-Laure Espinouse, Alexia Gouin

Laboratoire G-SCOP
46 av. Félix Viallet, 38000 GRENOBLE, France
ISSN : 1298-020X

n° 190

November 2010

Site internet : http://www.g-scop.inpg.fr/CahiersLeibniz/

SCHEDULING OF WASHING OPERATIONS IN A HOSPITAL STERILIZATION SERVICE FOR MINIMIZING THE MEAN PRE-DISINFECTION EXCESS TIME OF MEDICAL DEVICES

Onur OZTURK[(1)], Maria DI MASCOLO[(1)], Marie-Laure ESPINOUSE[(1)], Alexia GOUIN[(2)]

(1) Laboratoire G-SCOP (Grenoble-Science pour la Conception, l'Optimisation et la Production) UMR5272, CNRS, Grenoble INP, UJF, Grenoble-France

46, avenue Félix Viallet - 38031 Grenoble Cedex 1 - France

name.surname@g-scop.grenoble-inp.fr

(2) GIPSA-lab (Laboratoire Grenoblois de l'Image, de la Parole, du Signal et de l'Automatique) UMR 5216, CNRS, Grenoble INP, UJF, Grenoble-France

961, rue de la Houille Blanche, BP 46F - 38402 St. Martin D'Hères Cedex - France

name.surname@gipsa-lab.grenoble-inp.fr

Abstract

After use in operating blocs, medical devices (MD) are sent to the sterilization service. The sterilization process is composed of various steps. In the washing step, after pre-disinfection, different sets of MD, used for different surgical operations, may be washed together without exceeding washer capacity. It is generally not allowed to split MD sets among several washers. In this paper, we model the washing step as a batch scheduling problem, where MD sets are denoted as jobs having different sizes and different release dates, but equal processing times for the washing. The washing steps of sterilization services generally constitute a bottleneck for the entire sterilization process and long pre-disinfection times may corrode MD. Hence, the decisions for batching the MD sets and launching washing cycles are crucial in order to minimize the waiting time of MD in the washing step. First, we provide a MILP model for the minimization of mean pre-disinfection excess time and then a ε-constraint model for the minimization of a second criterion which is the number of washing

cycles. Afterwards, two heuristics are developed and experimented on the instances inspired from a real case.

Keywords: Hospital sterilization service, batch scheduling, mixed integer linear programming, heuristics

1. Introduction

Hospital sterilization services aim at minimizing all infectious risks due to the use of medical devices in surgical operations. Medical devices can be described as instruments used by surgeons. Sterilization is primordial for the fight against nosocomial infection, and thus, the reuse of medical devices is ensured thanks to the sterilization. The medical devices are firstly used in operating blocs, then sterilized, stored and finally reused in some other surgical operations. Therefore, it is more appropriate to name these instruments as reusable medical devices (or RMD). After each use, sterilization guarantees the desired hygiene level of RMD for other uses in operating blocs. However, there are some other medical devices for only one use.

Beside the sterilization concern, like all other sectors, hospitals face financial difficulties in their services, logistics or purchasing activities. Hence, it is possible to work on some decision and optimization problems for all departments of hospitals; the sterilization department may be one of them.

All RMD foreseen for a surgical operation must be sterilized. Sterilization process is regulated by some quality standards (see AFNOR (2005) for French quality standards of RMD sterilization). The sterilization is a cyclic process (Fig.1), which is composed of several steps. Starting from the use in operating blocs, RMD are sent to the sterilization service and pass the following steps: pre-disinfection, rinsing and washing, verification, packing, sterilization, storage and reuse in operating blocs.

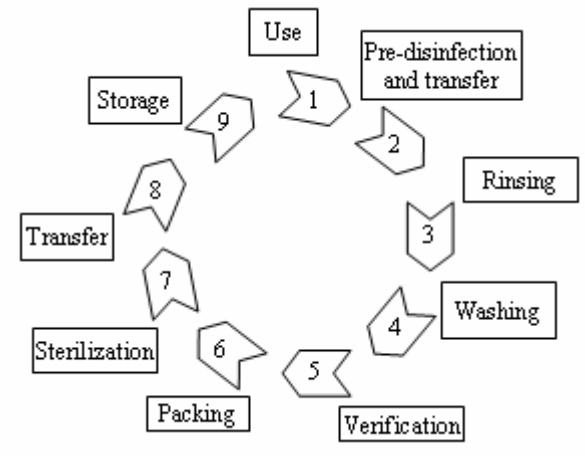


Figure 1. Sterilization Cycle

After use for a surgical operation, RMD are directly put in a substance, which enables pre-disinfection, and are transferred to the sterilization service. There, they are firstly rinsed and washed in washers. The rinsing is done either manually or automatically in washers. In our study, we consider that rinsing is carried out only by washers. After washing, RMD are verified and packed into corresponding boxes. All items must be packed individually or grouped into boxes before sterilization. Afterwards, they are sterilized in so-called "autoclaves", transferred to operating rooms and stored before reuse. Note that long pre-disinfection durations may corrode RMD and the washing step is usually a bottleneck over all the sterilization process **(**Albert *et al.* 2008; EESS, 2007). The ideal RMD pre-disinfection time is about 20 minutes, whereas 15 minutes of pre-disinfection are mandatory, and at most 50 minutes of pre-disinfection could be accepted. In the sterilization services we investigated, manual rinsing is usually in use, while having an automatic rinsing function in the washers.

One reason of this double rinsing is to let RMD wait for the washing without any risk of corrosion. Indeed, as the washing step is usually the bottleneck, the waiting time of RMD for the washing may be long, which may cause longer pre-disinfection times than the ideal one if there is only an automatic rinsing within the washers. Hence, rinsing the RMD manually at their arrivals prevents the risk of corrosion, while waiting for the washing. Then, if the washers have an automatic rinsing function, RMD can be rinsed again in the washers.

It can be possible to ensure the ideal pre-disinfection time without manual rinsing and having only automatic rinsing, if the waiting time for the washers is small enough. Hence, manual rinsing operators could be transferred to other working posts (for example to the packing step, which is always manual). Considering that the ideal pre-disinfection time is 20 minutes, we call “pre-disinfection excess time” the difference between the pre-disinfection time (if it is longer than 20 minutes) and the ideal pre-disinfection time (*i.e.* 20 minutes). Note that the pre-disinfection excess time is zero if it is smaller than (or equal to) 20 minutes.

Therefore, our main aim in this study is to minimize the mean pre-disinfection excess time of RMD sets during the washing step, in order to reach the ideal pre-disinfection time on average, and to see if manual rinsing could be removed from the system. Note that 20 minutes is the ideal pre-disinfection duration for the sterilization service we were in contact with. However, this duration can be easily changed in our solution approaches, if another pre-disinfection duration is considered as ideal.

Secondly, we introduce also an economical criterion in our work, by considering a second objective, which is the minimization of the number of launched washing cycles.

The remainder of this paper is organized as follows. In section 2, we describe the problem of fulfillment of washers and show how this problem can be treated as a batch scheduling problem. In section 3, we give a literature review about batch scheduling problems. Section 4 is dedicated to the solutions methods of the problem. First, a MILP model that minimizes the mean pre-disinfection excess time, with regard to the ideal pre-disinfection duration is developed. Afterwards, a ε-constraint model is proposed in order to minimize the second optimization criterion, *i.e.* number of washing cycles, while respecting the optimal mean pre-disinfection excess time. Following, two heuristics are proposed. Finally, section 5 is dedicated to the experimental design.

2. Problem description

The problem we treat in this paper is the fulfillment of washers in hospital sterilization services. In a typical hospital, there may be several surgical operations all day long. All reusable medical devices used in a surgical operation constitute the reusable medical device set of this operation (RMD set). As one can guess, there may be a great number of different RMD sets in a hospital. Moreover, the number of different types of RMD is generally very high and for a typical hospital, there may be hundreds of RMD references. Because each surgical operation may require different numbers and types of RMD, RMD sets may be of different sizes. For different reasons (surgery beginning times and durations, pre-disinfection procedure, etc.), RMD sets are ready for washing at different moments within a day.

The washing of RMD sets is carried out by washers, which can be identified to identical batching machines. It is possible to put more than one RMD set into a washer, as long as its capacity is not exceeded. The decisions to take are then; which RMD sets to put together in order to constitute a batch for the washing, and when to launch a washing cycle. The characteristic of unequal RMD set ready times for the washing complicates the decision of batching RMD sets. Note that in the washing step, RMD sets are not usually allowed to be split among several washers because of organizational and traceability reasons. In case of splitting, because of the multiplicity of RMD references, it takes a long time to reassemble the boxes of the RMD sets identically in the next steps. Moreover, splitting may cause some mistakes about the reassembling of RMD sets.

As mentioned before, the washing step is generally the bottleneck over all the sterilization process, because of great numbers of surgical operations and capacity constraint of washers. Considering that there is no manual rinsing in the system, the waiting time of RMD sets at the washing step (*i.e.* just before washing operations) is important for the pre-disinfection. Long waiting times at the washing step may cause long pre-disinfection times.

Moreover, the pre-disinfection liquid may corrode RMD. Hence, deciding how to batch RMD sets and when to launch washing cycles become important in order to reach the desired pre-disinfection durations. In a previous study (Ozturk *et al.* [2010a]), we worked on minimizing the makespan of washing operations, in order to cope with the washing step bottleneck. In another study ([Ozturk *et al.* 2010b]), we tried to minimize the sum of RMD set washing completion times. These studies mainly aim at minimizing the buffer size in front of washers and the total washing completion time, considering that there is no manual rinsing. However, the corrosion effect of the pre-disinfection liquid is not taken into account.

