Optimizing the Makespan of Washing Operations Of Medical Devices in Hospital Sterilization Services

O. Ozturk, M.-L. Espinouse, M. Di Mascolo
Laboratoire G-SCOP (Grenoble-Science pour la Conception, l'Optimisation et la Production)
UMR 5272, CNRS, Grenoble INP, UJF
Grenoble, France
{onur.ozturk; marie-laure.espinouse; maria.di-mascolo}@g-scop.grenoble-inp.fr

A. Gouin
GIPSA-lab (Laboratoire Grenoblois de l'Image, de la Parole, du Signal et de l'Automatique)
UMR 5216, CNRS, Grenoble INP, UJF
Grenoble, France
alexia.gouin@gipsa-lab.grenoble-inp.fr

Abstract— **In this paper, we deal with the problem of minimizing the makespan of washing operations in hospitals sterilization services. After use in operating blocs, reusable medical devices (RMD) are sent to the sterilization service which is composed of various processes. In the washing step, different sets of RMD, used for different operations, may be washed together without exceeding washer capacity. An RMD set must be washed in one cycle and so, we are not allowed to split RMD sets. In this case, we consider a batch scheduling problem where RMD sets may have different sizes and different release dates for washing. Note that if all release dates are equal, the problem is reduced to a bin packing problem. We provide a mixed integer linear programming model which aims at minimizing the makespan of washing operations. We provide and also experiment some heuristics based on classical bin packing algorithms.**

Keywords-Hospital Sterilization Service; Reusable Medical Device; Batch scheduling; Bin Packing problem; Heuristics; Mixed Integer Linear Programming

I. INTRODUCTION

Hospital sterilization services aim at eliminating all infectious risks subject to use of medical devices in surgical operations. At this level, the most important objective is to prevent nosocomial infection. As a primordial operation, sterilization provides the reuse of medical devices in the next coming surgeries. After each use, sterilization guarantees the desired hygiene level of reusable medical devices (RMD) for other utilizations in operating blocs. RMD can be defined as instruments used in surgeries that can be re-used.

Beside sterilization concern, in fact like all other sectors, hospitals are facing increased costs in their services, logistics or purchasing activities. Respecting hygiene conditions, increasing the number of reusable medical devices sterilized per day may help to cut costs about the purchasing of these equipments.

In this paper, we explore opportunities for a better grouping of RMD sets for the washing step in a sterilization service. Because the washing step is generally a bottleneck over all sterilization service [1], our aim is to minimize the total completion time of washing operations by optimizing the fulfilment of washers.

A. Sterilization process

Sterile devices are designed for just one or for several uses. In case a sterile device is used for more than once, we speak of reusable medical device. All RMD foreseen for a surgical operation must be sterilized. Sterilization process is regulated by some quality standards (see [2] for French quality standards for 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 sterilization service and pass the following steps: pre-disinfection, rinsing, washing, verification, packing, sterilization, storage and reuse in operating blocs.

After use for a surgical operation, RMD are directly put in a chemical substance, which enables pre-disinfection, and are transferred to the sterilization service. There, they are firstly rinsed, and then washed in washers. Sometimes, rinsing is included in the washers. After washing, RMD are verified and packed into corresponding boxes. All items must be packed individually or grouped into boxes before sterilization. Then, they are sterilized in machines which are called "autoclave", transferred to operating rooms and stored before reuse.

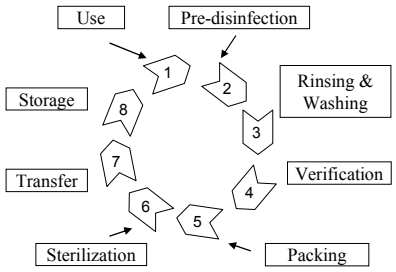

Figure 1. Sterilization Cycle

B. Problem of fulfillment of washers

The number of different types of RMD is generally very high and for a typical hospital, there may be hundreds of RMD references. RMD used for a surgical operation constitutes the RMD set for this surgery. Because each surgery may require different numbers and types of RMD, sets may be of different

978-1-4244-4998-9/10/$25.00 ©2010 IEEE

sizes. For different reasons (surgery characteristics, pre-disinfection procedure, organization, …), RMD sets are ready for washing at different moments.

