We consider the scheduling problem of cyclic job shop with assembly/disassembly tasks. Such a problem is made of several jobs, each job consisting of tasks (assembly/disassembly tasks and transformation tasks) being assigned to machines in a cyclic way. Many methods have been proposed to solve the cyclic scheduling problem. Among them, we focus on the mathematical programming approach. We will propose here an improvement of an existing mathematical model for cyclic scheduling with assembly tasks and Work-In-Process minimization.

1. INTRODUCTION

Assembly lines are a manufacturing process in which parts are added to a product. In these systems we find tasks, which consist of two (or more) resource with different part types that need to be assembled. That is, an assembly operation can only be performed whenever all upstream machines have delivered parts. Production systems with assembly tasks are so frequent on automotive manufacturing, electronic parts manufacturing,... Disassembly lines, are also classified as non-linear production systems. In the contrast of the assembly line, tasks consist on removing components, and this is so used in de-manufacturing system. De-manufacturing is the process of disassembling products and then reusing, recycling or refurbishing them.

In this paper, we consider production systems that include both assembly and disassembly tasks and transformations tasks. The assembly tasks show the synchronization between tasks, and the disassembly promotes parallelism. The problem is to find a periodic schedule which minimizes a given objective function. The most important application fields of these systems are maintenance and recycling.

This problem has to be optimized with regard to several criteria like throughput and Work-in-Process (WIP - the number of parts in the system). The WIP, which is an economic criterion, represents the intermediate stock. In the cyclic context, the throughput criteria will be replaced by minimizing the time difference between two successive occurrences of one task for a given set of constraints. This time difference is called Cycle Time (CT).

We are interested here in the cyclic scheduling problem with assembly/disassembly tasks and Work-In-Process minimization.

The remainder of this paper is organized as follows. In section 2, cyclic systems with assembly/disassembly tasks will be introduced. Besides, the WIP in these systems will be defined. In section 3, we will describe a mathematical model which presents the base of our research. In the next section (section 4) we will use an illustrative example to show the contribution of new variables and highlight the modifications made on the model. The last part will be reserved for conclusions and topics for future research.

2. CYCLIC SYSTEMS WITH ASSEMBLY/DISASSEMBLY TASKS

The production lines of manufacturing system often include assembly/disassembly tasks. This is led by the nature of the aimed output, which requires to assemble and/or disassemble several
parts. Many authors are interested by this field of production system (Mascle (2001), Sundaram (2001), Jianzhong (2002), Trouillet (2002), Trouillet (2006), Trouillet (2007), Ben Amar (2009)). More precisely, Mascle, Sundaram and Jianzhong aim to find or optimize the assembly/disassembly sequences. Their works deals with the conception of the product. However, Trouillet and Ben Amar, consider that assembly/disassembly production system are already fixed and the main objective is to search for a scheduling of tasks under various constraints (fixed cycle time, minimizing Work-In-Process,...). There are also many systems with only disassembly tasks, for example, disassembly lines used in recycling (Field (1991), Gupta (1996), Lambert (2003)). We can also find a system with only assembly tasks (like Mane (2002)). In this context, the system must include the suitable resources able to perform these tasks.

Generally, production lines of manufacturing system can include three types of tasks ("Figure.1."). We use Petri nets notation to represent the precedence constraints:

**Transformation tasks:** affect the state of the pieces without adding extra parts in the system.

**Assembly tasks:** consist of assembling at least two parts to produce a new one. In this case, the number of parts in the system will decrease.

**Disassembly tasks:** consist of disassembling one piece to produce, at least, two parts. In this case, the number of parts in the system will increase.

![Figure 1: Tasks types](image)

There are many production systems with both assembly and disassembly task (Sundaram (2001), Trouillet (2002)).

Within assembly/disassembly systems, we have to deal with non-linear job, which means that a job can include many branches and the precedence constraints of operations can be multiple.