2.1 The need for a modeling of the washing step

Remember that after use in operating blocs, RMD are pre-disinfected, rinsed and then washed. In case there is no manual rinsing, the pre-disinfection times may be considerably high. The desired pre-disinfection time for RMD is about 20 minutes while at least 15 minutes of pre-disinfection is mandatory. Each time there is a new RMD set that arrives to the sterilization service; it is put in an already existing batch in order to be washed in an available washer. When there is a new arriving RMD set which does not fit in the actual batch, the batch is closed and a washing cycle is launched with this batch. Note that this strategy is called “online FIFO” because of two reasons: 1- batches are formed with consecutive RMD sets, 2- no information about RMD set arrivals is known in advance. Naturally, applying an online FIFO may cause high pre-disinfection times and does not help to overcome the bottleneck in the washing step. However, a sterilization service may be coordinated with operating blocs, and operating bloc schedules may be used in order to estimate in advance the arrivals of RMD sets to the sterilization service. Thus, in our study, we suppose that RMD set arrivals are known in advance. And so, we would like to show how modeling the washing step may help to take good decisions about the launching of washing cycles in order to reach desired pre-disinfection times when there is no manual rinsing.

In fact, we define the problem of fulfillment of washers as a batch scheduling problem. A scheduling problem is an operations research problem which is to be solved in order to help decision makers to schedule their machines in a good way (or in an optimal way). Thus, we can consider operations research tools as a subsystem embedded in a decision support system used by decision makers [Van Hee and Lapinski, 1988].

In our problem, RMD sets are denoted as jobs and washers as parallel batching machines. More formally, if we make a connection with scheduling problems, we are given a list of jobs $L = (j_1, j_2,\ldots, j_n)$, all of which have the same processing time p, but they may have different pre-disinfection starting times t_j, different release dates r_j, and different sizes w_j. Note that the release date of a job stands for its arrival time to the sterilization service (and also for the time when it is available for washing). The aim is to schedule the jobs on identical parallel batch processing machines without pre-emption. A parallel batching machine is a machine that can simultaneously process more than one job as long as its capacity is not exceeded. Our aim is to explore opportunities for a better grouping of RMD sets for the washing.

2.2 Identification with a batch scheduling problem

Albert *et al*. (2008) simulate different online fulfillment strategies (*i.e.* when the information about job sizes and release dates are not known in advance) for washers, like launching a washing cycle when a predetermined machine capacity is reached, or when RMD wait at most for a predetermined time, in order to minimize the number of launched washing cycles and the RMD waiting time before washing. To the best of our knowledge, they are the first ones who study the problem of fulfillment of washers. Here, we model the problem of fulfillment of washers as a batch scheduling problem. So, in the following, RMD sets are denoted as jobs and washers as parallel batching machines. We make the following assumptions:

- There are N jobs to be processed. The release date and the size of a given job j are denoted by r_j and w_j, respectively. The pre-disinfection starting time of job j is denoted by t_j. The processing times are equal for all jobs and denoted by p.
- All machines have the same capacity B and the size of a job cannot be greater than the machine capacity.
- Several jobs can be batched together, respecting the machine capacity constraint.
- Once a processing for a batch is started, it cannot be interrupted.
- Since it is a parallel batching problem, the processing time of a batch is equal to the longest processing time of jobs in that batch (Potts and Kovalyov, 2000). As all the jobs in our problem have the same processing time, p, the processing time of any batch is p.
- We are not allowed to split a job into several batches.

Inspired from the Graham's notation (Graham *et al.*, 1979), we propose the following notation for our problem: $P \mid p\text{-}batch, r_j, p_j = p, w_j, B \mid (1/N)*\sum f_j$. In this notation, P stands for identical parallel machines, *p-batch* for parallel batching, which means that several jobs may be processed together in a machine at the same time. r_j and w_j denote job release dates and sizes, respectively, $p_j=p$ stands for equal processing times and B for machine capacity. Finally, $(1/N)*\sum f_j$ refers to the objective function. Remember that the desired pre-disinfection time is 20 minutes for any RMD set. The objective function penalizes excessive waiting times before washing. More precisely, pre-disinfection times more than 20 minutes are penalized for every minute exceeding 20. Thus, the formula of f_j is the following: "washing starting time for job j – pre-disinfection starting time for job j - 20 minutes". Negative values of f_j refer to 0 (*c.f.* MILP model given in 4.1.1).

Modeling the washing as a scheduling problem helps us to solve the decision problem of batching RMD sets. Hence, we would like to build an optimization model to be used as a

subsystem in an optimization-based decision support system. Our aim is to help washing operators for the batching of RMD sets. For more information on optimization for decision support systems, we refer the reader to Dutta (1996).

2.3 Problem complexity

Uzsoy (1994) studies the complexity of the problem $1 \mid p\text{-}batch, p_j = p, w_j, B \mid \sum C_j$. In that problem, there is a single parallel batching machine, all job release dates are equal and C_j stands for the processing completion time for job j. He proves that this problem is NP-hard. In fact, this is a special case of our problem. In order to get this special case, let us suppose that in our problem all job release dates and pre-disinfection starting times are equal to 0. Then, the formula of f_j in our problem is reduced to "washing starting time for job j – 20 minutes". Note that adding or subtracting constant numbers from f_j has no numerical effect. Thus, optimizing f_j becomes equivalent to optimizing C_j. Furthermore, if we have 1 machine, after these transformations, we have reduced our problem exactly to $1 \mid p\text{-}batch, p_j = p, w_j, B \mid \sum C_j$. As this special case of our problem is NP-hard, the problem we treat is also NP-hard.

3. Literature review

Scheduling literature is really vast and it is not possible to cover all in this section. However, we would like to say here a few words about batch scheduling problems.

In the batch scheduling literature, two main groups are considered: serial batching and parallel batching (Potts and Kovalyov, 2000). In the serial batching, jobs may be batched if they share the same setup on a machine and the processing time of a batch is equal to the sum of processing times of all jobs in that batch (*i.e.* one job is processed at a time) (Coffman *et al.*, 1990). In parallel batching, it is possible to process several jobs at the same time and the processing time of a batch is equal to the greatest processing time of jobs in that batch (Mathirajan and Sivakumar, 2006). Still, parallel batching problems can be divided into groups according to job sizes or job families. If the problem is subject to job families, all the

jobs in the same family have the same processing time, but they may have different sizes and release dates (Potts and Kovalyov, 2000). If batches are limited to jobs from a single job family, incompatible job families are under consideration, else, compatible job families are considered (or a single job family). For the classification according to job sizes, there are two sub-groups: jobs requiring one unit of machine capacity (Lee *et al.*, 1992), or jobs having different capacity requirements (jobs having different sizes) (Uzsoy, 1994). Note that for this last one, if all the release dates are equal, the problem is equivalent to a bin-packing problem when the optimization criterion is the makespan minimization. Clearly, our problem is a parallel batching problem with different job sizes and a single job family.

For parallel batching problems with different job sizes, the sum of jobs sizes which are put in a batch defines the size of that batch. Machines have a fixed capacity and the batch size should not exceed the machine capacity. Each job should be assigned to just one batch. Moreover, the processing time of a batch is given by the longest processing time of jobs that are put into the batch. We observe that the makespan minimization is the most common objective that is studied in the literature. It is possible to classify the papers on batch scheduling with different job sizes into four groups as follows: 1- single machine and identical release dates, 2- parallel machines and identical release dates, 3- single machine and unequal release dates, 4- parallel machines and unequal release dates. In table 1, we give a brief classification of the literature about parallel batch scheduling problems in presence of a single job family, different job sizes and different job processing times. The articles are grouped according to the previous classification. For the first group, Uzsoy (1994) works on the problem of minimizing the makespan and sum of job completion times. He shows that these two problems are strongly NP-hard. He provides several heuristic algorithms based on the first fit algorithm, which is one of the classical bin-packing algorithms, for the minimization of the makespan and a branch and bound algorithm for the minimization of the

sum of job completion times. Several heuristics are proposed by Ghazvini and Dupont (1998) in order to minimize total flow time. We see the consideration of job weights in Azizoglu and Webster (2000). The weight of a job can be described as the importance of that job. They give a branch and bound algorithm for the weighted sum of job completion times. Zhang *et al.* (2001) provide an approximation algorithm with a worst case ratio of 7/4 for the makespan minimization. They also analyze heuristics proposed by Uzsoy (1994) and compare them with their own solution method.

Table 1. The literature related to parallel batch scheduling problems with different job sizes and a single job family

Group number	Reference	Solution approach	Performance criterion
1	Uzsoy(1994)	Heuristic algorithms, B&B procedure	C_{max}, $\sum C_j$,
1	Ghazvini and Dupont (1998)	Heuristics	$\sum C_j$
1	Azizoglu and Webster (2000)	B&B procedure	$\sum w_j C_j$
1	Zhang *et al.* (2001)	Heuristics	C_{max}
1	Dupont and Dhaenens-Flipo(2002)	B&B procedure	C_{max}
1	Melouk *et al.* (2004)	MILP model, Simulated annealing	C_{max}
1	Kashan *et al.* (2006)	Genetic algorithms	C_{max}
2	Chang et al. (2004)	Genetic algorithm	C_{max}
2	Kashan *et al.* (2008)	Genetic algorithm	C_{max}
3	Li *et al.* (2005)	Heuristic	C_{max}
4	Chung *et al.* (2009)	MILP model, heuristics	C_{max}
4	Damodaran *et al.* (2009)	Meta-heuristic	C_{max}
4	Damodaran and Velez-Gallego(2009)	Heuristic	C_{max}

Objective: C_{max} = total completion time (makespan), $\sum C_j$ = sum of job completion times, $\sum w_j C_j$ = weighted sum of jobs completion times

For the same problem, a branch-and-bound algorithm is developed by Dupont and Dhaenens-Flipo (2002). They present dominance properties that can be used in an enumeration scheme.