In the washing step, RMD sets can not be split. It is not allowed to split an RMD set, because in case a splitting is done, it may take a long time to reassemble the RMD set identically due to the multiplicity of RMD references. More than one set may be put into a washer without exceeding machine capacity. The decisions to make are then which sets to put together in order to constitute a batch for the washing, and when to launch a washing cycle.

Different performance criteria may be considered for washing operations, like minimizing the waiting time of RMD sets before the washing step, minimizing the number of launched washing cycles, or minimizing the total washing completion time (Makespan). In our study, we take this latest as the criteria to optimize.

To the best of our knowledge, Albert *et al.* [1] are the first ones to study the problem of fulfilment of washers. In this work, authors test different fulfilment strategies, like launching a washing cycle when a predetermined machine capacity is reached or when RMD wait at most for a predetermined time or a combination of these two strategies.

The problem of fulfilment of washers can be identified to a batch scheduling problem where each job may have different size, different release date and equal processing time. More formally, if we make a connection with scheduling problems, we are given a list of jobs $L = (j_1, j_2, \ldots, j_n)$, which all have the same processing time p but may have different release dates r_j and different sizes w_j. We aim to schedule the jobs on identical parallel batch processing machines without pre-emption. A machine can simultaneously process more than one job as long as its capacity is not exceeded. Our aim is to minimize the makespan. Note that this problem may be seen as a one dimensional bin-packing problem [7] if all the release dates are equal.

C. Litterature Review

In the scheduling literature, batch scheduling problems may be divided into two groups as serial batching and parallel batching [3]. In 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)[4]. In parallel batching, several jobs may be processed at the same time and the processing time of a batch is equal to the greatest processing time of jobs in that batch [5]. Still, it is possible to divide parallel batching problems into two sub-groups, according to job sizes: jobs requiring one unit of machine capacity [6], or jobs having different capacity requirements (jobs having different sizes) [7]. Note that for this last one, if all release dates are equal, it can be possible to define the same problem as a bin-packing problem according to the criteria to optimize [7]. Clearly, our problem is a parallel batching problem with different job sizes.

In parallel batching problems with different job sizes, sum of job sizes that are put into a batch should not exceed machine capacity. Each job is assigned to just one batch. The processing time of a batch is again given by the longest processing time of jobs put into that batch. Uzsoy [7] studies a batch scheduling problem where jobs have equal release dates but different processing times. He works on the problem of minimizing the makespan and total completion time, which are proven to be NP-hard, on one machine. He provides several heuristic algorithms which are based on "first fit" algorithm. Melouk *et al.* [8] propose a MILP model and a simulated annealing (SA) approach to minimize makespan on a single batch-processing machine. Jobs have the same release date but may have different processing times. Ghazvini and Dupont [9] propose various heuristics for the problem of minimizing the mean flow time criteria on a single batching machine. In their problem, jobs have the same release date but may have different processing times. Another criterion that they take into account is that the total number of jobs put on a batch can not exceed a certain limit. Damodaran *et al.* [10] study batch scheduling problem on a single machine in order to minimize the makespan while all release dates are equal. They give a MILP model and propose a simulated annealing (SA) approach. Dupont and Dhaenens-Flipo [11] develop a branch and bound algorithm to minimize the makespan on a single batch processing machine. They determine a number of dominance properties depending on the job sizes and processing times used for the B&B procedure. Zhang *et al.*[12] provide two approximation algorithms for the cases when big sized jobs have the longest processing times and when job sizes and processing times are arbitrary. They also analyse heuristics proposed in [7] and compare them with their own algorithms. Xu and Bean [13] give a MILP model and a genetic algorithm which minimizes the makespan of scheduling operations on parallel non-identical batch processing machines. In their problem jobs have different processing times while all release dates are equal. Li *et al.* [14] study the problem of minimizing the C_{max} on a single batching machine in the presence of different job processing times, release dates and job sizes. They present an approximation algorithm with worst-case ratio $2+\varepsilon$.

The articles cited above do not treat a batch scheduling problem on parallel identical batching machines with different job release dates and different jobs sizes at a time. In this paper, we deal with the problem of minimizing makespan on parallel identical batch processing machines (the washers) subject to job (the RMD) release dates and different sizes.