We suppose that the planning step using workflow analysis and the performance evaluation using Petri nets (Camus (1996), Korbaa (2002)) has fixed the production and the optimal cycle time. The cycle time is fixed by the bottleneck(s) machine(s) (resource with the maximum workload). It has been proven that, it is always possible to reach a cyclic schedule with a cycle time fixed by the critical machine. Hence, we have to schedule a production plan, i.e. to determine the sequences of operations on resources and the starting time of each task. The schedule must be within the Cycle Time, and every cycle, we produce one piece of each category.

By minimizing the WIP of a system we look for cost optimization. In fact, in factories, WIP levels between machines have a capacity limit. This is mainly due to the limited physical space available to store the parts temporarily and the limits of the transport system. Also, if the number of WIP increases significantly, it can produce a deadlock by overloading the system.

To deal with optimizing WIP in assembly/disassembly systems, it may be useful to analyze this issue in linear job systems. Indeed, the WIP in system without assembly/disassembly tasks represents the number of products, since every part in the system belongs to a single product and every product is composed by only one part. Moreover, many studies (Bourdeaud'huy (2006), Driss (2007), Korbaa (2002)...) consider that parts remain clamped to their transport resource (for example, pallet) during their entire journey in the system. Hence, minimizing WIP or the number of pallets is the same. However, in our case, with non-linear production lines, the number of parts in systems does not match the number of products. In fact, if we consider that we have to produce a "chair," we have to assemble 6 parts: 4 legs, the back and the seating. This means that, after assembly, the number of parts changes from 6 to 1. So, the previous definition of WIP does not fit with this new environment.

In the literature, there are two main definitions for the WIP. Fournier (2002) supposes that all parts generated after a disassembly task or disappeared after assembly (i.e. parts which belong to the same product), represent only one WIP. Hence, if we consider a cyclic production system, and we suppose that there are several parts in the cyclic window which belong to \( n \) products, then, there is \( n \) WIP in the system. However, this definition does not consider the number of pallets in the system. In fact,
we can find two schedules that present two WIPs, for example, the first schedule needs 10 pallets while the second requires 15 pallets. With this definition, we cannot choose the first schedule (which needs less pallets) compared to the second one.

Another point of view concerning the definition of the WIP is proposed by Trouillet (2002). He considers that the WIP in a system is equal to the maximum number of parts present in a cyclic window.

In this paper, we choose the second point of view, since the first one cannot ensure optimizing the number of parts in the system, which can imply a possible system saturation.

3. MATHEMATICAL MODEL FOR CYCLIC SCHEDULING PROBLEM WITH ASSEMBLY TASKS

In this section, we will present the mathematical model developed previously by Ben Amar (2009). This section seems important to highlight the contribution of the new variables that will be explained in the next section.

We will use an illustrative example reserved to this section ("Figure.3") to explain decision variables used in the mathematical model.

![Illustrative example for "part 3"](image)

We use the following notations to define a job shop $F$:

- The set $M = \{m_1, m_2, ..., m_{|M|}\}$ defines the set of machines of $F$.
- $G = \{g_1, g_2, ..., g_{|G|}\}$ defines a job shop as a set of jobs. $G$ represents the cardinal number of the set $G$.
- $O^G$: the set of all operations of the problem.
- $o_{i,j}$: denote operations, while $i$ stands for the job and $j$ the index of the operation in the corresponding job.
- $d_{i,j}$: denote processing time of operation $o_{i,j}$.
- $s(i, j) = (I, J)$ where $o_{I,J}$ represent(s) the successor(s) of $o_{i,j}$. For example, the successor of the operation “a” in “Figure.3” are “b” “c” and “d”. Note that in this illustrative example, tasks “a” and “b” are performed on the machine $M_1$. Operations “c”, “d” and “e” are performed respectively on $M_2$, $M_3$, $M_4$.
- $A_i^G$ defines the set of assembly tasks in the job $i$. $A_i$ represents the cardinal number of this set.
- $D_i^G$ defines the set of disassembly tasks in the job $i$. $D_i$ represents the cardinal number of this set. For the illustrative example cited above, $A_1^G = \{e\}$ and $D_1^G = \{a\}$.
- $C_{max} = \max_{m \in M} \left( \sum_{(i,j) \in D_m^G} (d_{i,j}) \right)$ represents the Cycle Time (CT). It consists on the maximum workload on the bottleneck machine(s).
- $t_{ij}$ corresponds to the activation time of the operation $o_{ij}$ within the considered cycle.
- $\delta_{k,l}$ is a binary variable corresponding to the order between operations performed on the same machine, such that $\delta_{k,l} = 1$ if $t_{ij} < t_{k,l}$ and 0 otherwise. In "Figure.4", we can see the case of $\delta_{k,l} = 1$. If the task $b$ starts before the completion of operation $a$, then we will have $\delta_{k,l} = 0$.

