Applying MILP-based algorithms to automated job-shop scheduling problems in aircraft-part manufacturing

A. M. Aguirre[a], C. A. Méndez[a], A. García-Sánchez[b], M. Ortega-Mier[b]

[a]INTEC (UNL - CONICET), Güemes 3450, 3000 Santa Fe, Argentina.
{aaguirre,cmendez}@intec.unl.edu.ar
[b]ETSI Industriales, (UPM), C/José Gutierrez Abascal 2, 28006 Madrid, Spain.
{alvaro.garcia,miguel.ortega.mier}@upm.es

Abstract. This work presents efficient algorithms based on Mixed-Integer Linear Programming (MILP) for complex job-shop scheduling problems raised in Automated Manufacturing Systems. The aim of this work is to find alternative solution approaches of production and transportation operations in a multi-product multi-stage production process that can be used to solve industrial-scale problems with reasonable computational effort. The MILP model developed must take into account; dissimilar recipes, single unit per production stage, re-entrant flows, sequence-dependent free transferring times and load transfer movements in a single automated material-handling device. In addition, logical-based strategies are proposed to iteratively find and improve the solutions generated over time. These approaches were tested in different real-world problems appeared in the surface-treatment process of metal components in aircraft manufacturing industry.

Keywords: MILP-based algorithm, Automated Manufacturing Systems, Job-shop Scheduling problems, Real-world applications in aircraft-part fabrication process.

1 Introduction

The solution of real-world scheduling problems has greatly attracted the attention of the research and industrial community for many years. In particular, flow-shop scheduling is one of the most treated problems in literature, in which a set of jobs $i=1,2,3,...,N$ has to be transferred through several stages $s=0,1,2,...,M+1$, by using an automated job's transfer device r. In this kind of problems, each job is processed in a sequence of units $j=1,2,3,...,M$, during a flexible processing time, where every machine j can only perform one job at a time, e.g. it is a unary resource in where job preemption is not allowed. Flow-shop problems are usually focused on finding the best processing job sequence that minimizes the completion time of the last job in the system, which is widely known as the makespan *(MK)* criterion.

This type of automated manufacturing systems are commonly in the manufacture of printed circuit boards (PCBs) in electroplating plants and also in the automated wet-etch station (AWS) in semiconductor manufacturing systems. Moreover, many of those methods and tools developed for these problems, such as heuristic and meta-heuristics procedures (Geiger et al.[1];Shapiro and Nuttle[2]; Bhushan and Karimi[3]), full-space MILP models (Phillips and Unger[4]; Bhushan and Karimi[5]; Aguirre et al.[6]; Castro et al.[7]), constraint programming approaches (Zeballos et al.[8]; Novas and Henning[9]) and hybrid

MILP-based formulations (Castro et al.[10]; Aguirre et al.[11]), can be easily adapted, of their original versions, in order to incorporate the major complexities that appear in real-world industrial problems.

This work is focusing in the critical surface-treatment process of large metal components in the aircraft manufacturing industry (Paul et al.[12]). Surface-treatment operations of heavy aircraft-parts are characterized by a higher complexity than typical flow-shop scheduling problems. This particular process involves a series of chemical stages $s=0,1,2,...,L_i$, disposed in a single production line, in which an automated material-handling tool is in charge of all transfer movements of aircraft-parts between different stages, including from/to the input and output buffers disposed at front and at the end of the line.

The principal assumptions of this problem are; a) unique production sequence for each part, b) re-entrant and possible recycle flows to the same unit, c) flexible processing times and d) load transfer times, e) sequence-dependent times for free travelling operations, f) no intermediate storage between stages, g) single production unit per stage, h) a single automated material-handling device with finite storage capacity on a simple rail, i) stringent storage policies "Zero Wait" (ZW) and "Non-Intermediate Storage" (NIS) for each production stage. Moreover, it is important to remark that, transferring times are directly related to the initial and the final position of the device in the production line. A simple example *(MxN=4x3)* which represents the main features of this problem is shown in Fig. 1.

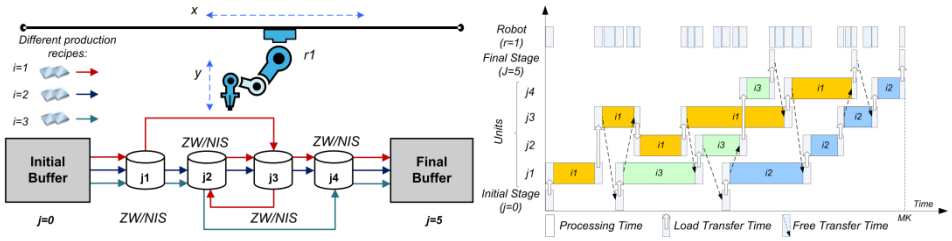

Fig. 1. Automated job shop scheduling problem of 3 different jobs in 4 processing units.

These features force that the material-handling tool, as a robot, must travel large distances, from one unit to another, moving big and weight lots of aircraft-parts, here called jobs, throughout the whole production line, wasting a lot of time and also decrementing the performance of the system.