Kashan *et al.* (2006) propose two genetic algorithms for the problem. They report that their solution method outperforms the simulated annealing proposed by Melouk *et al.* (2004). For the second group, Kashan *et al.* (2008) report that the genetic algorithm they develop outperforms the simulated annealing given by Chang *et al.* (2004). In the third and the fourth groups of work, unequal release dates are considered. Thus, these studies become more important for our study. For the third group, Li *et al.* (2005) provide an approximation algorithm with a worst case ratio of $2+\varepsilon$ where ε can be arbitrarily small. Finally, for the last group, Chung *et al.* (2009) propose a MILP model and a heuristic approach. Their heuristic uses a first parameter in order to define a time horizon in which jobs are selected to be batched, and a second parameter to define the desired fullness ratio of batches. They experiment the heuristic with different values of these parameters. Damodaran *et al.* (2009) develop a "Greedy Randomized Adaptive Search Procedure (GRASP)". They report that the GRASP approach guarantees the optimal solution for small instances and performs better than the heuristic proposed by Chung *et al.* (2009). Finally, Damodaran and Velez-Gallego (2009) propose a constructive heuristic. This heuristic operates by first determining a time horizon, and then it solves a 0-1 knapsack problem to select the jobs to be batched. They experiment the MILP model and heuristic given by Chung *et al.* (2009) and the GRASP approach developed by Damodaran *et al.* (2009). It is reported that their heuristic outperforms other heuristics in the literature and gives results close to those of the GRASP method. Note that our problem is similar to the problems treated in the fourth group. However, we have pre-disinfection starting times, equal job processing times and the objective function we minimize is different from makespan minimization. We would say that the objective function we treat is more similar to the minimization of "sum of job completion times" than to the makespan. To the best of our knowledge, neither the problem of "minimizing the sum of job completion times" nor our problem has been treated in the literature yet.

In this paper, we first provide a mixed integer linear programming model (MILP) that minimizes the average pre-disinfection excess time. Afterwards, a ε-constraint model is presented. That model implements first the previously spoken MILP model, and then minimizes the number of batches, respecting the optimal value given by the MILP. Thirdly, a compound algorithm is developed which works in an inverse way to the ε-constraint model. Finally, a heuristic is presented for instances which can not be solved with the previous solution methods.

4. Solution approaches for the problem

In this section, we provide exact and heuristic algorithms for our problem.

4.1. Exact solution methods

We first provide a MILP (mixed integer linear programming) model in order to minimize the mean pre-disinfection excess time of RMD sets at the washing step. The MILP model is then going to be used in a ε-constraint approach which has the batch number minimization as second objective. The ε-constraint model implements a second MILP model. Thus, let us call these MILP models $MILP_{time}$ and $MILP_{nbr\_batch}$, respectively.

4.1.1. MILP model for mean pre-disinfection excess time minimization: $MILP_{time}$

Indexes:

j: $1,\ldots,N$ for jobs

k: $1,\ldots,N$ for batches

m: $1,\ldots,M$ for machines

Parameters:

w_j: size of job j

r_j: release date of job j in minutes (time when job j is available for processing, *i.e.* arrival time to the sterilization service)

t_j: pre-disinfection starting time for job j in minutes

N: number of jobs

M: number of machines

B: machine capacity

p: job processing times in minutes

Q: a big number

nb: lower bound on the number of batches ($nb = \left\lceil \sum_{j=1}^{n} w_j / B \right\rceil$)

Decision variables:

x_{jkm}: 1 if job j is processed in batch k and on machine m, 0 otherwise

b_{km}: 1 if batch k is created on machine m, 0 otherwise

s_{km}: ready time of batch k on machine m

s'_j : starting time of processing for job j

f_j: pre-disinfection excess time for job j ($f_j = \max(s'_j - t_j\text{-}20, 0)$)

Let us start by describing the objective function. The pre-disinfection time can be described as the time spent by a job after its pre-disinfection starting until washing, *i.e.* the difference between the washing starting time and its pre-disinfection starting time. As the ideal pre-disinfection time is 20 minutes, jobs having more than 20 minutes of pre-disinfection are penalized. If the difference between the washing starting time and the pre-disinfection starting time of a job is less than or equal to 20 minutes, there is no penalty. In a schedule, at most N batches can be created. So the index of batches varies from 1 to N. Constraint (1) ensures the assignment of all jobs to a batch and to a machine.

$$\textit{Minimize} \qquad 1/N * \sum_{j=1}^{N} f_j \qquad (0)$$

st.

$$\sum_{k=1}^{N}\sum_{m=1}^{M} x_{jkm} = 1 \qquad \forall j \in [1, N] \qquad (1)$$

$$\sum_{j=1}^{N} w_j * x_{jkm} \leq B * b_{km} \qquad \forall k \in [1, N]; \forall m \in [1, M] \qquad (2)$$

$$\sum_{m=1}^{M} b_{km} \leq 1 \qquad \forall k \in [1, N] \qquad (3)$$

$$s_{km} \geq x_{jkm} * r_j \qquad \forall j, \forall k \in [1, N]; \forall m \in [1, M] \qquad (4)$$

$$s_{km} \geq s_{k-1,m} + p * b_{k-1,m} \qquad \forall k \in [2, N]; \forall m \in [1, M] \qquad (5)$$

$$s'_j \geq s_{k,m} - Q * (1 - x_{jkm}) \qquad \forall j, \forall k \in [1, N]; \forall m \in [1, M] \qquad (6)$$

$$s'_j \geq t_j + 15 \qquad \forall j \in [1, N] \qquad (7)$$

$$f_j \geq s'_j - t_j - 20 \qquad \forall j \in [1, N] \qquad (8)$$

$$b_{k;(k \bmod M)+1} = 1 \qquad \forall k \in [1, nb] \qquad (9)$$

$$x_{jkm} \in \{0,1\}; b_{km} \in \{0,1\}; s_{km} \geq 0; f_j \geq 0 \; s'_j \geq 0$$

Constraint (2) is the capacity constraint in case batch *k* is created on machine *m*. Constraint (3) assigns a batch at most to one machine. Constraint (4) sets the ready time of a batch as the greatest release date of jobs in that batch. In case more than one batch is assigned to a machine, (5) ensures a difference at least equal to the execution duration *p*, between the processing of these batches. If $b_{k,m}$ is null, then batch *k* on machine *m* is a dummy batch and its ready time is directly given to the next indexed batch on machine *m* and the processing duration of batch *k* is 0. Thus, another functionality of constraint (5) is to assign a ready time to all batches, from 1 to *N*, even if they are not created, *i.e.* dummy batches get also a ready time with (5). Constraint (6) sets the processing starting time of *j* equal to the processing starting time of the batch in which it is included. As said before, the minimum time required for the pre-disinfection is 15 minutes. Hence, constraint (7) allows the processing of a job at least 15 minutes after the beginning of its pre-disinfection. Remember that the desired duration for the pre-disinfection is between 15 and 20 minutes. If a job has a pre-disinfection

duration between 15-20 minutes, then we do not need to give a penalty to that job. But if that value is more than 20 minutes for a job, in this case we give a penalty for each minute that exceeds 20 minutes of pre-disinfection. Thus, constraint (8) defines the penalization for each job. As all batch processing times are equal, we suppose that, starting from the first batch; batches are placed consecutively on machines. The minimum number of batches is expressed as *nb*, so, at least *nb* batches should be created. By constraint (9), the first *nb* batches are placed consecutively on machines. *"k* mod *M"* determines the machine on which batch *k* will be processed. We add "1" to "(*k* mod *M*)" in order to prevent from having 0 as a machine index and without loss of generality; we place the first batch on machine number 2.

Improvement of the MILP model

In order to improve the resolution time of the MILP model, we will first consider two cases: the case with a single machine and the case with several machines. Let us start with the one machine case. Consider first that *nb* is strictly smaller than *N* (which is generally the case). Then, for the rest of indexes from *nb*+1 until *N*, we can not know if these batches are created or not. But supposing that there are more than *nb* batches to form, we can impose the following constraint: $b_{km} \geq b_{k+1m}$ where *k*=*nb*+1, ..., *N*-1 (constraint 10). For the case with several machines, we can think the same way to add a valid inequality to the model. We see that a batch can be assigned to only one machine. Starting from batch *nb+1*, (because the first *nb* batches have already been assigned to machines), we can predefine the variables b_{km} whose value is equal to 0, and whose value is greater than or equal to 0. Reasoning the same way as for constraint (9), we continue to assign the batches successively on the machines. Hence, the two constraints that we add to the system are: $b_{k,(k \bmod M)+1} \geq 0$ where *k*=*nb*+1, …, *N* (constraint 11) and $b_{k,m'} = 0$ where *k*=*nb*+1, …, *N*; for all $m' \neq (k \bmod M)+1$ (constraint 12). After adding these constraints to the system, the total number of variables is $N^2M + 2NM + 2N$ and the number of constraints is $2N^2M + M(3N - nb - 1) + 4N + nb$.

4.1.2. ε-constraint model for the minimization of the launched washing cycles

In T'kindt *et al.* (2006), the authors propose several methods for multi-criteria scheduling problems, ε-constraint being one of these methods. Our ε-constraint model consists in optimizing one criterion first; then, respecting the value of the first criterion, we try to improve the other criterion. Thus, the proposed ε-constraint model optimizes firstly the mean pre-disinfection excess time using the $MILP_{time}$ model proposed in section 4.1.1. Then, respecting the optimal value found previously, we decrease the number of formed batches by 1 and solve the $MILP_{time}$ model again. We continue this operation until the value of the optimal mean pre-disinfection excess time is increased or the lower bound on the number of batches, *nb*, is reached. However, the trivial lower bound, *nb,* defined in section "4.1.1." is not always reachable: Suppose that we have 5 jobs whose sizes are equal to "Cap/2+α" where α is a very small number while the batch capacity is equal to "Cap". We can easily see that the minimum number of batches for these five jobs is equal to 5. But, summing their sizes, dividing by the batch capacity and then rounding up to the nearest integer leads to 4 batches, which is not a feasible solution. So, in order to calculate a reachable lower bound on the number of batches, we provide a second MILP model.