![A possible schedule of tasks](image)
\[
\begin{align*}
\text{Minimize: } & \sum_{i,j \in \mathcal{D}_i} (N_{i,j} - 1) + \sum_{i,j \in \mathcal{D}_i} (\alpha_{i,j}^{s(i,j)} + \beta_{i,j}^{s(i,j)}) \text{ s.t.:} \\
& \forall i \in [1, G], \forall j \in [1, K_i], \\
& t_{i,j} - t_{s(i,j)} - B - \alpha_{i,j}^{s(i,j)} \geq 1 - B - d_{ij} \\
& t_{i,j} - t_{s(i,j)} - B - \alpha_{i,j}^{s(i,j)} \leq -d_{ij} \\
& t_{ij} - t_{s(i,j)} - B - \beta_{ij}^{s(i,j)} \geq C_{\text{max}}^* + 1 - B - d_{ij} \\
& t_{ij} - t_{s(i,j)} - B - \beta_{ij}^{s(i,j)} \leq C_{\text{max}}^* - d_{ij} \\
& \forall m \in M, \forall o_{ij}, o_{kl} \in \mathcal{O}_m^0 \text{ s.t. } \left(\frac{1}{2}\right) \leq \left(\frac{1}{2}\right), \\
& t_{ij} - t_{kl} + C_{\text{max}}^* \cdot \delta_{ij}^{kl} \leq -d_{ij} + C_{\text{max}}^* \\
& t_{kl} - t_{ij} - C_{\text{max}}^* \cdot \delta_{ij}^{kl} \leq -d_{kl} \\
& B \geq 2 \cdot C_{\text{max}}^* - 1 \\
\end{align*}
\]

**Figure 2: Mixed Integer Programming Model for the Cyclic Job Shop with Assembly Tasks Ben Amar (2009)**

not respected in a schedule, this means that the system will need extra WIP to process these operations. In this case, the decision variable \( \alpha \) related to these two operations is fixed to 1. It means that \( \alpha_{ij}^0 = 1 \) if \( o_b \) is executed before the completion time of \( o_a \), where \( o_b \) stands for a successor of operation \( o_a \) in the job. For the same operations, \( \beta_{ij}^0 = 1 \) if \( o_a \) overlaps two cycles and completes after the activation time of \( o_b \) in the next cycle. Formally:

- \( \alpha_{ij}^{k,l} = 1 \) if \( o_{ij} \) is executed before the completion time of \( o_{ij} \), where \( o_{ij} \) stands for a successor of operation \( o_{ij} \) in the job;
- \( \beta_{ij}^{k,l} = 1 \) if \( o_{ij} \) overlaps two cycles and completes after the activation time of \( o_{ij} \) on the next cycle;

In “Figure 4” we show, also, possible cases \( \alpha \) and \( \beta \) variables. More explanations for \( \alpha \) and \( \beta \) can be found in Ben Amar (2007), since these two variables keep the same meaning for systems with or without assembly/disassembly tasks.