According to all of this, is easy to see and understand that the daily operation of the material-handling tool in the surface-treatment process represents a complex issue for the decision-maker. In the past, simple heuristic procedures were used to provide a primary solution for this kind of problems, when full-space methods had become untreatable for solving industrial examples, due to the high number of decisions involved in the model. In the other hand, simple heuristic methods, like two-stage approaches (Bhushan and Karimi[5]), are difficult to implement when sequencing decisions of both stages are strongly linked. Thus, any changed in one stage's decisions could turn the problem infeasible if other decisions are not carefully revised. Due to this, sequential approaches, based on mathematical programming and logical-based procedures, that combine robustness and flexibility, seem to be much more appropriated to provide integrated solutions with moderate computational time.

The problem addressed in this work considers the scheduling of processing operations and transportation activities in the system by using a single automated job's transfer device (Robot). Thus, hybrid MILP/logical-based approaches are developed to obtain good-quality results of the entire problem in an iterative manner. The principal aim of these mathematical approaches is to provide good results to complex industrial-scale automated job-shop scheduling problems in a computationally efficient way.

2 General MILP model

The MILP model developed for this work corresponds to an extended version of the previous full-space MILP model presented in Aguirre et al.[13]. This new general approach considers empty transferring times of the robot between two consecutive load transfers. The performance of this approach will be demonstrated solving an industrial scale example of surface-treatments processes in real-world aircraft industry.

2.1 Nomenclature

Parameters.

- I, S, J Set of jobs ($i=i_1,...,N$), stages ($s=s_1,...,L_i$), units ($j=j_0,...,M+1$),
- I^{ins}, I^{rel} Set of inserted jobs and released jobs in the system,
- S_i , L_i Stages belonging to job i and the last stage in the sequence of job i,
- $j_{i,s}$ Production unit that performs job i in production stage s,
- $Seq_{(i)}$, $p_{(i)}$ Production recipe of job i and position of i in the processing sequence,
- $t^{min}_{(i,s)}$, $t^{max}_{(i,s)}$ Minimum and Maximum processing time of job i in stage s,
- $\pi^{min}_{(i,s)}$, $\pi^{max}_{(i,s)}$ Minimum and Maximum loaded transfer time of job i in stage s,
- $\pi^{seq\text{-}dep}_{(i,i',s,s')}$ Free transfer times from loaded transfer i',s' to loaded transfer i,s,
- M_T Large number (Big-M parameter),

Continuous Variables.

- $Ts_{(i,s)}$, $Tf_{(i,s)}$ Start time and Final time of job i in stage s,
- $\pi^{load}_{(i,s)}$, $\pi^{free}_{(i,s)}$ Load and Free Transferring time of task i,s,
- $t_{(i,s)}$ Processing time of task i,s,
- $Pos_{(i,s)}$ Position of task i,s in the transfer sequence of a single Robot,
- $K_{(i,s,i',s')}$ Immediate-precedence variable for transfer sequencing decisions,
- MK Makespan,

Binary Variables.

- $X_{(i,s,i',s')}$ General-precedence variable for job's sequencing decisions,
- $Y_{(i,s,i',s')}$ General-precedence variable for transfer's sequencing decisions,

2.2 Constraints

This MILP formulation takes into account flexible processing times under ZW/NIS policies by Eqs. (1-2), flexible load transfer times by Eqs. (3-5) and also sequence-

dependent free transfer times. Equations (6-8) and (9-11) are proposed to handle sequencing decisions in the same unit by $X_{(i,i',s,s')}$ and transfer's sequencing decisions in different units by $Y_{(i,i',s,s')}$. Then, Eqs. (12-14) are given to determine the specific location of transfers in the transfer sequence by $Pos_{(i,s)}$ parameter. The immediate-precedence variable $K_{(i,i',s,s')}$ for transfer's sequencing decisions in a single resource is provided by Eqs. (15-18). In addition, sequence-dependent free transferring times $\pi^{free}_{(i,s)}$ are determined by Eq. (18). Equation (19-20) are proposed for the partial reduction of the problem size when different jobs have the same production recipe ($Seq_{(i)}$). According to this, jobs with the same recipe must be processed following their lexicographic order. Finally, the objective function, makespan criterion *(MK)*, is presented in Eq. (21).

Flexible timing constraints. Flexible processing times between a minimum and a maximum level are considered by Eqs.(1-2) under stringent ZW policy in each stage *s*.

$$Tf_{(i,s)} = Ts_{(i,s)} + t_{(i,s)} \quad \forall i \in I^{ins}, s \in S_i : (s \neq L_i) \tag{1}$$

$$t^{\min}{}_{(i,s)} \leq t_{(i,s)} \leq t^{\max}{}_{(i,s)} \quad \forall i \in I^{ins}, s \in S_i \tag{2}$$

Flexible transfer constraints. Non-Intermediate Storages (NIS) policy is followed in the system by the robot as stated in Eqs. (3-5). According to this, once the processing time of an immersion process is reached, the production lot must be immediately transferred by the robot from *s-1* to the next stage *s* in its production sequence.

$$Ts_{(i,s)} = Tf_{(i,s-1)} + \pi^{load}{}_{(i,s)} \quad \forall i \in I^{ins}, s \in S_i : (s > s_1) \tag{3}$$

$$Ts_{(i,s)} \geq \pi^{load}{}_{(i,s)} \quad \forall i \in I^{ins}, s \in S_i : (s = s_1) \tag{4}$$

$$\pi^{\min}{}_{(i,s)} \leq \pi^{load}{}_{(i,s)} \leq \pi^{\max}{}_{(i,s)} \quad \forall i \in I^{ins}, s \in S_i \tag{5}$$