In section two, we first give a formal description and then a mixed integer linear programming (MILP) model for our problem. In section three, we develop heuristics based on classical bin packing problem heuristics. Section four is dedicated to experimental design. Finally in section five, we compare the proposed approaches with the general washing strategy in sterilization services.

II. PROBLEM DESCRIPTION AND MODELING

In this section, we give a formal description of our problem. Afterwards, the complexity of the problem is given and a mixed integer linear programming approach is explained.

A. Problem description

As explained previously, we identify the problem of fulfilment of washing resources as a batch scheduling problem. So in the following, RMD sets are denoted as jobs and washers as batching machines.

We make the following assumptions:

- There are n jobs to be processed. The release date for job j is denoted by r_j and processing times are equal for all jobs and denoted by p.
- All machines have the same capacity B and the size of a job j is w_j. We assume that the size of a job can not 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 can not be interrupted.
- The processing time of a batch is the longest processing time of jobs included in that batch. Since processing time is the same for all jobs, the processing time of any batch is p.
- We are not allowed to split a job into several batches.

A three field notation, inspired from Graham's notation [15], may be given for our problem as: $P|p\text{-}batch, r_j, p_j = p, w_j, B|C_{max}$. In this notation, P stands for identical parallel machines, *p-batch* for parallel batching, r_j and w_j for job release date and size, respectively, $p_j = p$ for equal processing times and B for machine capacity. Finally C_{max} refers to the makespan (the length of the schedule) *i.e.* the criteria to optimize.

B. Complexity

Note that for our problem, if all the release dates are equal to zero, then, minimizing the C_{max} on a single machine becomes equivalent to minimizing the number of batches. This special case of our problem can be seen as a bin-packing problem which is proved to be NP-hard [7]. As the special case is NP-hard, the problem we treat is also NP-hard.

C. Mixed Integer Linear Programming Model

The problem under study can be formulated as an integer linear program.

Index:
j : $1,...,N$ for jobs
k : $1,...,N$ for batches
m : $1,...,M$ for machines

Parameters:
w_j : size of job j
r_j : release date of job j
N : number of jobs
M : number of machines
B : machine capacity
p : processing time of jobs
nb : minimal number of batches needed

($nb = \left\lceil \sum_{j=1}^{n} w_j / B \right\rceil$, i.e. the smallest integer bigger than $\left\lceil \sum_{j=1}^{n} w_j / B \right\rceil$)

Decision variables:
x_{jkm} : 1 if job j is executed 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

Mathematical Formulation:

$$Minimize \quad C_{max} \tag{0}$$

subject to

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

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

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

$$b_{k;(k \bmod M)+1} \geq 1 \quad \forall k \in [1,nb] \tag{4}$$

$$s_{km} \geq x_{jkm} * r_j \quad \forall j, \forall k \in [1,N]; \forall m \in [1,M] \tag{5}$$

$$s_{km} \geq s_{k\text{-}1,m} + p * b_{k\text{-}1,m} \quad \forall k \in [2,N]; \forall m \in [1,M] \tag{6}$$

$$C_{\max} \geq S_{N,m} + p * b_{N,m} \quad \forall m \in [1,M] \tag{7}$$

$$x_{jkm} \in \{0,1\}; b_{km} \in \{0,1\}; s_{km} \geq 0; C_{\max} \geq 0$$

Our objective is to minimize the makespan. At most N batches can be created in a schedule. Constraint (1) ensures the assignment of all jobs to a batch and to a machine. Constraint (2) is the capacity constraint in case batch k is created on machine m. Constraint (3) assigns a batch at most on one machine. Note that, as all the batch processing times are equal, in an optimal solution, the batches can be assigned on the machines consecutively, in increasing order of batch ready times. So, (4) assigns the first nb batches consecutively on machines. (Here, k mod M determines the machine on which batch k will be executed. (k mod M)+1 prevents from having 0 as a machine index.) This way we prevent multiple equivalent solutions at least for the first nb batches. Constraint (5) 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, (6) ensures a difference at least equal to the execution duration p, between the starting time of these batches. Another functionality of constraint (6) is to assign a ready time to all batches, 1 to N, even if they are not created, *i.e.* dummy batches get also a ready time with (6). If $b_{k,m}$ is null, then the batch k on the machine m is a dummy batch and its ready time is directly given to the next indexed batch. Finally, constraint (7) defines C_{max}. Note that in our modelling, thanks to use of dummy batches, we do not need to define any precedence constraints between jobs which is generally done by inserting constraints that contain "big M" parameter. Thus, we avoid any possible excessive branching and increased computation that may arise from use of "big M". Note that the MILP model helps us to formalize our problem and enables us to test the efficiency of the heuristics presented in the next section.

