A Mathematical Model for Cyclic Scheduling With Assembly Tasks and Work-In-Process Minimization

Ben Amar M.A, Hubert Camus, O. Korbaa

A Mathematical Model for Cyclic Scheduling With Assembly Tasks and Work-In-Process Minimization

Mohamed Amin Ben Amar, Hervé Camus, Ouajdi Korbaa

***Abstract*—In this paper, we deal with the cyclic scheduling problem. More precisely, we consider the *cyclic job shop* with assembly 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. This kind of scheduling problem is well fitted to medium and large production demands, since the cyclic behavior can avoid the scheduling of the whole tasks by considering only a small temporal window (cycle). Thus, cyclic scheduling is a heuristic to solve the scheduling problems whose complexity is NP-hard in the general case. Many methods have been proposed to solve the cyclic scheduling problem. Among them, we focus on the mathematical programming approach. We will propose here a mathematical model for cyclic scheduling with assembly tasks and Work-In-Process minimization, and we illustrate this approach with an example from literature.**

I. Introduction

CYCLIC scheduling problems take place in different application areas such as compiler design, automated manufacturing systems, digital signal processing, railway scheduling, timetabling, etc. We will focus here on the automated manufacturing systems. In this domain, the production consists of cyclic jobs assigned to machines. Each job consists of assembly/disassembly tasks and transformations tasks. The assembly tasks show the synchronization between operations, and the disassembly promote parallelism.

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 Cycle Time (CT).

These two criteria are antagonistic. On one hand, to maximize the throughput, we have to use enough parts (WIP) to feed the bottleneck machine. On the other hand, with a few number of WIP (one for example) the resources will be pending for parts (in particle, the bottleneck machine(s)) and the throughput will not be optimized. Hence to take into account these two criteria, we will follow the resolution developed by Camus [4]. It consists of two phased approach. The planning step, which determine the optimal cycle time, and the scheduling step, which consists on minimizing the WIP while respecting the optimal cycle time (as a hard constraint). This approach ensures the existence of a scheduling, since a sufficient number of WIP allows to saturate the bottleneck[1] machine(s).

We focus here on the scheduling of operations in a precalculated cycle time, and we do not look for the best production ratios to be produced during a cycle time (an issue largely studied by Chrétienne [5] and Hanen [11]). In fact, we suppose that we know exactly what to produce during a cycle, which allows us to determine the optimal cycle time based on the workload of the critical resource.

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

The remainder of this paper is organized as follows. In section 2, we will introduce systems with assembly/disassembly tasks, and we will propose a cyclic approach to solve these problems. In addition, we will define the WIP in these systems. In section 3, we will describe the mathematical model. In section 4, we will use an illustrative example in the literature to explain our approach. To conclude, we propose several perspectives to extend our work.

II. Cyclic Scheduling Problem with Assembly/Disassembly Tasks

A. Systems with Assembly/Disassembly tasks

The production lines of manufacturing system often include assembly/disassembly tasks. This can be accounted for the nature of the aimed output, which requires to assemble and/or disassemble several parts ([13], [17], [18], [19], [20], [21]). There are also many systems with only disassembly tasks, for example disassembly lines used in recycling ([8], [10], [15]). We can also find a system with only assembly tasks (like [16]).

In this context, the system must include the suitable resources able to perform these tasks. In these systems ("Fig.1," [20]) there are three categories of operations:
Transformation tasks: affects the state of the pieces without adding extra parts in the system.

Manuscript received June 15, 2009.

M. A. Ben Amar is with the LI3 laboratory, ISG, University of Tunis, Tunisia, (e-mail: aminbenamar@yahoo.fr).

H. Camus is with the LAGIS laboratory, EC Lille, Villeneuve d'Ascq, France, (e-mail: herve.camus@ec-lille.fr).

O. Korbaa is with the LI3 laboratory (ISG, Tunis), ISITCom Hammam Sousse, Tunisia, (e-mail: Ouajdi_Korbaa@yahoo.fr).

[1] Machine with the *maximum workload*, i.e. machine $m \in \mathbb{M}$ for which the quantity $\sum_{(m,d_{ij})\in\mathbb{O}^{\mathbb{G}}} (d_{ij})$ is the greatest.

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

It follows that the precedence constraints of operations can be multiple. Which means that, with assembly, one task can have two or more predecessors. However, with disassembly one task can have two or more successors. Hence, we have to deal with non-linear job, which means that a job can include many branches.