Job's sequencing decisions. Decision variable $X_{(i,i',s,s')}$ is proposed in Eq. (6) under the ideas of general-precedence concepts (Méndez and Cerdá[14]). Then, it ensures that processing task *(i,s)* is processed after or before task *(i',s')* in the same production unit *j* by Eqs. (7-8).

$$X_{(i,i',s,s')} = \begin{cases} 1 & \text{if task } i,s \text{ is processed after task } i',s' \text{ in the same unit } j \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

$$Ts_{(i,s)} \geq Tf_{(i',s')} + \pi^{load}{}_{(i,s)} + \pi^{load}{}_{(i',s'+1)} + \pi^{free}{}_{(i,s)} - M_T(1 - X_{(i,i',s,s')})$$

$$\forall i,i' \in I^{ins} : (i > i'), s \in S_i, s' \in S_{i'}, j_{i,s} = j_{i',s'} \tag{7}$$

$$Ts_{(i',s')} \geq Tf_{(i,s)} + \pi^{load}{}_{(i',s')} + \pi^{load}{}_{(i,s+1)} + \pi^{free}{}_{(i',s')} - M_T(X_{(i,i',s,s')})$$

$$\forall i,i' \in I^{ins} : (i > i'), s \in S_i, s' \in S_{i'}, j_{i,s} = j_{i',s'} \tag{8}$$

Transfer's sequencing decisions. Sequencing variables $Y_{(i,i',s,s')}$ for pairs of transfer decisions *(i,s)* and *(i',s')* between different units are modeled by Eqs. (9-11).

$$Y_{(i,i',s,s')} = \begin{cases} 1 & \text{if transfer } i,s \text{ is processed after transfer } i',s' \text{ in different units} \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

$$Ts_{(i,s)} \geq Ts_{(i',s')} + \pi^{load}{}_{(i,s)} + \pi^{free}{}_{(i,s)} - M_T(1 - Y_{(i,i',s,s')})$$

$$\forall i,i' \in I^{ins} : (i > i'), s \in S_i, s' \in S_{i'} \quad (10)$$

$$Ts_{(i',s')} \geq Ts_{(i,s)} + \pi^{load}{}_{(i',s')} + \pi^{free}{}_{(i',s')} - M_T(Y_{(i,i',s,s')})$$

$$\forall i,i' \in I^{ins} : (i > i'), s \in S_i, s' \in S_{i'} \quad (11)$$

Estimate the position of transfers in robot's sequence. The absolute position ($Pos_{(i,s)}$) of transfer task *i,s* in the robot sequence is defined by Eqs. (12-14). This variable is derived by the information of global precedence variables $Y_{(i,i',s,s')}$. Thus, when $Y_{(i,i',s,s')}=1$, $Pos_{(i,s)} > Pos_{(i',s')}$ by Eq. (12) while $Pos_{(i,s)} < Pos_{(i',s')}$ if $Y_{(i,i',s,s')}=0$, as is stated in Eq. (13). Position $Pos_{(i,s)}$ is a positive and should be integer. But, in order to reduce model complexity $Pos_{(i,s)}$ could be defined as continuous variable using Eq. (14). This equation defines the integer bounds for variable $Pos_{(i,s)}$, forcing that it must take only one position in the robot sequence between 1 and the maximum number of transfers in the system.

$$Pos_{(i,s)} \geq Pos_{(i',s')} + 1 - M_T(1 - Y_{(i,i',s,s')}) \quad \forall i,i' \in I^{ins} : (i \geq i'), s \in S_i, s' \in S_{i'} : (i,s) \neq (i',s') \quad (12)$$

$$Pos_{(i',s')} \geq Pos_{(i,s)} + 1 - M_T(Y_{(i,i',s,s')}) \quad \forall i,i' \in I^{ins} : (i \geq i'), s \in S_i, s' \in S_{i'} : (i,s) \neq (i',s') \quad (13)$$

$$1 \leq Pos_{(i,s)} \leq \sum_{i' \in I^{ins}}^{N} \sum_{s' \in S_{i'}}^{L_{i'}} 1 \quad \forall i \in I^{ins}, s \in S_i \quad (14)$$

Immediate-precedence constraints. Using the absolute position information ($Pos_{(i,s)}$) a new variable $K_{(i,i',s,s')}$ is proposed in Eqs. (15-17) to determine the immediate-precedence of transfer *i,s* in the robot sequence. This new variable $K_{(i,i',s,s')}$ is then used to estimate the sequence-depending free transfer times in Eq. (18). $K_{(i,i',s,s',r)}$ is defined as a free variable, but doing some minor changes it can be redefined as a positive or an integer variable in Eqs. (15-18).

$$K_{(i,i',s,s')} = 0 \quad \text{if transfer } i,s \text{ is done just after transfer } i',s' \text{ in the robot} \quad (15)$$

$$K_{(i,i',s,s')} = Pos_{(i,s)} - Pos_{(i',s')} - 1 + M_T(1 - Y_{(i,i',s,s')})$$

$$\forall i,i' \in I^{ins} : (i \geq i'), s \in S_i, s' \in S_{i'} : (i,s) \neq (i',s') \quad (16)$$

$$K_{(i',i,s',s)} = Pos_{(i',s')} - Pos_{(i,s)} - 1 + M_T(Y_{(i,i',s,s')})$$

$$\forall i,i' \in I^{ins} : (i \geq i'), s \in S_i, s' \in S_{i'} : (i,s) \neq (i',s') \quad (17)$$

$$\pi^{free}{}_{(i,s)} \geq \pi^{seq-dep}{}_{(i,i',s,s')} - M_T(K_{(i,i',s,s')}) \quad \forall i,i' \in I^{ins}, s \in S_i, s' \in S_{i'} : (i,s) \neq (i',s') \quad (18)$$