III. HEURISTIC APPROACHES

As we have seen before, a special case of the problem when release dates are equal can be modelled as a bin packing problem. In this section, we present heuristics adapted from the classical bin packing heuristics, which are first fit (FF), best fit (BF), worst fit (WF), taking into account job release dates. These heuristics will help us to see the effect of minimizing the

number of batches on minimizing the C_{max}. We also propose a simple first in first out (FIFO) based on job release dates.

A. « Modified Batch First Fit » Heuristic

This heuristic operates by first creating a list of jobs sorted in increasing order of job release dates. Each time a new batch is formed, the previously created list is scanned from its first element, and then we try to insert each job into the created batch, respecting the batch capacity. If there are still un-batched jobs, we create another batch and carry out the same operation until all jobs are batched.

1 Sort the jobs in increasing order of job release dates r_j: L_1
2 While L_1 is not empty
2.1 Choose the item at the head of the list L_1 and place it to the first batch it fits. If it fits in no batch, create a new batch for the item.
2.2 Update L_1.
End while
3 Set ready times of batches as the greatest release date of jobs contained in each batch
4 Order batches consecutively onto machines in increasing order of their ready times.

B. « Modified Batch Best Fit » Heuristic

This heuristic inserts the job that will leave the least amount of space left in a given batch. Each time a new batch is created, a list containing jobs sorted in increasing order by job release dates gives the its first element to the batch.

1 Sort the jobs in increasing order of job release dates $r_j : L_1$
2 Sort the jobs in decreasing order of job sizes $w_j : L_2$
3 While L_1 is not empty
3.1 Put the first job of L_1: j_k in a new batch: B_k
3.2 Erase job j_k from lists L_1 and L_2
3.3 For each of the elements : j_l of L_2
3.3.1 If j_l fits the batch B_k
Then $B_k = B_k \cup j_l$
Erase j_l form lists L_1 and L_2
End if
End for
End while
4 Set ready times of generated batches as the greatest release date of jobs contained in each batch
5 Order batches consecutively onto machines in increasing order of their ready times.

C. « Modified Batch Worst Fit » Heuristic

Instead of the previous one, this heuristic inserts the job that will leave the most amount of space left in a given batch.

1 Sort the jobs in increasing order of job release dates $r_j : L_1$
2 Sort the jobs in increasing order of job sizes $w_j : L_2$
3 While L_1 is not empty
3.1 Put the first job of L_1: j_k in a new batch: B_k
3.2 Erase job j_k from lists L_1 and L_2
3.3 For each of the elements : j_l of L_2
3.3.1 If j_l fits the batch B_k
Then $B_k = B_k \cup j_l$
Erase j_l form lists L_1 and L_2
End if
End for
End while
4 Set ready times of generated batches as the greatest release date of jobs contained in each batch
5 Order batches consecutively onto machines in increasing order of their ready times.

D. FIFO Heuristic

Once jobs are sorted increasingly according to release dates, FIFO heuristic places jobs consecutively into the open batch. If the next job does not fit in, batch is closed and a new batch is opened for the job. For this reason, the FIFO is more sensitive to job release dates but less to the number of batches created according to other algorithms proposed.

1 Sort the jobs in increasing order of job release dates $r_j : L_1$
2 While L_1 is not empty
2.1 Choose the job at the head of the list L_1 and place it to the open batch. If it does not fit, close the current batch and create a new batch for the item.
2.2 Update L_1.
End while
3 Set ready times of generated batches as the release date of jobs contained in each batch
4 Order batches consecutively onto machines in increasing order of their ready times.

IV. NUMERICAL EXPERIMENTATION

Random and real instances were considered to test the effectiveness of heuristics in terms of solution quality. Real instances are inspired from a private French hospital. Four factors are identified for the experimental design: number of jobs, job sizes, release dates and number of machines. We grouped our experiments into sets, based on number of machines and number of jobs. The number of machines varies from 1 to 4 while the job numbers to test are 10, 15, 20, 25, 30, 50 and 70.