We will consider here systems that contain only one disassembly operation and one assembly operation. The disassembly operation comes before the assembly one. Between these two tasks we will find at least two branches (on *Stage_2* – "Fig.1"). Thus, the disassembly and the assembly tasks will delimit the different stages of the system ("Fig.1"). This choice can be viewed as a primary model type that takes into account the most important constraint which is encountered in assembly/disassembly systems: the synchronization of tasks. This choice can be justified by the need of simplification to start studying this scheduling problem. However, in future works, we will consider other types of models within extended constraints (like systems with several assembly and disassembly tasks and/or with imbricated stages).

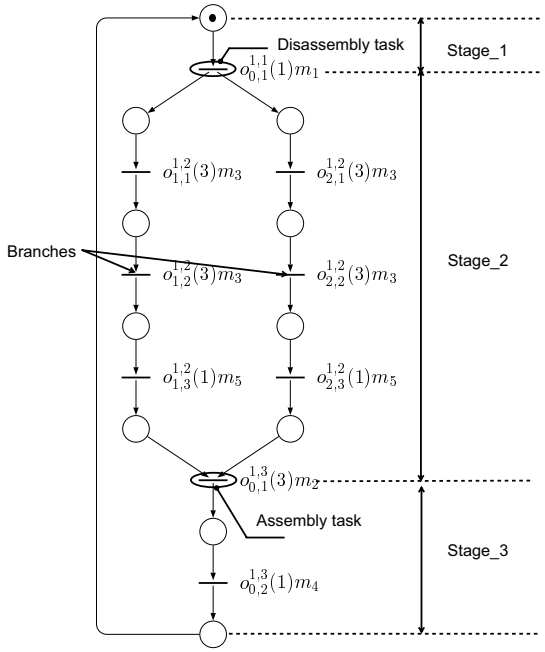

Fig 1. Assembly/Disassembly problem.

B. Cyclic behavior

We propose to schedule a production plan, i.e. to determine the sequences of operations on resources and the time of lunching of each task. The schedule must be within the Cycle Time, and every cycle we produce one piece.

The Scheduling problems are well known to be highly combinatorial. It has been shown that project planning problems are of polynomial complexity and that cyclic scheduling problems are NP-complete. Taking into account transformation tasks and assembly/disassembly one, makes the first problem NP-hard in most cases and keeps the second one in the NP-complete class. Hence, the use of heuristics is generally recommended. The scheduling of cyclic production system can be a possible solution to global scheduling. The answer to the total demand will be given by the repetition of a sequence known as cyclic scheduling. However, the optimal scheduling of a cycle does not guarantee the optimality of the total production, since the "*the sum of optimal sub-paths is not necessarily an optimal path"* [1]. That's why the cyclic behavior is still a heuristic. We can evaluate the performance of the cyclic scheduling by comparing the total time production with a lower bound computed from workflow analysis and the workload of the bottleneck machine.

In this work, we suppose that the production and the optimal cycle time have been fixed in the planning step from workflow analysis and performance evaluation using Petri nets (Camus [4], Korbaa [14]).

C. Work-In-Process

The aim of minimizing the WIP of a system is mainly due to the minimization of costs (intermediate stock, pallets design, and manufacturing). In factories, WIP levels between machines have capacity limits. 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, it can produce a deadlock by overloading the system.

To understand the WIP in systems with assembly/disassembly tasks, we suggest to present this concept in linear jobs systems with (system within which each task has only one predecessor and one successor operation i.e. without assembly/disassembly tasks). In linear Job systems, the WIP represents the number of products in a system. Since there are no assembly/disassembly tasks, then every part in the system is bound to a single product. Moreover, many studies [3], [7], [12], [14], [6] ...) 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, the number of parts in systems with non-linear production line 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. Hence, the previous definition of WIP has to be reviewed.

In this context, Fournier [21], has proposed a definition for the WIP. He 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 x products, then, there is x 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 can not 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 [19]. 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 will consider the definition of Trouillet, which aims to minimizing the maximum number of parts in the system. Moreover, we will consider that parts remain clamped to their transport resource (for example pallet) during their entire journey in the system.

III. A Mathematical Model for Cyclic Scheduling Problem with Assembly Tasks

A. Job Shop

We use the following notations to define a job shop F.