- \( B \in \mathbb{N} \) is a constant used to constrain the discrimination variables \( \alpha_{ij} \) and \( \beta_{ij} \) in a linear way. It has to be “big enough” (lower bound: \( 2 \cdot C_{\text{max}}^* - 1 \)) in order to make the inequalities (2) to (5) valid. This lower bound was computed as follow:

In order to consider (2) as a valid inequality, we must have: \( t_{ij} - t_{s(i,j)} - B - \alpha_{ij} \leq -d_{ij} \), and if we consider \( \alpha_{ij} = 1 \) then: \( t_{ij} - t_{s(i,j)} + d_{ij} \leq B \). In addition, we know that \( t_{ij} \leq C_{\text{max}}^* - 1 \) and \( d_{ij} \leq C_{\text{max}}^* \) then \( B \) must respect the following inequality:

The objective function (1), corresponds to the minimization of the WIP of the considered scheduling. It consists of two parts: a constant and decision variables \( \alpha, \beta \). Note that, if we consider only the decision variables in the objective function (1), the mathematical model will compute, only, the WIP needed for one path and the extra WIP required by the other branches. Hence, to know the WIP of the schedule, we have to add the WIP needed to perform the rest of the branches \( \sum_{o_{ij} \in \mathcal{D}_i} (N_{i,j} - 1) \). Indeed, in “Figure 5”, we notice that, after a disassembly task, the system generates \( n - 1 \) new WIPs that stands for the number of branches after the disassembly task. For example, before firing transition \( t_1 \), the system presents one WIP, then, after firing \( t_1 \), the system presents 3 WIP, which means that we need \( 2 = (3 - 1) \) extra WIP to process the rest of branches in the next stage. Then, if \( n \) represents the number of branches after a disassembly task, there is a need of \( (n - 1) \) new pallets in the system to handle the new WIPs generated after the disassembly. Formally, in the objective function, we denote by \( N_{i,j} \) the number of branches generated after the disassembly task \( o_{ij} \). The number of pallets to add to system after all disassembly tasks is equal to \( \sum_{o_{ij} \in \mathcal{D}_i} (N_{i,j} - 1) \).

Let’s recall that “Figure 2” presents the first mixed integer programming model corresponding to the cyclic scheduling problem with assembly tasks. This model will be enhanced by the add of two new variables and four constraints associated to them. The need to these modifications arises after modeling and resolving several examples. We have...
released that there is sometimes a difference between the WIP found by the mathematical model and the WIP found using the Gantt diagram. That’s why we have work to fix this issue by using new variables.

We had to mention also that this problem with this first version of the mathematical model does not affect the scheduling found. It is about the method used to compute the WIP generated by the scheduling. That’s why, either we use the first mathematical model or the modified version, we will find the same scheduling of task. The difference relies only on the number of WIP needed for the production.

In the next section, we will highlight the importance of the new variables added to the mathematical model.

4. NEW VARIABLES IN THE MATHEMATICAL MODEL

We have shown, previously, that the mathematical model was based mainly on two types of decision variables: \( \alpha \) and \( \beta \). These variables are used to compute the WIP in the system. In order to minimize the WIP we have to optimize an objective function, which consists of two parts: a constant that represents the extra WIP required by the branches between assembly/disassembly tasks and the sum of decision variables \( \alpha \) and \( \beta \). We will focus here on this sum, and how to optimize it.

In this section, we will show that we must not compute all \( \alpha \) and \( \beta \) variables in the objective function. We will use an illustrative example “Figure.6” to highlight the limits of the formulation proposed in the last section. Then we will explain how to fix it by introducing new variables and constraints.

The illustrative example consists on transformations tasks, three disassembly tasks and three assembly ones. This production system has to be repeated in order to satisfy the production demand. We use the following notation to distinguish tasks:

\[
q(3) m2
\]

- \( q \): index of the task.
- \( m2 \): processing time.
- \( m2 \): index of the used machine.