We have 4 possibilities for the number of machines and there are 7 different numbers of jobs, which makes 28 different machine/job combinations. Each combination contains 18 different cases due to job sizes and release dates. Jobs size are randomly generated and divided into three cases: smaller than half of machine capacity, inspired from the real case, smaller than machine capacity. Release dates are inspired from the real case or regularly spaced by 20, 30, 40, 50 or 60 minutes. 10 instances were generated for all cases which makes 180 instances for each machine/job combination.

The machine capacity, batch processing time and some instances are inspired from the real case. The machine capacity is assumed to be 6 and the processing time is 60 minutes. In real case inspired instances, job sizes vary uniformly between 0.5 and 4, and there may be 0 to 40 minutes of difference between job release dates.

An Intel Corel 2 Duo, 3 Ghz CPU computer with 3.25 GB Ram is used for all computational experiments. All heuristics are coded in C++ language and CPLEX version 10.2 is used to implement the MILP model.

In case of small and mean sized instances, such as 10, 15, 25 jobs, we compare heuristic results to optimal results. Numerical experiments show that the resolution limit of the MILP model is 25 jobs and 1 machine. Beyond that, we can not have the optimal solution in a reasonable time. Thus, we fix a maximum execution time, 60 minutes, which is equal to processing time of batches, and we make a comparison with the results given within 60 minutes. For large instances such as 30, 50 and 70 jobs, heuristics are compared among themselves.

Table I shows the average gap between CPLEX and heuristic results in terms of solution quality, and also mean execution times in seconds for CPLEX results. The Gap reported for heuristics is given by $[C_{max(heuristic)}-C_{max(Cplex)}]/C_{max(cplex)}$. Heuristic execution times are equal to some milliseconds for small instances.

As we see in table I, for all the cases, FIFO gives better results than other heuristics in terms of solution quality. We can also see the number of times that heuristics give the same solution as the optimal solutions (table II). We understand from tables I and II that with the increasing number of machines, performance of heuristics increases in terms of solution quality as well as reaching the same results as CPLEX gives. Having such fast heuristics is in fact good for real life applications. Indeed,, in sterilization services, we may have from 10 to 50 RMD sets per day while having 2, 3 or 4 washers in the washing step [16].

For large instances of the problem, we compare heuristics among themselves. For each number of machines and number of jobs, we try to see which heuristic gives mostly the best solution. Table III shows the percentage of heuristics reaching the best solution compared to other heuristic results. We see that FIFO is the best among all other heuristics. After FIFO, it is the MBFF which gives better results than others. FIFO and MBFF are more sensitive to job release dates while MBBF and MBWF are rather sensitive to job sizes. The reason why these two heuristics give worse results than others is that batch constitutions may be done with jobs having a large gap between release dates. This causes to have late batch ready times. However, batching with jobs having small gap between release dates prevents to have late batch ready times, thus earlier executions of the batches. Now, we shall see the impact of a deterministic FIFO system on washing operations. In the following paragraph, we explain the actual washing strategy in sterilization services. Then, we show the improvement given by a deterministic FIFO system.

TABLE I. GAP BETWEEN CPLEX AND HEURISTIC RESULTS

		Average gap in % between optimal and heuristics				Average CPLEX resolution time
Number of jobs	***Number of machines***	***FIFO***	***MBFF***	***MBBF***	***MBWF***	
10 jobs	1 mach	4	6.9	6.8	8.1	<1 sec
	2 mach	1.2	2	2	2	<1 sec
	3 mach	0.7	0.7	0.6	0.6	≈ 1,5 sec
	4 mach	≈ 0	0.1	0.1	0.1	≈ 10 sec
15 jobs	1 mach	4.4	8	7.3	9.1	<1 sec
	2 mach	1.2	2.9	3	3.3	≈ 9 sec
	3 mach	0.2	1	0.8	1.2	≈ 95 sec
	4 mach	0.5	1	0.8	0.9	≈ 440 sec
20 jobs	1 mach	11	14	13	19	≈ 1,5 sec
	2 mach	4	7	6	9	≈ 165 sec
	3 mach	1,5	2 ,5	2	3	≈ 810 sec
	4 mach	0,8	1,4	1,4	1,6	≈ 1665 sec
25 jobs	1 mach	11	15	15	22	≈ 95 sec