MILP model for batch number minimization

The MILP model, $MILP_{nbr\_batch}$, we present in this section simply solves a bin-packing problem in order to find a lower bound on the number of batches needed for a given problem (For more information on LP modeling of bin-packing problems, we refer the reader to Carvalho (2002).).

Indexes:	**Decision variables:**
j: *1,…,N* for jobs	x_{jk}: 1 if job j is assigned to batch k, 0 otherwise
k: *1,…,N* for batches	b_k: 1 if batch k is created, 0 otherwise

Parameters:

w_j: size of job j

N: number of jobs

B: machine capacity

$$Minimize \quad \sum_{k=1}^{N} b_k \qquad (0)$$

st.

$$\sum_{k=1}^{N} x_{jk} = 1 \qquad \forall j \in [1, N] \qquad (1)$$

$$\sum_{j=1}^{N} w_j * x_{jk} \leq B * b_k \qquad \forall j \in [1, N]; \forall k \in [1, N] \qquad (2)$$

$$x_{jk} \in \{0,1\}; b_k \in \{0,1\}$$

In this model, the first constraint ensures the assignment of all jobs to a batch and the second constraint is the capacity constraint in case job j is assigned to batch k. The objective is the minimization of the number of batches. Now, we can give the ε-constraint resolution steps. (The model starts first by implementing $MILP_{time}$, so let us call it $\varepsilon\text{-}constraint_{time}$**)**

The proposed ε-constraint model: $\varepsilon\text{-}constraint_{time}$

1- Solve $MILP_{time}$ and find the optimal mean pre-disinfection excess time.

2- Evaluate the number of batches formed by $MILP_{time}$: NB (NB for number of batches)

3- Calculate the lower bound on the number of batches, nb, with $MILP_{nbr\_batch}$

4- If the number of batches created by $MILP_{time}$ is equal to the lower bound, then end the algorithm. Else, decrease NB by 1, put it as a parameter in $MILP_{time}$ for the number of batches (*i.e.* let the index of batches vary between 1 and NB) and solve $MILP_{time}$

5- If the objective function (the mean pre-disinfection excess time) increases, stop the iteration. Else, go to step 4.

The resolution scheme of the ε-constraint model is shown on figure 2.

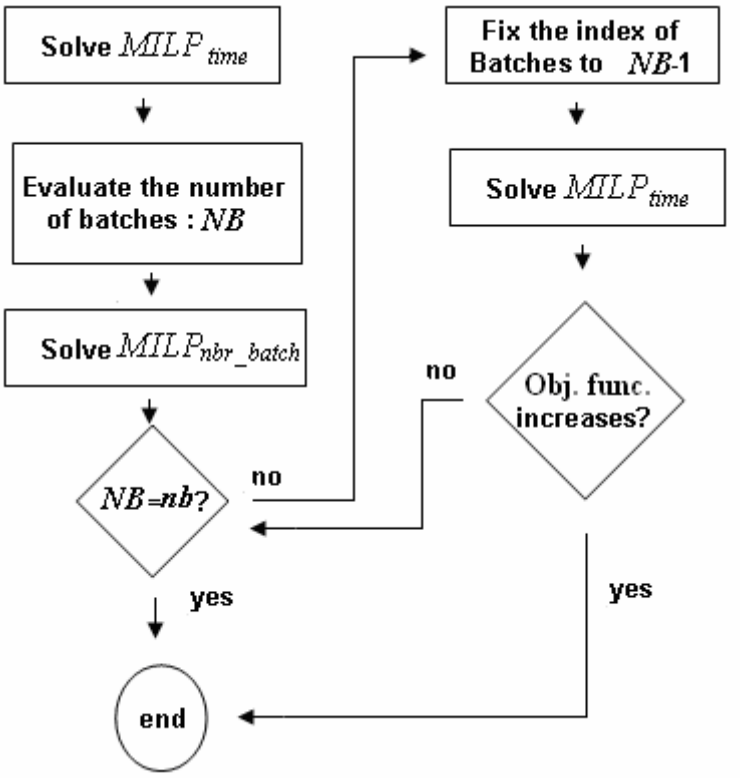


Figure 2. Resolution scheme for the ε-$constraint_{time}$

We see that by the first implementation of $MILP_{time}$, the optimal mean pre-disinfection excess time is found. However, the number of batches created by that first resolution of $MILP_{time}$ could be decreased, respecting the optimal mean pre-disinfection excess time. Thus in each iteration, we decrease the number of batches by one, our aim being to optimize this second criterion respecting the optimal mean pre-disinfection excess time.

4.2. Heuristic approaches

In this section, two heuristics are presented. The first one is constituted of the two MILP models explained in the previous sections. The steps of the first heuristic are reverse of the ε-$constraint_{time}$ model. By starting the resolution of $MILP_{time}$ with a small number of batches, we aim to have a smaller resolution time than the case where the index of jobs varies from 1 to N. Afterwards; we present a second heuristic for larger instances of the problem.

4.2.1. Compound MILP model (*CompA*)

Inspiring from the resolution scheme presented in Chung *et al.* (2009), we present a similar algorithm for our problem. This method is a mirror like of the ε-$constraint_{time}$. In each

iteration, we are allowed to degrade the second optimization criterion, *i.e.* the number of batches, in order to develop the first optimization criterion, *i.e.* the mean pre-disinfection excess time. The algorithm consists first in finding a lower bound on the number of batches, say *nb*, for a given problem. Then, this value is assigned to the upper bound for the number of batch index in $MILP_{time}$. This way the variable *k* in the $MILP_{time}$ does not vary from 1 to *N*, but from 1 to *nb*. After finding the optimal mean pre-disinfection excess time, *nb* is increased by one and then the $MILP_{time}$ model is solved again. This iteration continues until the value of the mean pre-disinfection excess time is no more improved. Note that for finding the *nb*, we use $MILP_{nbr\_batch}$. Our aim with this heuristic is to have a more reasonable computational time for the resolution of $MILP_{time}$, than we had in $\varepsilon\text{-}constraint_{time}$. It is obvious that the probable numbers of batches fall into a range of 1 to *N* where *N* is the number of jobs in the problem. Thus, starting the resolution of $MILP_{time}$ with *nb* batches, rather than *N* batches, lets a smaller solution space for the $MILP_{time}$ model and hence we may gain in the resolution time. The resolution scheme for the compound heuristic is presented on figure 3.

Compound Algorithm (*CompA*)

1- Set the initial solution for the mean pre-disinfection excess time (objective function) equal to a very big number

2- Solve $MILP_{nbr\_batch}$ in order to find a lower bound on the number of batches, *nb*.

3- Set *N'* = *nb*

4- Apply *N*' as the upper bound for the index of batches in $MILP_{time}$ (*i.e.* let the index of batches vary between 1 and *N'*)

5- Solve $MILP_{time}$

6- If there is a decrease in the objective function (the mean pre-disinfection excess time), set *N'* = *N'*+1 and go to step 4. Else end the algorithm.

With the compound algorithm, we can not be sure of the optimality for the mean pre-

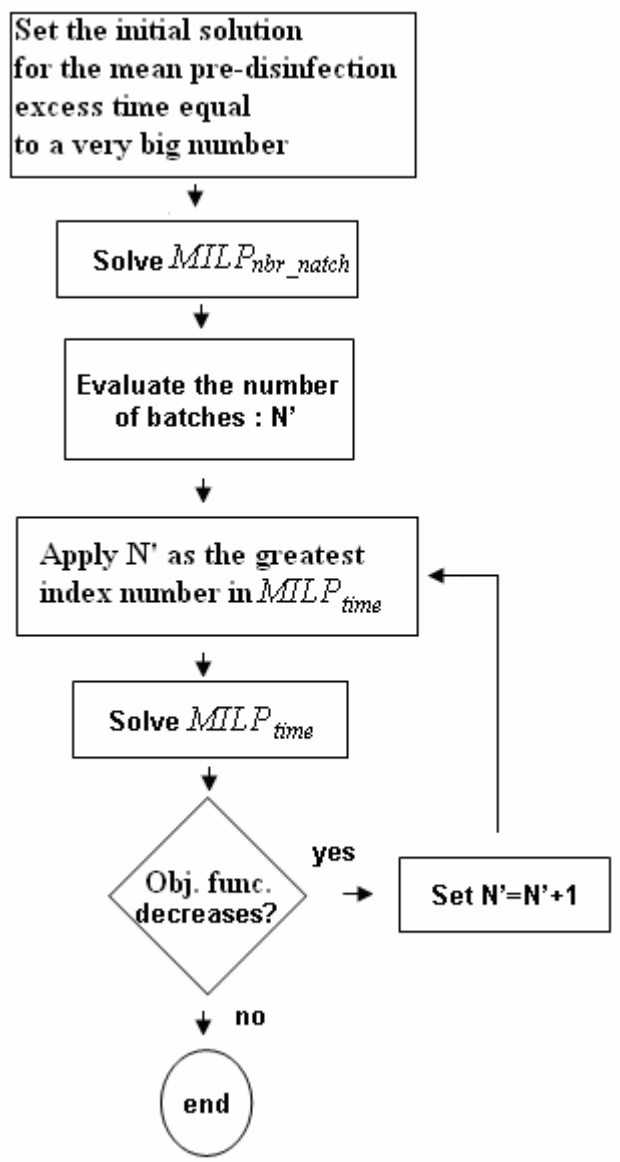


Figure 3. Resolution scheme for the compound algorithm model

disinfection excess time criterion. It is possible to have locally optimal solutions, while with greater number of batches; the mean pre-disinfection excess time may get smaller. Because the algorithm stops with the first increase of the number of batches which does not improve the mean pre-disinfection excess time criterion, we may risk having a local optimum. That is why the method is presented as a heuristic.