Machines: The set $\mathbb{M} = \{m_1, m_2, ..., m_{|\mathbb{M}|}\}$ defines the set of **machines** of F. These resources are renewable and not shared by any operations. This means that they are reusable once they have finished the execution of a task and can only process one task at a time.

Operation: We define an **operation** of F using the **machine** $m \in \mathbb{M}$ as a pair $(m,d) \in \mathbb{M} \times \mathbb{N}^*$ where d is called the **duration** of the corresponding operation. We denote by $\mathbb{O}^\infty \equiv \mathbb{M} \times \mathbb{N}^*$ the set of possible (machine, duration) pairs.

Job: We define a **job** g of the job shop F as a **sequence of operations**. Among these operations, we can find Assembly/Disassembly tasks. We denote by:

$\mathbb{O}^{\mathbb{G}}$: set of all operations of the problem.

E_i: Number of stages of the Job i.

$b_{i,j}$: Number of branches in stage j of the job i.

$E_k^{i,j}$: Number of operations in the branch k of the stage j of the job i.

We denote operations by $o_{k,l}^{i,j}$, while i, j, k and l stands respectively for: *the job, the stage, the branch* and *the index of the operation for the corresponding branch*.

$s(i,j,k,l) = (I,J,K,L)$ where $o_{K,L}^{I,J}$ represent(s) the successor(s) of $o_{k,l}^{i,j}$.

Job Shop: We define a **job shop** as a set of **jobs** $\mathbb{G} = \{g_1, g_2, ..., g_{|\mathbb{G}|}\}$. We denote by G the cardinal number of $\mathbb{G}$, and we order the jobs of the *job Shop* by the formal parameter $i \in [\![1, \mathsf{G}]\!]$.

B. Cyclic Scheduling Problem

The goal of cyclic scheduling is to schedule the *cyclic pattern*, within which each operation of the job shop is scheduled in a time range called *cycle time*. The optimal cycle time is defined as the sum of durations of tasks associated with the bottleneck machine(s).

Since this optimal cycle time is reachable (we have to use enough parts: a WIP for each tasks in the system), classical scheduling problems consist in minimizing the number of pieces in the system for a given cycle time, equal to $C^*_{max} = \max_{m \in \mathbb{M}} \left(\sum_{(m,d_{ij}) \in \mathbb{O}^{\mathbb{G}}} (d_{ij}) \right)$.

We will work here with a Cycle Time (CT) which is equal to C^*_{max}.

$$\mathrm{CT} = C^*_{max}$$

C. Mixed Integer Programming Model

We define below, "Fig.2," a mixed integer programming model corresponding to the cyclic scheduling problem defined in section III.B.

$$\text{Minimize} \sum_{o_{k,l}^{i,j} \in \mathbb{O}^{\mathbb{G}}} nb + (\alpha^{i,j,k,l}_{s(i,j,k,l)} + \beta^{i,j,k,l}_{s(i,j,k,l)}) \text{ such that:} \quad (1)$$

$$\bullet \forall i \in [\![1,\mathrm{G}]\!], \forall j \in [\![1,E_i]\!], \forall k \in [\![1,b_{i,j}]\!], \forall l \in [\![1,E_k^{i,j}]\!]$$

$$\begin{cases} t_{k,l}^{i,j} \in [\![0, C^*_{max} - 1]\!] & (2) \\ \alpha^{i,j,k,l}_{s(i,j,k,l)} \in \{0,1\} & (3) \\ \beta^{i,j,k,l}_{s(i,j,k,l)} \in \{0,1\} & (4) \\ t_{k,l}^{i,j} - t_{sK,sL}^{sI,sJ} - B.\alpha^{i,j,k,l}_{sI,sJ,sK,sL} \geq 1 - B - d_{k,l}^{i,j} & (5) \\ t_{k,l}^{i,j} - t_{sK,sL}^{sI,sJ} - B.\alpha^{i,j,k,l}_{sI,sJ,sK,sL} \leq -d_{k,l}^{i,j} & (6) \\ t_{k,l}^{i,j} - t_{sK,sL}^{sI,sJ} - B.\beta^{i,j,k,l}_{sI,sJ,sK,sL} \geq C^*_{max} + 1 - B - d_{k,l}^{i,j} & (7) \\ t_{k,l}^{i,j} - t_{sK,sL}^{sI,sJ} - B.\beta^{i,j,k,l}_{sI,sJ,sK,sL} \leq C^*_{max} - d_{k,l}^{i,j} & (8) \end{cases}$$