TABLE II. PERCENTAGE OF REACHING THE SAME AS CPLEX SOLUTION

Number of jobs	**Number of machines**	**FIFO**	**MBFF**	**MBBF**	**MBWF**
10 jobs	1 mach	48	26	22	24
	2 mach	87	78	77	79
	3 mach	93	91	91	91
	4 mach	98	97	96	96
15 jobs	1 mach	48	21	22	22
	2 mach	85	74	73	72
	3 mach	89	83	84	82
	4 mach	91	87	88	87
20 jobs	1 mach	11	14	13	19
	2 mach	69	42	42	44
	3 mach	84	76	73	72
	4 mach	93	84	82	81
25 jobs	1 mach	26	1	1	4

V. IMPACT OF AN OFFLINE FIFO SYSTEM

As a general strategy in some sterilization services, washing is performed by maximizing the charge of washers without any knowledge about next coming RMD sets. The size of a batch is evaluated with each new coming RMD set. If machine size is not exceeded, the job is placed in the batch and the batch is put to wait. Else, a washing cycle is launched for that batch. The only way to decide if a batch will be launched is then to see the next RMD set. This case is the "online" version of a FIFO system. Thus, the only difference between an online and offline FIFO is the moment of closing the batches. Else, batch constructions are the same for these two. Indeed, for the offline FIFO, as the arrival times and sizes of RMD sets are known in advance, it is the last RMD set of a batch that sets the ready time of the batch (ready time for the launch). But in online FIFO, it is the first job that does not fit in the batch that sets the ready time of the batch. However, having the surgery schedule may let us know the order of RMD set arrivals to the sterilization service. Note that, having the precise surgery scheduling in order to know RMD set ready times may require some modifications in the organization of sterilization service.

In the following of this part, we implement an offline FIFO system to a real case. We make a comparison with the actual washing strategy of a real sterilization service. The given time slot is 12 hours per day and there are 45 to 50 RMD sets to be treated each day. The RMD set arrivals per hour can be seen on figure 2. The washing step of the sterilization service contains 4 washers. The comparison is done for a period of 5 days.

TABLE III. BEST SOLUTION GIVING HEURISTICS

		% of giving the best solution			
Number of jobs	***Number of machines***	***FIFO***	***MBFF***	***MBBF***	***MBWF***
30 jobs	1 mach	75	36	35	21
	2 mach	91	82	74	69
	3 mach	97	88	87	85
	4 mach	99	92	92	90
50 jobs	1 mach	76	30	35	17
	2 mach	94	71	71	68
	3 mach	98	86	85	83
	4 mach	98	92	88	87
70 jobs	1 mach	72	36	35	23
	2 mach	92	73	71	72
	3 mach	98	88	86	84
	4 mach	100	92	91	89

In figure 3, we see the profit we can get per day in terms of minutes with an offline FIFO system. On Monday, Wednesday and Thursday, there are RMD set arrivals until the end of the time slot and these days, we can gain from half an hour to nearly one hour by applying an offline FIFO system. However, for Tuesday and Friday, there is nearly no difference between the online and offline version of FIFO. That's because on these days, we observe having large numbers of RMD sets instantly in some moments. Then, some part of RMD sets to treat per day becomes known and while we are treating these RMD sets, there are other RMD set arrivals in time. But, on Monday, Wednesday and Thursday, RMD set arrivals are more uniform. So, the online FIFO system has to wait for each new RMD set to evaluate if a washing cycle will be launched or not.