Predefined transfer decisions. Eqs. (19-20) are proposed to reduce the search space of the entire problem without losing optimal results. Thus, sequencing decisions of two production lots *i* and *i'* with the same production recipe $Seq_{(i)} = Seq_{(i')}$, could be defined beforehand by Eq. (19) ensuring that job *i* is produced after job *i'* in unit *j*. In addition, certain transfer sequencing decisions could be predefined by Eq. (20) if these two transfer tasks *i,s* and *i',s'* are following the same production recipe. Thus, if $(i \geq i')$ and $(s \geq s')$ then Eq. (20) enforce that transfer *i,s* will be performed after transfer *i',s'* in the robot sequence.

$$X_{(i,i',s,s')} = 1 \quad \forall i,i' \in I^{ins} : (i > i'), s \in S_i, s' \in S_{i'}, j_{i,s} = j_{i',s'}, Seq(i) = Seq(i') \tag{19}$$

$$Y_{(i,i',s,s')} = 1 \quad \forall i,i' \in I^{ins} : (i \geq i'), s \in S_i, s' \in S_{i'} : (s \geq s'), Seq(i) = Seq(i') \tag{20}$$

Objetive Function: Makespan Minimization. The principal aim of this problem is to maximize the total throughput of aircraft-parts in the production line by minimizing the makespan criterion *MK* as a measure of performance. Then, *MK* is estimated as the completion time of all tasks in the last production stage, as is stated in Eq. (21).

$$MK \geq Ts_{(i,s)} \quad \forall i \in I^{ins}, s \in S_i : (s = L_i) \tag{21}$$

3 MILP/logical-based methods

3.1 Sequential MILP-based algorithm

An MILP-based iterative solution method is presented here for dealing with this complex optimization problem in a sequential manner. Thus, an adapted version of bi-level approach, developed by Bhushan and Karimi[5] and later used by Aguirre et al.[15] for job-shop scheduling problems, is proposed in Fig. 2. The solution algorithm allows solving the whole problem in two stages. In the first stage, a relaxed model is solved considering overestimated transfers and a relaxed solution of this problem is obtained in each iteration. Then, in the second stage, transfers are adjusted according to sequence-dependent variable $K_{(i,i',s,s')}$ and then a reduced model is solved by fixing the job's sequencing $X_{(i,i',s,s')}$ and transfer sequencing decisions $Y_{(i,i',s,s')}$ provided before. Here, it is worth to remark that the best job's sequence of the first stage does not always provide the best result when adjusted transfers are taken into account in the second step. In this case, additional integer cuts are applied by Eq. (22) in order to find alternative job's sequences that provide improved results. In addition, continuous variable $Ts_{(i,s)}$ and $Tf_{(i,s)}$, are copied in each sub-model to accelerate the convergence. The algorithm ends when an iteration limit is reached.

As we can see in Fig. 2, the first stage algorithm could report a feasible result of the entire problem considering relaxed transfers by using Eqs. (1-11, 19-21). In order to provide a feasible result, a MILP model is solved considering overestimated transfer times, estimated as the maximum transfer value. Then, job and transfer's sequencing decisions are fixed and a LP model is solved by considering sequence-dependent transfers in Eqs. (1-21). Then, in order to find improved results, integer cuts are applied by Eq. (22) and the previous job's sequence is removed from the feasible region for the

following iterations. Finally, a new alternative result is found by the model and the best solution is reported.

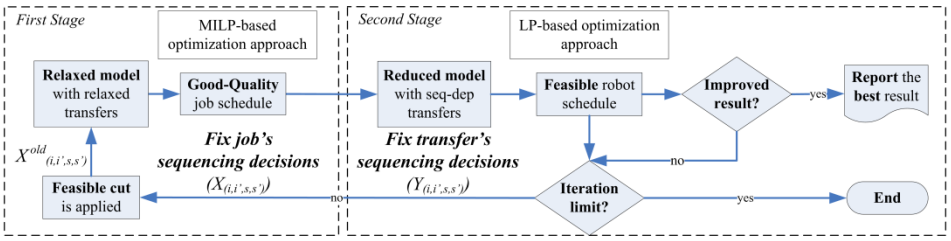

Fig. 2. Pseudo-code of the sequential solution approach

Additional integer cuts for alternative results.

$$\left(\sum_{i,i'>i}\sum_{s\in Si}\sum_{s'\in Si'}\sum_{X^{old}_{(i,i',s,s')}=1} X_{(i,i',s,s')}\right) \le \left(\sum_{i,i'>i}\sum_{s\in Si}\sum_{s'\in Si'}\sum_{X^{old}_{(i,i',s,s')}=1} 1\right) + \left(\sum_{i,i'>i}\sum_{s\in Si}\sum_{s'\in Si'}\sum_{X^{old}_{(i,i',s,s')}=0} X_{(i,i',s,s')}\right) - 1$$

$$\forall i,i' \in I^{ins} : (i > i'), s \in S_i, s' \in S_{i'}, j_{i,s} = j_{i',s'} \tag{22}$$