This illustrative example was formulated by the mathematical model presented previously. The resolution with CPLEX solver led to a possible scheduling presented by a Gantt diagram in “Figure.7”. We present also, in the same figure the WIPs needed for this schedule. The rule is that a new WIP is added when the precedence constraints between operations, fixed by the production system (in our case presented in “Figure.6”), cannot be satisfied. This is modeled in the mathematical model by using \( \alpha \) and \( \beta \) variables. For example, when comparing starting times of tasks \( a \) and \( b \) we have \( \alpha_{ab} = 1 \). Note that if we want to follow the notation used by the mathematical model, we must denote, for example, \( \alpha_{i,a} \) as \( \alpha_{i,a} \), but we have ignored the index “1” in order to simplify the notation, since there is only one job.

In our case, the bottleneck machines are \( M2 \) and \( M4 \) with a workload equal to 15 u.t.

The WIP needed for the whole scheduling is equal to 10 (as showed by “Figure.7”). However, when resolving this example with CPLEX solver, and using the mathematical model, the WIP found is equal to 11. In fact, on one hand, by comparing the execution time of tasks, we notice that \( \alpha_{ab} = \alpha_{b} = \alpha_{i} = \alpha_{p} = \alpha_{a} = 1 \), and the other decision variables are equal to zero. Then the WIP added by the decision variables is equal to 6. On the other hand, as cited in the objective function (1), we have to add extra WIP to handle the new pieces added in the system after disassembly tasks. First we have to identify the disassembly tasks set:

\[
D_i = \{b, c, k\}
\]

In our case, we have to add:

\[
\sum_{o_{i,j} \in D_i^i} (N_{i,j} - 1) = (N_b - 1) + (N_c - 1) + (N_k - 1) = 5.
\]

\( N_{i,j} \) represents the number of branches after the disassembly task \( o_{i,j} \).

As a consequence, the total WIP found by the mathematical model is equal to 5 + 6 = 11. However, informally, this cannot be the real need of WIP for our production system, since by dressing the Gantt diagram, we found 10 WIP. In fact, for an assembly task, all parts that have to be assembled must be ready and available. For example, in our case we have three assembly tasks, \( A_i^i = \{i, p, u\} \). Task \( i \) can
Figure 6: Illustrative example

Figure 7: Scheduling of tasks
only be performed if tasks \(c, f\) and \(h\) have delivered their parts. This means that an extra WIP will be added, only, if the task \(i\) began before the end of the last task among \(c, f\) and \(h\) (which is in our case the task \(h\)).

As a conclusion, the precedence constraints for an assembly task, can produce at most one WIP.

In our case, this means that, we must have:

\[
\begin{align*}
\alpha^e_i + \alpha^f_i + \alpha^h_i & \leq 1 \\
\alpha^m_p + \alpha^p_p & \leq 1 \\
\alpha^i_u + \alpha^p_u + \alpha^t_u & \leq 1
\end{align*}
\]

However, with the actual mathematical model, this property is not guaranteed. For example, in our case, we have \(\alpha^m_p + \alpha^p_p = 2\). Thus, we have to improve the formulation of the objective function in order to take in consideration this special property of assembly tasks.

To solve this modeling issue, we have used, for each assembly task, a new variable \(\lambda\). This former is equal to 0 if the sum of the associated decision variables \(\alpha\) is equal to 0, otherwise \(\lambda = 1\). More precisely, if we suppose that \(o_{i,j}\) is an assembly task:

\[
\begin{align*}
\text{Let } s(pi, pj) = (i, j), & \text{ if } \sum (\alpha s_{i,j}) \geq 1 \\
\text{then } \lambda_{i,j} = 1 & \text{ else } \lambda_{i,j} = 0 \quad (*)
\end{align*}
\]

Formally, property (*) is formulated in the mathematical model by adding two inequalities:

\[
\begin{align*}
\forall o_{i,j} \in \mathcal{A}_p^G: s(pi, pj) = (i, j) & \quad \sum (\alpha s_{i,j}) \leq B \cdot \lambda_{i,j} \\
\sum (\alpha s_{i,j}) - B \cdot \lambda_{i,j} \geq 1 - B
\end{align*}
\]