4.2.2. Time interval heuristic (*TIH*)

The previous solution methods use only MILP models, which may be inefficient in time for problems of large size; so, we present a polynomial time algorithm in this part. This

heuristic operates by first deciding a time window *t*. And then within each time window, a knapsack problem is solved, where all jobs have the same weight (or importance) but may have different sizes. For the resolution of the knapsack problem, we use a procedure inspired form the “first fit heuristic” of classical bin-packing algorithms (Johnson *et al.* 1974). However, we stop the first fit resolution procedure when only one batch is created. (Note that the bin packing problems aim at minimizing the number of bins, though knapsack problems aim at maximizing the sum of the products of job weight and job size for all the jobs to be put in a bin/batch, i.e; used batch capacity in our problem.) This way a batch is created and it is assigned to the first available machine. The procedure for the creation of a single batch in the given time horizon is given below:

Procedure first fit (PFF)

1- Sort jobs in non-decreasing order of release dates in a list *L*
2- Open a batch
3- Starting from the first element, run through the list *L*, if the job fits the batch, put it in the batch, else, continue with the following element of *L*

In fact, it is not absolutely suitable to call this resolution procedure as a knapsack resolution as we may have fractional job sizes in our problem while knapsack problems have integer job sizes. The difference between PFF and FF is that FF packs all the elements of a list, say *L*. But with PFF, only one batch is created. Note that in PFF procedure, the jobs are added in a batch in increasing order of their release dates. Moreover, the whole job list is scanned to fill the batch. Thus, release dates are taken into account while maximizing the used batch capacity. Concerning the time window *t*, we shall say that *t* is defined by max (r_k ; first machine availability among all machines) where r_k is the *k-th* earliest job release date among the unassigned jobs. The parameter *k* is varied for 1 to *N*, *i.e.* the number of jobs. The heuristic is as follows:

Time Interval Heuristic (*TIH*)

1 Sort jobs in non-decreasing order of release dates r_j: L_0, and set $L_1 = L_0$

2 Set the initial solution, sol_{in}, for the mean pre-disinfection excess time, equal to a big number

3 For k = 1 to N,

3.1 While L_1 is not empty,

3.1.1 If the number of elements in L_1 is smaller than k, set k = number of elements in L_1

3.1.2 Define the time window t = max (release date of the k^{th} element of L_1; first machine availability among all machines)

3.1.3 Apply PFF on jobs whose release dates are smaller than t and erase batched jobs from L_1

3.1.4 Among the batched jobs, find the job whose pre-disinfection beginning time is the biggest, and equal to pre_{max}. Set t' = max (pre_{max} + 15; first machine availability among all machines; biggest job release date in the batch)

3.1.5 Once t' is reached, launch a washing cycle with the batch formed by PFF, and calculate the new availability of the machine on which the batch is processed

End while

3.2 Let sol_{mpt} be the obtained mean pre-disinfection excess time

3.3 If $sol_{in} > sol_{mpt}$, set $sol_{in} = sol_{mpt}$

3.4 Set $L_1 = L_0$

End for

The algorithm defines the length of the time interval according to parameter k in step 3. Each time a batch is formed using PFF heuristic, the parameter t' is calculated for the launching time of the batch. Step 3.1.4 lets us respect the minimum pre-disinfection time which is 15 minutes for all jobs in the batch and also it helps to redefine the processing starting time of a batch according to job release dates in that batch. In each resolution with a different value of k, a new mean pre-disinfection excess time is found. So, thanks to the step 3.3, we choose the smallest value for the objective function.

The complexity of the algorithm can be stated as $O(N^3 \log N)$: We first have a parameter k which varies from 1 to N and for each different value of k, we enter a while loop. For each different value of k, in the worst case, we can suppose that we enter the while loop N

times. The dominant complexity among the steps in the while loop belongs to the step 3.1.3 where we apply PFF algorithm. PFF may have a complexity equal to O(Nlog$N$) in the worst case. Thus, the worst case complexity of the algorithm can be expressed as O(N^3log$N$).

5. Experimental design

In this section, we test the effectiveness of the proposed solution methods. MILP models are solved with the commercial program CPLEX 10.2. The instances we test are inspired from a real case. The data given by a private French hospital are used to create these instances. More precisely, the machine capacity, batch processing time, job sizes and release dates are inspired from the real case. The machine capacity is assumed to be 6 DIN (DIN is a standard measure type for the volume of washers) and the washing duration is 60 minutes. In the real case, there are nearly 36 different RMD set sizes and these may take any value between 0 and 6. We observed the frequencies of different RMD sets sizes within 5 days data and created different job sizes, respecting the proportionality of these frequencies. For the release dates, it is observed that there are on average 4 to 5 RMD sets arrivals to the sterilization service per hour and there may be 0 to 40 minutes of difference between different RMD sets arrivals. For the pre-disinfection beginnings, the difference between the pre-disinfection beginning in the operating blocs and the arrival to the sterilization service of a RMD set may be randomly fixed to 5 to 30 minutes.

Beside random RMD arrivals, in some sterilization services, there is a regular collecting of RMD sets among operating blocs. Hence, we define 2 more different types of job release dates. We consider 20 minutes and 40 minutes of regular job release dates (or RMD set arrivals to the sterilization service). Thus, we have defined 3 types of instances according to RMD set release dates. Let us name them as 1st, 2nd and 3rd types of instances for irregular arrivals, 20 minutes of regular collecting, and 40 minutes of regular collecting, respectively. However, note that in any arrival type, there may be more than one job released

at the same time, *i.e.* more than one RMD set may arrive to the sterilization service at the same time. Moreover, the difference between the pre-disinfection beginning and the arrival to the sterilization for a RMD set may be bigger than 30 minutes for 3rd type.

We group our experiments according to the number of jobs and the number of machines. The number of machines varies from 1 to 4 machines, while job numbers are 10 and 15 for small instances, and 50 for real case inspired instances. For each job number/machine number combination, we test 90 different instances. Thus, for each type of instance, *i.e.* 1st, 2nd and 3rd types, 30 instances are tested in any job number/machine number combination. An Intel Corel 2 Duo, 3 Ghz CPU computer with 3.25 GB Ram is used for all computational experiments. The solution approaches are coded in C++ language, and CPLEX version 10.2 is used to implement the MILP models. The resolution time limit with CPLEX is set to one hour.

5.1. Comparison of mean pre-disinfection excess time for different solution approaches on small instances

First of all, let us show, on different instance types, the average resolution times and the percentage of optimally solved instances with the implementation of the $MILP_{time}$.

Table 2. Average resolution times for different instance types

		Instance type 1		**Instance type 2**		**Instance type 3**	
Nbr. of jobs	**Nbr. of mach.**	**%opt**	**ART**	**%opt**	**ART**	**%opt**	**ART**
10	1	100 %	<1	100 %	≈3.2	100 %	<1
10	2	100 %	<1	100 %	<1	100 %	<1
10	3	100 %	<1	100 %	<1	100 %	<1
10	4	100 %	<1	100 %	<1	100 %	<1
15	1	100 %	595	100 %	373	100 %	906
15	2	100 %	1100	100 %	923	≈ 50%	2560
15	3	100 %	1350	70 %	1373	0 %	>3600
15	4	100 %	1699	0 %	>3600	0 %	>3600

% opt: Percentage of optimally solved instances, **ART:** Average resolution time in seconds

Thanks to table 2, the effectiveness of $MILP_{time}$ can be evaluated. For the cases with 10 jobs, all instances are solved optimally in a very small time. However, for larger instances, it

is not the case. For instances of type 1, all instances are optimally solved within one hour. However, zero instances are optimally solved within one hour for 15 jobs/4machines case of 2nd type instances, and 15 jobs/3-4machines cases of 3rd type instances. Consequently, we can define the resolution limit of the $MILP_{time}$ as these instances. Note that for these instances, CPLEX is stopped at the end of one hour and the solution at that time is recorded.

In the following tables, we first evaluate the results with an operational point of view: Remember that the desired pre-disinfection time is 20 minutes whereas 50 minutes is the limit. Hence for a given instance, we would like to see the mean excess of pre-disinfection time beyond 20 minutes. In tables 3, 5 and 6, we are interested in the average of "mean pre-disinfection excess time" results after solving all instances in a given X jobs/Y machines combination. Thanks to the grouping of instances according to job release dates, the instance characteristics are similar within a given group. Thus, it is reasonable to calculate the average of mean pre-disinfection excess times within a group to see the performance of our solution methods.

We present the results of 1st, 2nd and 3rd types of instances in different tables. In the following tables, we compare the results found by ε-$constraint_{time}$, *CompA* and *TIH*. We do not need to compare the results of $MILP_{time}$ as it is the first step of the ε-constraint model. Table 3 is dedicated to the 1st type of instances. The "*MPT*" column refers to "mean pre-disinfection excess time". We compare the average of *MPT* results found for solved instances by different solution methods.

Note that all instances of 1st instance type are solved optimally by the ε-constraint model in less than one hour. Besides, the results found by the compound algorithm are exactly the same as ε-constraint results for all instances. In regard to *TIH* heuristic results, nearly 70 % of the instances are optimally solved by *TIH* (*i.e.* same results as ε- $constraint_{time}$) over all instances of type 1.

Table 3. Average "mean pre-disinfection excess time" comparison for 1st type instances resolution

Nbr. of jobs	Nbr. of mach.	ε- $constraint_{time}$ *Average of MPT*	*CompA* *Average of MPT*	*TIH* *Average of MPT*
10	1	19.3	19.3	21.4
10	2	2	2	2.3
10	3	0.2	0.2	0.3
10	4	≈0	≈0	≈0
15	1	30.9	30.9	34.8
15	2	6.68	6.68	8.22
15	3	1.72	1.72	2.86
15	4	0.44	0.44	0.5

Average of MPT: Average of mean pre-disinfection excess times in minutes for solved instances

However, we see that for the case of 15 jobs and 1 machine, even in the optimal solution, the average of mean pre-disinfection times is greater than 30 minutes. That leads to an average of more than 50 minutes of pre-disinfection. More clearly, for the case of 15 jobs and 1 machine, 16 instances over 30 have more than 30 minutes of penalization for the mean pre-disinfection times. We said that our aim is to evaluate the results with an operational point of view. Remember that for a RMD set, the difference between the pre-disinfection beginning in the operating blocs and the arrival to the sterilization service may be randomly fixed to 5 to 30 minutes. Thus, even in optimal solutions, it is sometimes impossible to have 0 minutes of penalization. Accordingly, it would be worth seeing the percentage of solved instances having less than 30 minutes of pre-disinfection. For that reason, we show in table 4, the mean pre-disinfection times of the instances solved for the 1st instance type.