3.2 Hybrid Constructive-Improvement algorithm

The constructive-improvement algorithm developed in this work is explained as follow in Fig. 3. This iterative solution method allows decompose the problem in small sub-problems that can be solved separately, in a sequential way, consuming short computational time (Aguirre et al.[11]). Each step algorithm consists in 5 phases: initialization, selection procedure, setting binary variables, model resolution and updating parameters. In each iteration (iter) of the constructive step, NSJ jobs are selected, from the Job's List $(i_1,i_2,...,i_N)$, to be inserted in the system I^{ins} by following the NEH ordering rule (Nawas et al.[16]). Thus, jobs with the maximum total production time are selected first to be included into set I^{rel} in order to be scheduled by optimizing variables $X_{(i,i',s,s')}$ and $Y_{(i,i',s,s')}$. Before solving the MILP model, binary variables $X_{(i,i',s,s')}$ and $Y_{(i,i',s,s')}$ of already inserted but non-selected jobs can be fixed. Then, a reduced MILP model is solved obtaining a new sequence p and MK. When all jobs are already inserted, this step finishes reporting an initial feasible schedule $p=(p_1,p_2,...,p_N)$ and the Best makespan result. Starting from this solution, the second step algorithm determinates the jobs to be realized I^{rel} per iteration by chosen the NRJ consecutive jobs in the p sequence. Job's released in the selecting phase are re-scheduled in the system by optimizing $X_{(i,i',s,s')}$ and $Y_{(i,i',s,s')}$ while binary variables of non-released jobs remain fixed. Releasing consecutive jobs allows synchronizing transfer operations efficiently. After solving, the MK result of the MILP model is compared with the Best solution obtained until this iteration. Better solutions are reported and their sequence p is updated. The improvement step finish when no more released jobs can enhance the Best solution found.

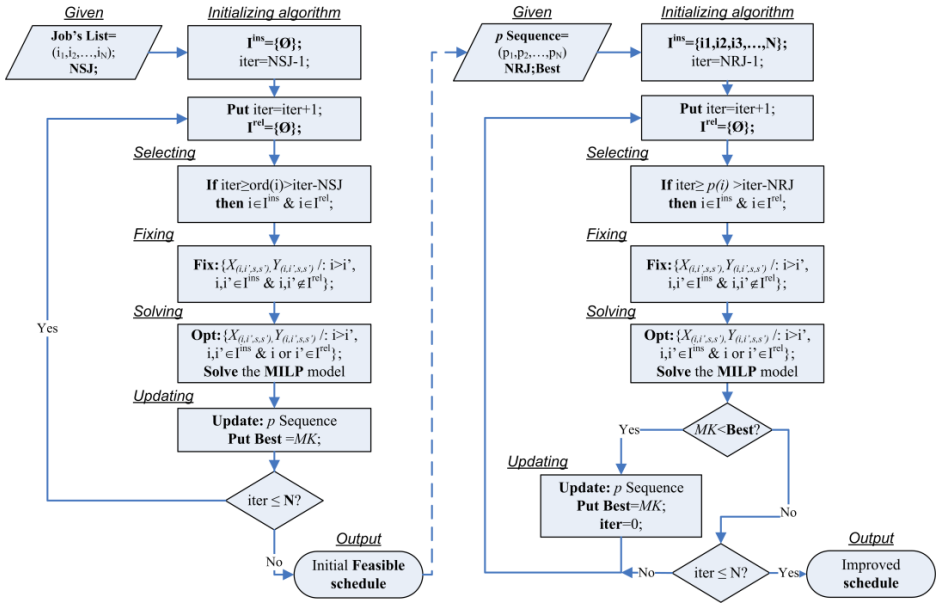

Fig. 3. Pseudo-code of Constructive-Improvement algorithm

4 Computational Analysis

4.1 Motivating Case Study

The following is a small case study proposed by Aguirre et al.[13] in where sequence-dependent transferring times are taken into account in job-shop system. In it, jobs i_1-i_6 must be schedule in j_1-j_{36} units by following specific sequences or recipes $Seq_{(i)}$ which their information is presented in Table 1.

Table 1. Flexible processing and transferring times of task *(i,s)* in unit *j* [min.]

Seq	jobs	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
1	i_1-i_5	j_3:10-15 π:1-6	j_5:5-15 π:1-6	j_4:1 π:1-6	j_5:10-15 π:1-6	j_7:10 π:1-6	j_{25}:1 π:3-6	j_{35}:30-60 π:2-6	j_{36}: 0 π:1-6
2	i_2-i_3-i_4	j_3:10-15 π:1-6	j_5:5-10 π:1-6	j_7:1-5 π:1-6	j_9:5-10 π:1-6	j_{16}:51 π:2-6	j_{35}:30-60 π:3-6	j_{36}: 0 π:1-6	
3	i_6	j_3:10-15 π:1-6	j_5:5-10 π:1-6	j_{35}:30-60 π:3-6	j_{36}: 0 π:1-6	-	-	-	

In this work, free transfer movements are also considered. Free transfer time of tasks *(i,s)* is derived by the information of the current position of the robot along the line which is closely related to the last transfer movement *(i',s')* in robot sequence. According to this, the sequence-dependent transfer time to move job *i* from unit $j_{i,s}$ to unit $j_{i,s-1}$ is estimated by $\pi^{seq\text{-}dep}_{(i,i,s,s)} = 0.05[min.]^{*}abs(j_{i,s} - j_{i,s-1})$.