These inequalities are derived from the following relation:

\[
1 + (\lambda_{i,j} - 1) \cdot B \leq \sum (\alpha s_{i,j}) \leq B \cdot \lambda_{i,j}
\]

Similarly, this relation is a consequence of the following property:

\[
\begin{align*}
\forall \lambda \in \{0, 1\}, \quad X \in \mathbb{Z} \quad \text{and} \quad B \in \mathbb{N} & \quad \text{with } B \geq X; \quad \text{if } X \text{ satisfies } 1 + (\lambda - 1) \cdot B \leq X \leq \lambda \cdot B \text{ then this means:} \\
\left\{ \begin{array}{l}
X > 0 \iff \lambda = 1 \\
X \leq 0 \iff \lambda = 0
\end{array} \right. \quad (10)
\end{align*}
\]

If we replace \(X\) in (10) by \(\sum (\alpha s_{i,j})\) and \(\lambda\) by \(\lambda_{i,j}\) we find inequalities (8) and (9).

Finally, for each assembly task, the new variables \(\lambda\) will replace the associated decision variables \(\alpha\) in the objective function. That’s why the objective function must be changed by considering only the decision variables \(\alpha\) associated to operations in the set:

\[
S = \mathcal{O}^G \setminus \{o_{i,j} \text{ s.t.. } s(i, j) = (pi, pj) \text{ with } o_{pi, pj} \in \mathcal{A}^G\}
\]

The objective function will be:

\[
\begin{align*}
\sum (N_{i,j} - 1) + \sum (\alpha s_{i,j}) & + \sum (\lambda + \gamma_{i,j}) \\
\sum (\beta_{i,j}) & \leq B \cdot \gamma_{i,j} \quad (11) \\
\sum (\beta_{i,j}) - B \cdot \gamma_{i,j} & \geq 1 - B \quad (12)
\end{align*}
\]

In order to illustrate the new variables added, we will recalculates the WIP associated to scheduling cited in “Figure.7”, but using the new formulation.

The new objective function consists on three parts, the first one \(\left( \sum (N_{i,j} - 1) \right)\) is the same part used in the previous version. That’s why, it will keep its value (equal to 5) already calculated and explained.

The second part \(\sum (\alpha s_{i,j})\) will compute the WIP added by decision variables \(\alpha\) and \(\beta\) not related with assembly tasks. The WIP added by this part is equal to \(3\) since \(\alpha^e = \alpha^f = \alpha^h = 1\).

The last part in the objective function is:

\[
\sum (\lambda + \gamma_{i,j}) = \lambda_i + \lambda_p + \lambda_u + \gamma_i + \gamma_p + \gamma_u
\]

(Wrap text)
improved model, the number of WIP found is either less than or equal to that found by the former.

Note that this mathematical model has been implemented, and we have solved the mathematical models of illustrative example by using the CPLEX solver. Experiments were carried out using a 2.0 Ghz Core 2 duo with 4 Go of RAM.

The resolution with the old mathematical model has taken 0.70 second, and led to 11 WIP. Whereas, the resolution of the new version of the mathematical model has taken 0.72 second, and led to 10 WIP.

5. CONCLUSION

In this paper, we have focused, firstly, on exposing the cyclic scheduling problem with Work-In-Process minimization. Then, we have presented a mixed integer programming model for the cyclic job shop with assembly/disassembly tasks and WIP minimization. Besides, we have tried to highlight the modifications made in this version in order to improve the way of computing the WIP. The improvements consist on using two new variables in order to formulate a specific property of assembly tasks. In addition, we have explained the advantages of this improvement by using an illustrative example and comparing the WIP found by the two versions. In terms of perspective, we can consider other production constraints like systems with limited buffer. In this case, the WIP will be fixed, and we will look for optimizing the Cycle time.

6. REFERENCES


Formal Approach for FMS Cyclic Scheduling, “IEEE
SMC Transactions”, Part C, Vol. 37, Issue 1, pp. 126-
137.