Table 4. % of instances having three different pre-disinfection durations for the 1st type instances

Nbr. of jobs	Nbr. of mach.	ε- $constraint_{time}$ ≤30	30 to 50	>50	*TIH* ≤30	30 to 50	>50
10	1	= 0	≈ 81	≈ 19	= 0	≈ 78	≈ 22
10	2	≈ 90	≈ 10	= 0	≈ 85	≈ 15	= 0
10	3	= 100	= 0	= 0	= 100	= 0	= 0
10	4	= 100	= 0	= 0	= 100	= 0	= 0
15	1	= 0	≈ 50	≈ 50	= 0	≈ 44	≈ 56
15	2	≈ 84	≈ 16	= 0	≈ 80	≈ 20	= 0
15	3	= 100	= 0	= 0	= 100	= 0	= 0
15	4	= 100	= 0	= 0	= 100	= 0	= 0

Note that even in the optimal solution, there are no instances having exactly the ideal pre-disinfection time (between 15 and 20 minutes). This is because of pre-disinfection beginning times and RMD arrival times characteristics. Seeing that a RMD set arrives to the sterilization service at most 30 minutes after its pre-disinfection beginning, in table 4, we show the percentage of three different pre-disinfection durations: smaller than 30 minutes, between 30 and 50 minutes and more than 50 minutes. Clearly, when there is more than one washer in the system, most of the instances have less than 30 minutes of pre-disinfection in the optimal solutions. Moreover, 30 minutes of pre-disinfection is quite good in an operational point of view. We see also that *TIH* gives again good results regarding optimal solutions. Let us make the same kind of comparison for instances of 2nd and 3rd type. (*c.f.* tables 5 and 6).

Table 5. Average "mean pre-disinfection excess time" comparison for 2nd type instances resolution

Nbr. of jobs	**Nbr. of mach.**	***ε-constraint$_{time}$*** *Average of MPT*	***CompA*** *Average of MPT*	***TIH*** *Average of MPT*
10	1	32.5	32.5	44.2
10	2	5.8	5.8	7.8
10	3	0.3	0.3	0.4
10	4	0	0	≈0
15	1	36	36	41
15	2	9.5	9.5	12
15	3	5.9	5.9	6.3
15	4	5.3	5.3	5.7

Average of MPT: Average of mean pre-disinfection excess times in minutes for solved instances

Table 6. Average "mean pre-disinfection excess time" comparison for 3rd type instances resolution

Nbr. of jobs	**Nbr. of mach.**	***ε-constraint$_{time}$*** *Average of MPT*	***CompA*** *Average of MPT*	***TIH*** *Average of MPT*
10	1	17	17	18.8
10	2	4.5	4.5	5
10	3	3.11	3.11	3.16
10	4	3.04	3.04	3.04
15	1	43	43	53
15	2	15.9	15.9	16.7
15	3	12.9	12.9	13.3
15	4	12.8	12.8	12.8

Average of MPT: Average of mean pre-disinfection excess times in minutes for solved instances

For 2nd and 3rd types of instances, we see that the average performance of the *TIH* heuristic is quite good. For the 2nd type, *TIH* finds the same solution as the ε-constraint method nearly for 65% on all tested instances. This ratio is equal to 85% for the 3rd type of instance. However, even for the optimal solution, 1 machine cases lead to a mean pre-disinfection excess of more than 30 minutes in 2nd type instances. In the 3rd type of instances, we observe the same result for 15 jobs/1 machine case. Thus, it would not be convenient to remove manual rinsing from the system if there is only one machine (washer) in the system. Note that for all instances of 2nd and 3rd type, the results of the "compound algorithm" are again equal to the results found by the ε-constraint model in 100% of the tested instances. Let us enlarge our analysis and focus on the mean pre-disinfection time values of solved instances for the 2nd and 3rd types.

Table 7. % of instances having three different pre-disinfection durations for the 2nd type instances

Nbr.	**Nbr.**	***ε- constraint$_{time}$***			***TIH***		
of jobs	**of mach.**	**≤30**	**30 to 50**	**>50**	**≤30**	**30 to 50**	**>50**
10	1	= 0	≈ 49	≈ 51	= 0	≈ 42	≈ 58
10	2	≈ 51	≈ 49	= 0	≈ 51	≈ 49	= 0
10	3	≈ 99	≈ 1	= 0	≈ 99	≈ 1	= 0
10	4	= 100	= 0	= 0	= 100	= 0	= 0
15	1	= 0	≈ 30	≈ 70	= 0	≈ 20	≈ 80
15	2	≈ 60	≈ 40	= 0	≈ 56	≈ 44	= 0
15	3	≈ 72	≈ 28	= 0	≈ 66	≈ 34	= 0
15	4	≈ 81	≈ 19	= 0	≈ 73	≈ 27	= 0

Table 8. % of instances having three different pre-disinfection durations for the 3rd type instances

Nbr.	**Nbr.**	***ε- constraint$_{time}$***			***TIH***		
of jobs	**of mach.**	**≤30**	**30 to 50**	**>50**	**≤30**	**30 to 50**	**>50**
10	1	= 0	≈ 18	≈ 82	= 0	≈ 9	≈ 91
10	2	≈ 16	≈ 83	≈ 1	≈ 10	≈ 86	≈ 4
10	3	≈ 25	≈ 75	= 0	≈ 20	≈ 72	≈ 2
10	4	≈ 22	≈ 78	= 0	≈ 20	≈ 80	= 0
15	1	= 0	≈ 0	≈100	= 0	= 0	≈100
15	2	= 0	= 100	= 0	= 0	≈ 92	≈ 8
15	3	= 0	= 100	= 0	= 0	≈ 98	≈ 2
15	4	≈ 3	≈ 97	= 0	= 0	≈ 100	≈ 0

Thanks to tables 7 and 8, we see that one machine cases of any instances lead to long pre-disinfection times even in the optimal solutions. However, with the increasing number of

machines, the pre-disinfection times get closer to a better level. Notice that with 40 minutes of regular collecting, *i.e.* 3rd instance types, the difference between the pre-disinfection beginning and the arrival to the sterilization for a RMD may be considerably high. Thus, in these instances, pre-disinfection times vary rather between 30 and 50 minutes in the optimal solutions.

We saw that ε- $constraint_{time}$ and *CompA* gave exactly the same results for all tested instances. However, the resolution times for these two models are not the same. Let us compare their average resolution times in the following table.

Table 9. Average resolution times in seconds for ε-constraint and compound algorithm models

		Instance type 1		**Instance type 2**		**Instance type 3**	
Nbr. of jobs	**Nbr. of mach.**	***ε-const***	***CompA***	***ε-const***	***CompA***	***ε-const***	***CompA***
10	1	<1	<1	<1	<1	<1	<1
10	2	<1	<1	<1	<1	<1	<1
10	3	<1	<1	<1	<1	<1	<1
10	4	<1	<1	<1	<1	<1	<1
15	1	640	101	441	196	1344	81
15	2	1395	324	1050	312	3289	185
15	3	1975	812	1859	551	>3600	386
15	4	2866	2104	>3600	1368	>3600	543

Table 9 shows the average resolution times for ε-constraint and compound algorithm models. We see that there is a very big gap between their resolution times for 15 jobs instances. Beside the resolution time concern, the optimization criterion, *i.e.* the mean pre-disinfection excess time results of the compound algorithm model are always equal to the results found by the ε-constraint method. Thus, we can conclude that the compound algorithm is more efficient than the ε-constraint model for small instances. However, that does not prove the optimality of the compound algorithm for the mean pre-disinfection excess time criterion. For larger instances, the compound algorithm may find only locally optimum results (with the increasing number of batches the objective function may stay stable while some more increase

in the number of batches may again decrease the mean pre-disinfection excess time). Note that all the resolution times are some milliseconds for *TIH*.

5.2. Comparison for the number of batches criterion on small instances

The second optimization criterion in our problem is the minimization of the number of batches. Computational experiments show that the number of batches formed by the compound algorithm is always equal to the number of batches formed by the ε-constraint model (this is also reasonable as they give the same results for the mean pre-disinfection excess time criterion on the tested instances). Thus, in this section, we compare the number of batches given by $MILP_{time}$, $\varepsilon\text{-}constraint_{time}$ (also compound algorithm results) and *TIH*. We also show the average of minimal number of batches found by $MILP_{nbr\_batch}$. Let us show the average number of formed batches by different solution approaches on 1st, 2nd and 3rd type of instances on the following tables.