Table 2 shows the results obtained by the monolithic and the sequential approach presented above. The optimal solution *MK=259.5 min*. is reached by the MILP model in 415 sec. while the sequential algorithm could provide only a feasible result of *301.6 min.* in 40 sec., after 10 iterations of the algorithm.

Table 2. Statistics and Results of the small example analyzed

Units x Jobs	Statistics	Monolithic MILP-based model	MILP model with relaxed transfers	LP model with seq-dep transfers
36x6	Binary Var.	1075	950	-
	Cont. Var.	1887	206	1887
	Equations	9149	2048	9149
	MK *(Gap%)*	**259.5 (0.0%)**	**304**	**301.6**
	CPUtime(s)*	320	3.8	0.6
	Total time*(s)*	415	**40	

**Using Gurobi 5.0 in a PC Intel Core 2 Quad 2,5 GHz with parallel processing in 4 threads.*
***Maximum number of iterations by the algorithm = 10. Time limit per iteration = 120 sec.*

Different values of NSJ/NRJ algorithm parameter are tested. The results reported in Table 3 show that the decomposition algorithm could find an optimal solution *259.5 min.* in less than 60 sec. using certain configurations, e.g. NSJ/NRJ=3/1 or 2/2.

Table 3. Computational results by using our decomposition approach in the small example

Jobs	Constructive Step algorithm		Improvement Step Algorithm NRJ=1		NRJ=2		NRJ=3	
(2,3,1)	**MK**	**CPU**/iter	**MK**(ip)	**CPU**/iter	**MK**(ip)	**CPU**/iter	**MK**(ip)	**CPU**/iter
NSJ =1	**290.55**	**4.8**/6	**290.55**	**21**/6	**259.5**(11)	**58**/21	**259.5**(11)	**127**/7
NSJ =2	**268.55**	**18**/5	**268.55**	**24**/5	**259.5**(3.4)	**38**/15	**259.5**(3.4)	**48**/2
NSJ =3	**268.55**	**49**/4	**259.5**(3.4)	**9**/6	**259.5**(3.4)	**13**/5	**259.5**(3.4)	**26**/4

Using Gurobi 3.0 in a PC Intel Core 2 Quad 2,5 GHz with parallel processing in 4 threads. ip = Improvement percent from the initial solution. iter = Iterations to reach the solution. Time limit per iteration = 120 sec.
**Computational time limit = 3600 sec.*

4.2 Industrial application example

An industrial application example of real-life operations in the aircraft industry is presented in this work. This information was obtained from a previous work of Aguirre et al.[15]. In this example, ten jobs i_1-i_{10} have to be schedule in different units, from j_0-j_{36}, where j_0 and j_{36} represent the input and the output buffer. The information of processing times $t_{(i,s)}$, transferring times $\pi_{(i,s)}$ of every task *(i,s)* and the processing sequences $Seq_{(i)}$ of each job *i* is presented in Table 4. Free transfer times between two consecutive load transfer are estimated as sequence-dependent variable $\pi^{seq\text{-}dep}_{(i,i,s,s)}=0.05[min.]^{*}abs(j_{i,s} - j_{i,s-1})$, according to the absolute distance of departure and arrival units $j_{i,s}$ and $j_{i,s-1}$.

Table 4. Processing and transferring times of task *(i,s)* in unit *j* [min.] of the industrial problem

Seq	Jobs	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}
1	i_1-i_5-i_6-i_9-i_{10}	j_3:12-15 π:1-6	j_5:5-15 π:1-6	j_7:1 π:1-6	j_9:10-15 π:1-6	j_{35}:30-60 π:3-6	j_{36}:0 π:1-6	-	-	-	-
2	i_2-i_3	j_3:12-15 π:1-6	j_5:5-15 π:1-6	j_4:6-10 π:1-6	j_5:5-10 π:1-6	j_7:1 π:1-6	j_9:10-15 π:1-6	j_{35}:30-60 π:3-6	j_{36}:0 π:1	-	-
3	i_4-i_7-i_8	j_3:12-15 π:1-6	j_5:5-15 π:1-6	j_7:8-10 π:1-6	j_8:5-10 π:1-6	j_9:5-10 π:1-6	j_{16}:56 π:2-6	j_{18}:5-10 π:1-6	j_{30}:5-15 π:2-6	j_{35}:30-60 π:2-6	j_{36}:0 π:1-6

Table 4 shows the main statistics and results of the industrial problem analyzed. Here, monolithic model cannot reaches the optimal result after 1 hour, providing a good

initial solution with *4%* of relative gap in 250 seconds. However, the sequential approach can obtain better result *383.75 min.* in very short CPU time of 35 seconds.

Table 5. Statistic and Results of the industrial problem analyzed

Units x Jobs	Statistics	Monolithic MILP-based model	MILP model with relaxed transfers	LP model with seq-dep transfers
36x10	Binary Var.	3494	2926	-
	Cont. Var.	6157	607	6157
	Equations	30868	6928	30868
	MK *(Gap%)*	**384.15 (4.0%)**	**392.4**	**383.75**
	CPUtime(s)*	250	6.2	0.8
	Total time*(s)*	3600	**35	

**Using Gurobi 3.0 in a PC Intel Core 2 Quad 2,5 GHz with parallel processing in 4 threads.*
***Maximum number of iterations by the algorithm = 10. Time limit per iteration = 120 sec.*

The solution obtained by the decomposition algorithm *(378 min.)*, can improve the one reported by sequential approach in *1,5%* after 500 seconds.