$$\bullet \forall m \in \mathbb{M}, \forall o_{k,l}^{i,j}, o_{K,L}^{I,J} \in \mathbb{O}_m^{\mathbb{G}} \text{ s.t. } (i,j,k,l) \lesssim (I,J,K,L),$$

$$\begin{cases} \delta^{i,j,k,l}_{I,J,K,L} \in \{0,1\} & (9) \\ t_{k,l}^{i,j} - t_{K,L}^{I,J} + C^*_{max}.\delta^{i,j,k,l}_{I,J,K,L} \leq -d_{k,l}^{i,j} + C^*_{max} & (10) \\ t_{K,L}^{I,J} - t_{k,l}^{i,j} - C^*_{max}.\delta^{i,j,k,l}_{I,J,K,L} \leq -d_{K,L}^{I,J} & (11) \end{cases}$$

$$\text{with } B \in \mathbb{N}, B \geq 2.C^*_{max} - 1$$

Fig 2. Mixed Integer Programming Model

- Variable $t_{k,l}^{i,j} \in [\![0, C^*_{max} - 1]\!]$ corresponds to the activation date of the operation $o_{k,l}^{i,j}$ within the considered cycle;
- Variable $\delta^{i,j,k,l}_{I,J,K,L} \in \{0,1\}$ is the binary variable corresponding to the order between operations performed on the same machine, such that $\delta^{i,j,k,l}_{I,J,K,L} = 1$ if $t_{k,l}^{i,j} < t_{K,L}^{I,J}$ and 0 otherwise. "Fig.3" presents a scheduling of the illustrative example used below ("Fig.6"). In this schedule, we have $\delta^{1,2,2,2}_{1,2,2,1} = 1$, since: $t_{2,1}^{1,2} < t_{2,2}^{1,2}$.

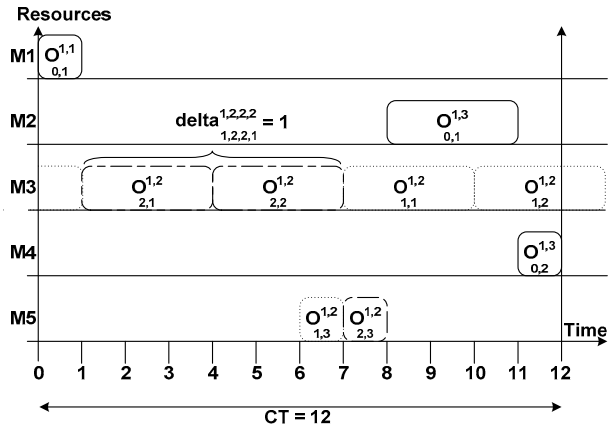

Fig 3. Variable δ (delta)

- Variables $\alpha_{sI,sJ,sK,sL}^{i,j,k,l}$ and $\beta_{sI,sJ,sK,sL}^{i,j,k,l}$ correspond to binary variables used to compute the WIP:
  - $\alpha_{sI,sJ,sK,sL}^{i,j,k,l} = 1$ if $o_{sK,sL}^{sI,sJ}$ is executed before the completion time of $o_{k,l}^{i,j}$, where $o_{sK,sL}^{sI,sJ}$ stands for a successor of operation $o_{k,l}^{i,j}$ in the job;
  - $\beta_{sI,sJ,sK,sL}^{i,j,k,l} = 1$ if $o_{sK,sL}^{sI,sJ}$ overlaps two cycles and completes after the activation time of $o_{k,l}^{i,j}$ on the next cycle;

"Fig.4" presents a scheduling of a cyclic linear job. This job consists of 3 tasks: $o_{0,1}^{1,1}$, $o_{0,2}^{1,1}$ and $o_{0,3}^{1,1}$

$o_{0,1}^{1,1}$ is followed by $o_{0,2}^{1,1}$.
$o_{0,2}^{1,1}$ is followed by $o_{0,3}^{1,1}$.
$o_{0,3}^{1,1}$ is followed by $o_{0,1}^{1,1}$.

We presents in "Fig.4" a scheduling to illustrate the case when we have $\alpha_{sI,sJ,sK,sL}^{i,j,k,l} = 1$ and $\beta_{sI,sJ,sK,sL}^{i,j,k,l} = 1$.