Table 10. Comparison for average number of batches on 1st type instances resolution

Nbr. of jobs	**Nbr. of mach.**	$MILP_{time}$ *Av. nbr. of batches*	$\varepsilon\text{-}constraint_{time}$ *Av. nbr. of batches*	*TIH* *Av. nbr. of batches*	$MILP_{nbr\text{-}batch}$ *Av. nbr. of batches*
10	1	4.46	3.5	3.5	3
10	2	6.3	3.8	4.9	3
10	3	7.9	7.3	6.2	3
10	4	8.1	8	7.6	3
15	1	6.9	5.2	5.1	5
15	2	9.5	7	7.6	5
15	3	11.8	10	9.9	5
15	4	13.1	11.6	11.5	5

Table 11. Comparison for average number of batches on 2nd type instances resolution

Nbr. of jobs	**Nbr. of mach.**	$MILP_{time}$ *Av. nbr. of batches*	$\varepsilon\text{-}constraint_{time}$ *Av. nbr. of batches*	*TIH* *Av. nbr. of batches*	$MILP_{nbr\text{-}batch}$ *Av. nbr. of batches*
10	1	4.5	3.8	4.2	3.1
10	2	6.6	4	4.9	3.1
10	3	8.3	4.5	5.5	3.1
10	4	8.7	5	6.1	3.1
15	1	7.2	5.2	5.2	4.9
15	2	8.8	6.8	7.2	4.9
15	3	11.2	8.4	8.4	4.9
15	4	12.6	8.5	8.6	4.9

Table 12. Comparison for average number of batches on 3rd type instances resolution

Nbr. of jobs	**Nbr. of mach.**	$MILP_{time}$ *Av. nbr. of batches*	ε-$constraint_{time}$ *Av. nbr. of batches*	***TIH*** *Av. nbr. of batches*	$MILP_{nbr\text{-}batch}$ *Av. nbr. of batches*
10	1	4.4	3.5	3.6	3
10	2	6.5	3.3	4.3	3
10	3	7.8	3.6	4.5	3
10	4	8.7	3.6	4.5	3
15	1	6.4	5	5.4	4.9
15	2	8	6.2	6.5	4.9
15	3	11.6	6.7	6.8	4.9
15	4	12.3	6.7	6.8	4.9

We understand from tables 10, 11 and 12 that the number of batches is very well improved by the ε-constraint model according to the number of batches given by $MILP_{time}$. Moreover, the *TIH* batch number results are also close to the results found by ε-constraint model. Besides, remember that the compound algorithm was giving the same results for the mean pre-disinfection excess time minimization as the ε-constraint model. The same result is observed for the batch number minimization while the resolution times are considerably smaller with the compound algorithm model. Thanks to tables 10, 11 and 12, we see also that the $MILP_{nbr\text{-}batch}$ results for the batch numbers are almost never two times smaller than the number of batches formed with ε-$constraint_{time}$.

We see that *TIH* heuristic gives good results for the tested instances regarding ε-$constraint_{time}$ and *CompA*. Now, we can widen test instances and try to see the efficiency of *TIH* on instances tested by Damodaran and Velez-Gallego (2009).

5.3. Benchmarking for *TIH* on other instances

As pointed out in Damodaran and Velez-Gallego (2009), the heuristic they propose outperforms other heuristics in the literature. Their problem is the minimization of the makespan. Note that in their model, there is no parameter like the pre-disinfection starting time for jobs, and their objective is different from ours. Thus, it would not be suitable to compare the *TIH* results to the results obtained with their heuristic. However, we can test our

TIH heuristic with instances of Damodaran and Velez-Gallego (2009) in order to see if *TIH* still gives good results. For that purpose, we set the release dates of jobs equal to their pre-disinfection starting times, and remove" pre_{max} + 15" condition in the step 3.1.4 of *TIH*. The objective function we minimize is the mean waiting time (mwt), *i.e.* $(1/N)*\sum(s_j - r_j)$ where s_j is the processing starting time for job j and r_j its release date. So in steps 2 and 3.2 of *TIH*, we can replace the "mean pre-disinfection excess time" by "mean waiting time" and "sol_{mpt}" by "sol_{mwt}". We remove also completely constraint 7 and "-20" from the constraint 8 of $MILP_{time}$. Note that the new objective function does not have any physical significance for our own problem, but we would like to see the numerical performance of *TIH* on some other instances.

In these new instances release dates and job sizes follow a uniform distribution with release dates, r_jЄU[1,αZ], and sizes, w_jЄU[1, $MaxW$] where Z is the sum of job processing times, α a parameter between 0 and 1, and $MaxW$ is a parameter that defines the biggest job size. The instances are defined for different values of these parameters where α = 0.05, 0.10 and 0.50, and $MaxW$=5, 20. The machine capacity is set equal to 20 units and the processing time of a batch is equal to 10 hours. For each combination of α and $MaxW$, we test 10 instances. In the following tables, we compare the results found for different values of α for the mean waiting time and the number of batches.

On the following tables, we see the performance of *TIH* and the optimal solutions found with the ε-constraint model. For all the tested instances, the average for the mean waiting times and the average for batch numbers are shown in tables 13, 14 and 15.We see that *TIH* gives quite good results, both in terms of mean waiting time and number of batches. Moreover, ε-$constraint_{time}$ and *CompA* give again exactly the same results for all the tested instances.

Table 13. Testing of benchmark instances for α equal to 0.05

No. of jobs	No. of mach.	ε-constraint$_{time}$ MWT	ε-constraint$_{time}$ Nbr.batches	CompA MWT	CompA Nbr.batches	TIH MWT	TIH Nbr.batches
10	1	3.2	3.4	3.2	3.4	3.8	2.8
10	2	1.14	4.5	1.14	4.5	1.57	3.5
10	3	0.52	5.7	0.52	5.7	0.9	4.2
10	4	0.23	7	0.23	7	0.7	4.4
15	1	8.2	4.5	8.2	4.5	9.8	2.7
15	2	2.4	6	2.4	6	3.7	2.7
15	3	0.7	5.8	0.7	5.8	1.9	3.1
15	4	0.3	7.3	0.3	7.3	1.6	3.4

MWT: Average of mean waiting times in hours, **Nbr.batches**: Number of batches

Table 14. Testing of benchmark instances for α equal to 0.1

No. of jobs	No. of mach.	ε-constraint$_{time}$ MWT	ε-constraint$_{time}$ Nbr.batches	CompA MWT	CompA Nbr.batches	TIH MWT	TIH Nbr.batches
10	1	3.1	3.9	3.1	3.9	4.1	3.9
10	2	0.8	5.9	0.8	5.6	1.12	5.5
10	3	0.25	7.1	0.25	7.1	0.54	5.8
10	4	0.1	8.4	0.1	8.4	0.5	5.8
15	1	5.7	5	5.7	5	8.4	3.4
15	2	1.6	5.4	1.6	5.4	5.4	4.2
15	3	0.7	6.7	0.7	6.7	4.2	4.5
15	4	0.2	8.1	0.2	8.1	3.3	5.2

MWT: Average of waiting times in hours, **Nbr.batches**: Number of batches

Table 15. Testing of benchmark instances for α equal to 0.5

No. of jobs	No. of mach.	ε-constraint$_{time}$ MWT	ε-constraint$_{time}$ Nbr.batches	CompA MWT	CompA Nbr.batches	TIH MWT	TIH Nbr.batches
10	1	1.45	8.9	1.45	8.9	1.8	7.7
10	2	0.16	9.6	0.16	9.6	0.28	8.9
10	3	≈ 0	8.9	≈ 0	8.9	0.16	8.9
10	4	≈ 0	10	≈ 0	10	0.15	8.9
15	1	3.4	8.7	3.4	8.7	7.2	4
15	2	0.7	11.6	0.7	11.6	2.1	7.4
15	3	≈ 0	12.6	≈ 0	12.6	1.2	9.2
15	4	≈ 0	13	≈ 0	13	1.1	9.8

MWT: Average of mean waiting times in hours, **Nbr.batches**: Number of batches

5.4. Performance of the TIH heuristic on real case inspired instances

Note that for real life problem, we may have far more jobs/RMD sets than 15. For a typical hospital, there may be more than 30 surgical operations per day. That is why previously tested solution methods are not very usable because of long computational times except for the *TIH*. Moreover, we observe that *TIH* gives good results both for mean pre-disinfection excess time minimization, and batch number minimization. Therefore, *TIH* can

be used for bigger sized problems in order to get rapid and efficient results. In this section, we apply *TIH* heuristic on a real situation inspired instance and compare it to a general washer fulfillment strategy that is used in the sterilization service we studied.

As explained in section 2.1, the general strategy for the fulfillment of washers is a "fifo" system in an "online" manner. In that system, no information about the future RMD sets is needed and washers are filled with consecutive RMD sets. Once there is an RMD set that does not fit in the washer which is being filled, a washing cycle is launched. However, ignoring any information about future arrivals of RMD set and batching only consecutive RMD sets may cause long pre-disinfection durations. We would like to show how knowing future RMD set arrivals may help us to take better decisions for the fulfillment of washers. In the previous sections, we have seen that the *TIH* heuristic was giving quite good results. Thus, in this section we compare the *TIH* heuristic to an *online FIFO* system.

We inspire from the data given by a private hospital in Caen, France (Di Mascolo *et al*, 2006). In the sterilization service, there are 4 washers, while the washing step works between 9 a.m. and 19 p.m. (The first RMD sets arrive around 9 a.m.). The RMD sets arrive irregularly at any time of the day, and there are nearly 50 RMD sets sent to the sterilization service. There are on average 4 to 5 RMD sets arrivals per hour. Thus, we prepare an instance for this real case similar to the 1st type of instances explained in the previous section. We have 50 RMD sets to treat and the processing time of a washing cycle is equal to 60 minutes.

In the following table, we see the average of mean pre-disinfection excess times tested on 30 problem instances.

Table 16. Average mean pre-disinfection excess time comparison for *TIH* and *online FIFO*

No. of jobs	**No. of mach.**	*TIH*	*online FIFO*
50	4	1.09 minutes	30.7 minutes

Clearly, the average result of these tested 30 real case inspired instances is much better with the *TIH* than with *online FIFO*. The average of mean pre-disinfection excess times which is found as 1.09 minutes means that the waiting of RMD sets is penalized just with 1.09 minutes on average on all the tested instances. We can guess that this value is very close to the optimal (for any problem, the best possible lower bound for the pre-disinfection excess could be 0 minute). Remember that only durations that are bigger than 20 minutes of pre-disinfection are penalized. Thus, 1.09 minutes of penalization means a pre-disinfection time of 21.09 minutes on average after the starting of pre-disinfection. There is no instance with a pre-disinfection time more than 50 minutes with *TIH* and 15 instances over 30 have pre-disinfection between 15 and 20 minutes. Thus in that case, manual rinsing could be removed from the system. *TIH* guarantees the desired pre-disinfection durations and hence rinsing could be done only automatically in washers. However, with the *online FIFO*, 16 over 30 instances have more than 50 minutes of pre-disinfection. Thus, the penalization is really high with the *online FIFO* system.