Table 6. Computational results by using our decomposition approach in the industrial problem

Jobs	Constructive Step algorithm		Improvement Step Algorithm NRJ=1		NRJ=2		NRJ=3	
(5,2,3)	**MK**	**CPU**/iter	**MK**(ip)	**CPU**/iter	**MK**(ip)	**CPU**/iter	**MK**(ip)	**CPU**/iter
NSJ =1	**395.75**	**101.3**/10	**395.75**(0.0)	**74.1**/5	**378**(4.5)	**1039**/15	**378**(4.5)	**2014**/14
NSJ =2	**391.2**	**167.6**/9	**379.25**(3.0)	**130.2**/14	**378**(3.3)	**965.8**/13	**378**(3.3)	**1331**/7
NSJ =3	**379.4**	**476.6**/8	**378**(0.4)	**91.1**/7	**378**(0.4)	**952.7**/14	**378**(0.4)	**1106**/2

Using Gurobi 3.0 in a PC Intel Core 2 Quad 2,5 GHz with parallel processing in 4 threads. ip = Improvement percent from the initial solution. iter = Iterations to reach the solution. Time limit per iteration = 120 sec.
**Computational time limit = 3600 sec.*

Table 6 shows that the best solution reached by the algorithm is obtained in less than 10 minutes using NSJ/NRJ=3/1. In general, larger values of NSJ/NRJ can provide better results but as expense of more CPU time. Thus, the reported solution starts from a good-quality result in 476 sec. using NSJ=3, and then it is improved *0.4%* until achieving the best result in few minutes. The detailed schedule of the best solution is shown in Fig. 4.

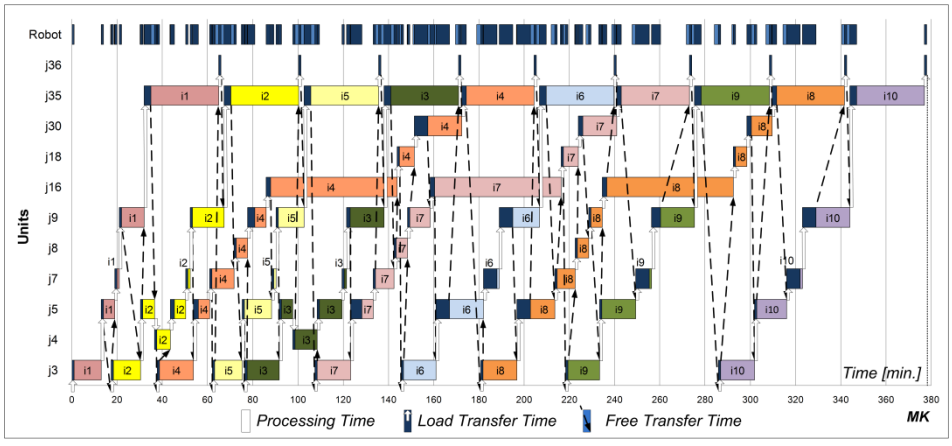

Fig. 4. Solution Schedule of the industrial example

4.3 Testing a daily scheduling problem

This problem, provided by Paul et al.[12], represents a real industrial example at the surface treatment process of aircraft-parts used at the body and wings of airplanes. Here, 12 jobs have to be scheduled following one of the production recipes $Seq_{(i)}$ where the initial and final units are j_0 and j_{20}. Also, information of flexible processing and load transferring times are shown in Table 7.

Table 7. Processing and transferring times of task *(i,s)* in unit *j* [min.] of the tested problem

Seq	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}
1	j_1:23-26 π:1-6	j_2:3-4 π:1-6	j_3:3-4 π:1-6	j_4:15-∞ π:1-6	j_6:3-∞ π:1-6	j_5:15-∞ π:1-6	j_{20}:0 π:2-6	-	-	-	-
2	j_7:2-3 π:2-6	j_8:3-4 π:1-6	j_9:3-4 π:1-6	j_6:3-∞ π:1-6	j_5:15-∞ π:1-6	j_{20}:0 π:2-6	-	-	-	-	-
3	j_{13}:70-73 π:2-6	j_{14}:2-3 π:1-6	j_{15}:2-∞ π:1-6	j_{18}:15-20 π:1-6	j_{17}:3-∞ π:1-6	j_{16}:40-45 π:1-6	j_{11}:2-3 π:1-6	j_{10}:2-∞ π:1-6	j_6:3-∞ π:1-6	j_5:15-∞ π:1-6	j_{20}:0 π:2-6
4	j_{13}:70-73 π:2-6	j_{14}:2-3 π:1-6	j_{15}:2-∞ π:1-6	j_{12}:40-45 π:1-6	j_{11}:2-3 π:1-6	j_{10}:2-∞ π:1-6	j_6:3-∞ π:1-6	j_5:15-∞ π:1-6	j_{20}:0 π:2-6	-	-

Load and free transfer times have been changed to their original version in order to put much more emphasis in robot activities. For this, pick-up and drop-down activities have been estimated in 30 seconds each while the travelling time has been defined in 3 sec./meter. The distance between adjacent baths is 1 meter. Thus, the free travelling time from j_1 to j_2 takes 3 sec. while load travel time is rounded in 1 min. According to this, for small distances, lower than 15 meters, the time to travelling considering pick-up and drop-down movements, is estimated in 1 min. while for medium distances (>15 meters) is defined in 2 minutes. The current product mix of the problem is *(8,2,1,1)*.