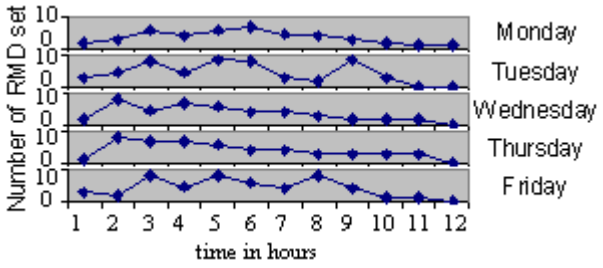

Figure 2. RMD set arrivals per day

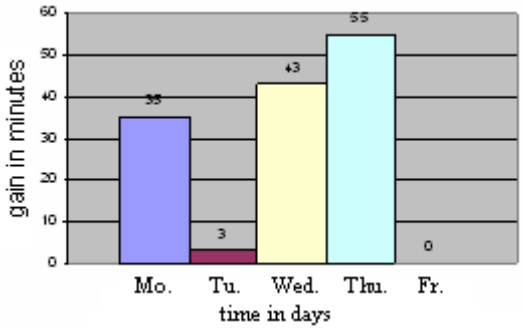

Figure 3. Gain in minutes with an offline FIFO system

VI. CONLUSION AND PERSPECTIVES

In this work, we modeled the washing step of hospital sterilization services as a batch scheduling problem. Firstly, a MILP model is introduced for the problem. Seeing that, the MILP model is insufficient for big instances, we developed heuristics based on classical bin packing heuristics. Theoretical and real case inspired instances were tested. Experiments showed that a FIFO system gives better results than other heuristics. Then, we tested a FIFO system on a real situation to see the development we could get in terms of minutes. We saw that in case of regular RMD set arrivals, we could gain from half an hour to nearly one hour per day.

For future work, the impact of an offline FIFO system can be more deeply studied in case of modifications in the sterilization organization. This research can also be extended by taking into account other optimization measures.

REFERENCES

[1] F. Albert, M. Di Mascolo, E. Marcon,"Analyse de différentes stratégies de remplissage de laveurs dans un service de stérilisation de dispositifs médicaux ", Conférence MOSIM'08, Paris, 2008.

[2] Standard AFNOR FD S98-135 (2005). Stérilisation des dispositifs médicaux – Guide pour la maîtrise des traitements appliqués aux dispositifs médicaux réutilisables.

[3] C.N. Potts, and M. Y. Kovalyov, "Scheduling with batching: a review" European Journal of Operational Research, vol. 120(2), pp. 228-249, January 2000

[4] E. G. Coffman, Jr., M. Yannakakis, M. J. Magazine and C. Santos, "Batch sizing and job sequencing on a single machine", Annals of Operations Research, vol.26, pp.135-147, Dec. 1990

[5] M. Mathirajan, A.I. Sivakumar, "A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor", The Int. Jour. of Advanced Manufacturing Technology, vol. 29, pp. 990-1001, July 2006.

[6] C.Y. Lee, R. Uzsoy, and L. A. Martin-Vega, "Efficient algorithms for scheduling semiconductor burn-in operations", Operations Research , vol. 40(4), pp. 764-775, July-August 1992

[7] R. Uzsoy, "Scheduling a single batch processing machine with non identical job sizes", Int. J. Prod Res, vol. 32(7), pp. 1615-1635, 1994

[8] S. Melouk, P. Damodaran, and P.Y. Chang, "Minimizing makespan for single machine batch processing with non-identical job sizes using simulated annealing", Int. J. Prod. Econ., vol. 87, pp. 141-147, 2004

[9] F. J. Ghazvini, L. Dupont, "Minimizing mean flow time criteria on a single batch processing machine with non-identical job sizes", Int. J. of Production Economics, vol. 55(3), pp. 273-80, 1998

[10] P. Damodaran, K. Srihari, and S. S. Lam, "Scheduling a capacitated batch processing machine to minimize makespan", Robotics and Computer-Integrated Manufacturing, vol. 23(2), pp. 208-216, April 2007

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

[12] G. Zhang, X. Cai, C.Y. Lee, and C.K. Wong, "Minimizing makespan on a single batch processing machine with non-identical job sizes", Naval Research Logistics, vol. 48, pp. 226–240, 2001.

[13] S. Xu and J.C. Bean, "A genetic algorithm for scheduling parallel non-identical batch processing machines", IEEE Symposium on Computational Intelligence in Scheduling, 2007

[14] S. Li, G. Li, X. Wang, Q. Liu, "Minimizing makespan on a single batching machine with release times and non-identical job sizes", Oper. Res. Lett., vol. 33, pp. 157–164, 2005

[15] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey", Ann. Discrete Math., vol. 5, pp.287-326, 1979

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