Let us expand the comparison and show the number of batches formed by both methods.

Table 17. Average number of batches formed by TIH and FIFO online

No. of jobs	**No. of mach.**	*TIH*	*FIFO online*
50	4	31 batches	16 batches

Obviously, the *online FIFO* is better than the *TIH* for the criterion of number of batches. In fact, it is not surprising to have smaller number of batches with the *online FIFO*. The strategy of the *online FIFO* is to close batches whenever an RMD set does not fit in an existing batch. This is exactly the same of the next fit algorithm which is one of the classical bin packing heuristics. However, minimizing the number of batches, as *online FIFO* does, causes enormous pre-disinfection durations.

For the sake of curiosity, we can extend the comparison supposing regular collecting of RMD sets, *i.e.* with testing 2nd and 3rd instance types.

Table 18. Average mean pre-disinfection excess time comparison for *TIH and online FIFO*

No. of jobs	**No. of mach.**	**2nd type instance**		**3rd type instance**	
		TIH	*online FIFO*	*TIH*	*online FIFO*
50	4	5 min.	36 min.	17 min.	46min.

We see that even with 2nd and 3rd instance types, the *online FIFO* does not give very good results for the mean pre-disinfection excess time of RMD sets. In table 19, we also show the number of formed batches for the same instances.

Table 19. Average number of batches formed by *TIH and online FIFO*

No. of jobs	**No. of mach.**	**2nd type instance**		**3rd type instance**	
		TIH	*online FIFO*	*TIH*	*online FIFO*
50	4	26 batches	19 batches	21 batches	18 batches

Though, the gap between the number of batches formed by *TIH* and online *FIFO* tends to get smaller with the regular collecting. Especially, with 40 minutes of regular collecting of RMD sets, *i.e.* 3rd type of instance, the mean pre-disinfection excess time found by *TIH* is much better than that of *online FIFO* while there is no big difference between their batch numbers.

6. Conclusion and future issues

In this study, we modeled the washing step of a sterilization service as a batch scheduling problem. When there is no manual rinsing before the washing step, the pre-disinfection durations may be more than the desired level. Note that high pre-disinfection durations may corrode RMD. Thus, we aimed at minimizing the mean pre-disinfection excess time of RMD sets at the washing step, when there is no manual rinsing in the system.

Our batch scheduling problem has the following specifications: parallel batching machines which can process several jobs at the same time, job release dates, job sizes, limited machine capacity and equal job processing times. If unequal job processing times are considered, our problem becomes a special case for this new problem. We developed some

exact and heuristic algorithms and tested their effectiveness on real case inspired instances. We see that the *TIH* heuristic works quite good compared to exact solution methods and it requires very small computational times. However, tests on small instances showed that even having the optimal solution with one machine cases, long pre-disinfection times can not be prevented. Thus, when the number of machines is more than 1, exact and heuristic methods work really well for having good pre-disinfection times. We can conclude that for sterilization services having more than 1 machine at the washing step, the manual rinsing can be removed from the system as we can have desired pre-disinfection durations with the proposed resolution methods.

In fact, our main aim in this study is to prepare a framework which can be used by washing operators in order to take good decisions about the batching of RMD sets. Applying a simple online FIFO strategy and combining only consecutive RMD sets for the launching of washer may cause high pre-disinfection times in case there is no manual rinsing. However, we have seen that real life inspired cases are also solved efficiently with the *TIH* heuristic.

Nevertheless, in our work we suppose that future RMD set arrivals are known in advance. Therefore, the next steps of this study may be to develop a coordination system between operating blocs and the sterilization service in order to estimate the arrivals of RMD sets in advance. Then, a final step could be preparing a user interface for the washing operators.

Note that no real life problem is fully deterministic. So, in the future work uncertainties about RMD set arrivals should also be taken into account. For such cases, we can modify the TIH heuristic in order that it becomes flexible if RMD set arrivals do not occur 100% as expected. Each time there is an RMD set that does not obey the foreseen arrival; TIH may be re-executed with the unexpected arrival in order to modify the washing schedule.

Finally, this work may be extended with other objective functions like bi-criteria optimization of number of batches and mean pre-disinfection excess time.

References

AFNOR, 2005, Standard AFNOR FD S98-135. Stérilisation des dispositifs médicaux, Guide pour la maîtrise des traitements appliqués aux dispositifs médicaux réutilisables

Albert F, Di Mascolo M, Marcon E. Analyse de différentes stratégies de remplissage de laveurs dans un service de stérilisation de dispositifs médicaux, 7ème Conférence de Simulation et Modélisation MOSIM'2008, Paris, France

Azizoglu M, Webster S. Scheduling a batch processing machine with non-identical job sizes, International Journal of Production Research 2000; 38; 2173-2184.

Carvalho JM Valério de. LP models for bin packing and cutting stock problems, European Journal of Operational Research 2002, 141(2), 253-273.

Chang PY, Damodaran P, Melouk S. Minimizing makespan on parallel batch processing machines. International Journal of Production Research 2004; 42; 4211–4220.

Chung SH, Tai YT, Pearn WL. Minimizing makespan on parallel batch processing machines with non-identical ready time and arbitrary job sizes. International Journal of Production Research 2009; 47; 5109-5128.

Coffman Jr. EG, Yannakakis M, Magazine MJ, Santos C. Batch sizing and job sequencing on a single machine. Annals of Operations Research 1990; 26; 135-147

Damodaran P, Velez-Gallego MC, Maya J. A GRASP approach for makespan minimization on parallel batch processing machines. Journal of Intelligent Manufacturing 2009. doi: 10.1007/s10845-009-0272-z

Damodaran P, Velez-Gallego MC. Heuristics for makespan minimization on parallel batch processing machines with unequal job ready times. The International Journal of Advanced Manufacturing Technology 2009. doi: 10.1007/s00170-009-2457-1

Di Mascolo M., Gouin A., Ngo Cong K, "Organization of the production of sterile medical devices", INCOM'06, Saint Etienne, 17-19 may 2006

Dupont L, Dhaenens-Flipo C. Minimizing the makespan on a batch processing machine with non-identical job sizes: an exact procedure. Computers & Operations Research 2002; 29; 807–819.

Dutta A., Integrating AI and optimization for decision support: a survey, Decision Support Systems 1996, 18(3-4), 217-226 .

EESS, Enquête Electronique sur les Services de Stérilisation. 2007 (http://www.laspi.fr/ipi)

Ghazvini FJ, Dupont L. Minimizing mean flow time criteria on a single batch processing machine with non-identical job sizes, International Journal of Production Economics 1998; 55(3); 273-80.

Graham RL, Lawler EL, Lenstra JK, Rinnooy Kan AHG. Optimization and approximation in deterministic sequencing and scheduling: a survey, Annals of Discrete Mathematics 1979; 5; 287-326.

Johnson D.S., Demers A, Ullman J.D., Garey M.R. and Graham R.L.. Worst-case performance bounds for simple one-dimensional packing algorithms. SIAM Journal on Computing 1974; 3(4), 299-325

Kashan AH, Karimi B, Jolai F. Minimizing Makespan on a Single Batch Processing Machine with Non-identical Job Sizes: A Hybrid Genetic Approach. EvoCOP, 2006; 135-146

Kashan AH, Karimi B, Jenabi M. A hybrid genetic heuristic for scheduling parallel batch processing machines with arbitrary job sizes. Computers & Operations Research 2008; 35; 1084–1098.

Lee CY, Uzsoy R, Martin-Vega LA. Efficient Algorithms for Scheduling Semiconductor Burn-In Operations. Operations Research 1992; 40(4); 764-775.

Li S, Li G, Wang X, Liu Q. Minimizing makespan on a single batching machine with release times and non-identical job sizes, Operations Research Letters 2005, 33, 157–164

Mathirajan M and Sivakumar A.I. A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor. The International Journal of Advanced Manufacturing Technology 2006; 29; 990-1001.

Melouk S, Damodaran P, Chang P.Y. Minimizing makespan for single machine batch processing with non-identical job sizes using simulated annealing. International Journal of Production Economics 2004; 87; 141-147.

Ozturk O, Espinouse M-L, Di Mascolo M and Gouin A.(2010a) Optimizing the Makespan of Washing Operations Of Medical Devices in Hospital Sterilization Services, IEEE Workshop on Health Care Management, Venice, Italy, 2010

Ozturk O, Di Mascolo M, Espinouse M-L and Gouin A. (2010b) Minimizing the sum of job completion times for washing operations in hospital sterilization services, 8th ENIM IFAC International Conference of Modeling and Simulation (MOSIM'10), Tunisia, 2010

Potts CN and Kovalyov MY. Scheduling with batching: a review. European Journal of Operational Research 2000; 120(2); 228-249

T'kindt V., Billaut J.C., Scott H, Multicriteria Scheduling: Theory, Models and Algorithms, Springer-Verlag New York, 2006

Uzsoy R. Scheduling a single batch processing machine with non identical job sizes. International Journal of Production Research 1994; 32(7); 1615-1635

Van Hee K.M. and Lapinski A., OR and AI approaches to decision support systems, Decision Support Systems 1988, 4(4)n 447-459

Zhang G, Cai X, Lee CY, Wong CK., Minimizing makespan on a single batch processing machine with nonidentical job sizes. Naval Research Logistics 2001; 48; 226–240

Les cahiers Leibniz ont pour vocation la diffusion des rapports de recherche, des séminaires ou des projets de publication sur des problèmes liés au mathématiques discrètes.

Responsables de la publication : Nadia Brauner et András Sebő
ISSN : 1298-020X - © Laboratoire G-SCOP