Table 8. Statistics and Results of the tested problem analyzed

Units x Jobs	Statistics	Monolithic MILP-based model	MILP model with relaxed transfers	LP model with seq-dep transfers
20x12	Binary Var.	4523	4234	-
	Cont. Var.	8185	353	8185
	Equations	41318	9950	41318
	MK *(Gap%)*	**268.3 (4.2%)**	**279**	**270.55**
	CPUtime(s)*	1660	240	0.7
	Total Time*(s)*	3600	**1155	

**Using Gurobi 3.0 in a PC Intel Core 2 Quad 2,5 GHz with parallel processing in 4 threads.*
***Maximum number of iterations by the algorithm = 10. Time limit per iteration = 120 sec.*

Table 8 and Table 9 show the main results of monolithic, sequential approach and decomposition algorithm for this particular problem. This problem seems to be quite complex due to the number of variables and equations in MILP formulation. Despite of this, results indicate that the monolithic approach is solved up to *4.2%* of relative gap in 1660 sec. while sequential procedure provides similar solution in 1155 seconds. For this case, the decomposition algorithm could provide a good-quality result *(271.25 min.)* after 1650 sec. using NSJ/NRJ=2/1 configuration. It is worth to remark that the decomposition approach provides a poor initial result in less than 500 sec. but an improved solution, between *5-12%*, could be reached by the algorithm using different NSJ/NRJ configurations after 1 hour.

Table 9. Computational results of the tested problem analyzed

Jobs	Constructive Step algorithm		Improvement Step Algorithm NRJ=1		NRJ=2		NRJ=3	
(8,2,1,1)	**MK**	**CPU**/iter	**MK**(ip)	**CPU**/iter	**MK**(ip)	**CPU**/iter	**MK**(ip)	**CPU**/iter
NSJ =1	**308.1**	**150**/12	**288.9**(6)	**3200**/24	**272.4**(11)	**3100**/13	**270.0**(12)	**3300**/11
NSJ =2	**289.45**	**250**/11	**271.25**(6)	**1394**/19	**271.25**(6)	**3600**/34	**270.0**(7)	**3600**/32
NSJ =3	**285.75**	**440**/10	**285.45**	**2173**/47	**271.25**(5)	**3600**/20	**270.0**(5)	**3600**/16

Using Gurobi 5.0 in a PC Intel Core 2 Quad 2,5 GHz with parallel processing in 4 threads. ip = Improvement percent from the initial solution. iter = Iterations to reach the solution. Time limit per iteration = 300 sec. [*] *Computational time limit = 3600 sec.*

5 Conclusions

An MILP-based model and sequential solution approaches were developed for the scheduling of multiple aircraft-parts in the surface-treatment process in the aircraft industry. Results demonstrate that the monolithic model could obtain good-quality results in less than 1 hour for all instances. While, logical-based algorithms, were able to decompose the problem in reduced sub-problems that were solved in moderate CPU time. Thus, a primary solution of these complex scheduling problem have been easily found while extra computational time has been used to improve the solutions obtained over time. Finally, different algorithm parameters were tested in order to find the best configuration, in terms of *MK* and CPU effort, for each particular problem instance.

6 Acknowledgments

Financial support received from CONICET under Grant PIP-2221, from UNL under Grant PI-66-337 and from Erasmus Mundus Scholarship Program is fully appreciated.

7 References

1. Geiger, C.E., Kempf, K.G., Uzsoy, R., 1997. J. of Manufacturing Systems, 16(2), 102-116.
2. Shapiro, G.W., Nuttle, H.L., 1988. IIE Transactions 20(2), 157-167.
3. Bhushan, S., Karimi, I.A., 2003. Ind. Eng. Chem. Res., 42(7), 1391-1399.
4. Phillips, L.W., Unger, P.S., 1976. AIIE Transactions 28(2), 219-225.
5. Bhushan, S., Karimi, I.A., 2004. Comp. Chem. Eng., 28(3), 363–379.
6. Aguirre, A. M., Méndez, C. A., Castro, P. M., 2011. Comp. Chem. Eng., 35(12), 2960-2972.
7. Castro, P.M., Zeballos, L.J., Méndez, C.A., 2012. AIChE J., 58 (3), 789-800.
8. Zeballos, L. J., Castro, P.M., Méndez, C.A., 2011. Ind. Eng. Chem. Res., 50, 1705-1715.
9. Novas, J. M., Henning, G. P., 2012. Comp. Chem. Eng. 42, 189-205.
10. Castro, P.M., Aguirre, A.M., Zeballos, L.J., Méndez, C.A., 2011. Ind. Eng. Chem. Res., 50, 10665-10680.
11. Aguirre, A. M., Méndez, C.A., Gutierrez, G., De Prada, C., 2012. Comp. Chem. Eng. 47, 217-226.
12. Paul, H. J., Bierwirth, C., Kopfer, H., 2007. Int. J. Production Economics 105, 54-69.
13. Aguirre, A. M., Méndez, C. A., Castro, P. M., De Prada, C., 2012. Comp. Aided Chem. Eng., 30, 477-481.
14. Méndez C.A. Cerdá, J., 2003. Optimization and Engineering, 4, 7-22.
15. Aguirre, A.M., Méndez, C., De Prada, C., 2012. Comp. Aided Chem. Eng., 31, 1085-1089.
16. Nawaz, M., Enscore, E.E., Jr. and Ham, I., 1983. Omega, 11, 65–74.