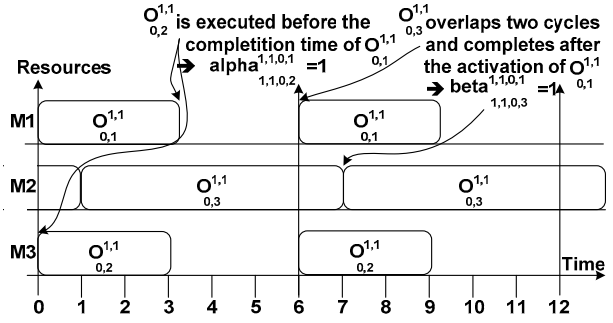

Fig 4. Binary variables used to compute the WIP (α and β).

More explanations for α and β can be found in [2], 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 $\alpha_{sI,sJ,sK,sL}^{i,j,k,l}$ and $\beta_{sI,sJ,sK,sL}^{i,j,k,l}$ in a linear way. It has to be "*big enough*" (lower bound: $2.C_{max}^{*} - 1$) in order to make the inequalities (5) to (8) valid. This lower bound was computed as follow:

In order to consider "(6)" as a valid inequality, we must have: $t_{k,l}^{i,j} - t_{sK,sL}^{sI,sJ} - B.\alpha_{sI,sJ,sK,sL}^{i,j,k,l} \leq -d_{k,l}^{i,j}$. If $\alpha_{sI,sJ,sK,sL}^{i,j,k,l} = 1$, then:

$$t_{k,l}^{i,j} - t_{sK,sL}^{sI,sJ} + d_{k,l}^{i,j} \leq B$$

In addition, we know that $t_{k,l}^{i,j} \leq C_{max}^{*} - 1$ and $d_{k,l}^{i,j} \leq C_{max}^{*}$, then B must respects the following inequality:

$$2.C_{max}^{*} - 1 \leq B$$

- Remaining inequalities (5) to (8), (10) and (11) constrain the previous variables according to their meanings.
- Finally, the objective function "(1)," corresponds to the minimization of the WIP of the considered scheduling. It consists on two parts: a constant plus decision variables (α and β). Note that, if we consider only 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. Indeed, in "Fig.5" we notice that, after a disassembly task, the system generates *(nb–1)* new WIP, *nb* stands for the number of branches in the second stage (*Stage_2*). 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 second stage.

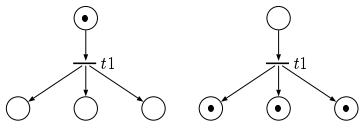

Fig 5. Variations of the Numbers of Tokens

In addition to this mathematical model, we define two other properties. Firstly, we consider that the first operation of the first job (operation $o_{0,1}^{1,1}$) starts at time 0 (which means that $t_{0,1}^{1,1} = 0$). Indeed, the steady state can be observed at different dates and always presents the same WIP. This is due to the cyclic behavior of the production. Hence, generality is maintained by considering that the cycle begins when the execution of operation $o_{0,1}^{1,1}$ starts. Secondly, the total WIP is made of the WIP needed to achieve each job separately. Let T_i be the total duration of the *ith* possible path from the first operation to the last one. If T_i is great to *CT*, then this sequence of operations is longer than a cycle and has to be cut into several cycles. This is done by introducing several parts of this sequence of operations: WIP. This number has to be at least equal to the integer superior or equal to T_i by *CT*.

$$WIP_{\min} = \sum_{i:\ Possible\ operations\ sequences} \left\lceil \frac{T_i}{CT} \right\rceil \qquad (12)$$

Concerning the optimality of the solutions, this model reflects exactly the constraints that can be found in a

scheduling problem, namely the constraints of precedence between operations and the constraints of resource sharing. Therefore, this model ensures the optimality of the solutions. On the other hand, we work with an exact approach and the resolution is done by a linear program solver (CPLEX).

Note that the mathematical model can deal, either, with problems with *linear jobs* [2] or with *Assembly/Disassembly* tasks.

IV. Illustrative Example

In this section we will use the example "Fig.1" of Disassembly/Assembly system, in order to illustrate the approach to compute optimal WIP with the mathematical model. The original example "Fig.6" was used by Trouillet in [20].

There are two main differences between "Fig.1" and "Fig.6":

- Transition t_7 in "Fig.6" will be considered as the first operation in "Fig.1." In fact, this is possible since the problem is cyclic.
- In "Fig.6," all the transitions are fired once except t_1 and t_2 which are fired twice. This property is replaced by the use of two successive operations for each transitions t_1 and t_2. Indeed, this choice is justified by the fact that we use the same resource M_3 for these two transitions. Hence, necessarily, transitions t_1 and t_2 will be performed on M_3 sequentially. Here, we look to work with ordinary Petri net for reasons of understanding and readability for our model.

The system contains 3 stages. The second stage contains 2 branches.

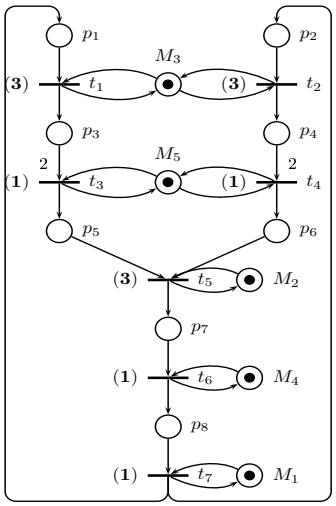

Fig 6. Illustrative example used by Trouillet in [20]

There are five resources denoted by M_1, M_2, M_3, M_4, M_5. The Cycle Time (*CT*) is equal to *12*, which is the workload of M_3 (bottleneck resource).

To compute a lower bound for the WIP, we know the optimal cycle time *CT* and the total duration of each possible path from the first task to the last one, through the different branches. For our example, we have two paths:

- $o_{0,1}^{1,1}, o_{1,1}^{1,2}, o_{1,2}^{1,2}, o_{1,3}^{1,2}, o_{0,1}^{1,3}, o_{0,3}^{1,3}$.
- $o_{0,1}^{1,1}, o_{2,1}^{1,2}, o_{2,2}^{1,2}, o_{2,3}^{1,2}, o_{0,1}^{1,3}, o_{0,3}^{1,3}$.

If we suppose that we will process only the first path, then we will need at least 12 t.u., which means, at least, one part, i.e. one WIP. Then, the second path needs at least one WIP, as well.

The WIP lower bound is equal to: $\left\lceil \frac{12}{12} \right\rceil + \left\lceil \frac{12}{12} \right\rceil = 2$.

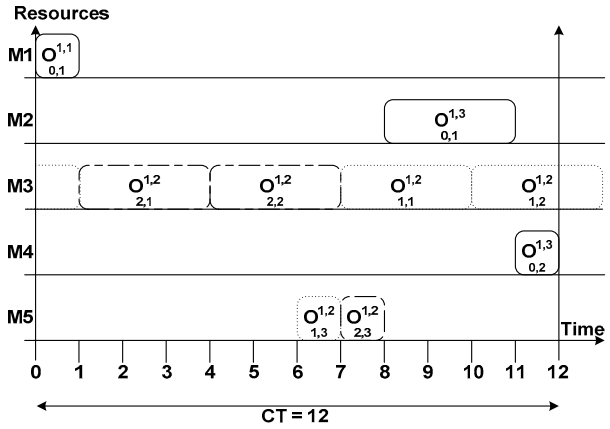

Fig 7. Scheduling on resources

"Fig.7" represents the computed schedule of tasks on the resources using linear program solver *CPLEX 9.0* on an *Intel Pentium 4* at *2.8 GHz* and *1Go RAM*, under *Windows XP*. The resolution takes about *1s*.

"Fig.8" represents the same schedule, but, here, we focus on the number of pallets used in the system.

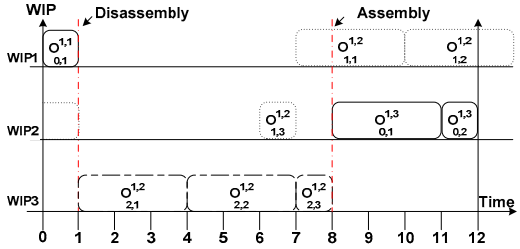

Fig 8. Scheduling from the part's point of view

"Fig.8" shows that the schedule requires 3 pallets. Hence the optimal number of WIP is equal to 3. This level of WIP was found by the mathematical model through variables α and β:

$$\alpha_{1,2,1,1}^{1,1,0,1} + \alpha_{1,2,2,1}^{1,1,0,1} + \alpha_{1,2,1,2}^{1,2,1,1} + \alpha_{1,2,1,3}^{1,2,1,2} + \alpha_{1,2,2,2}^{1,2,2,1} + \alpha_{1,2,2,3}^{1,2,2,2} + \alpha_{1,3,0,1}^{1,2,2,3} + \alpha_{1,3,0,1}^{1,2,1,3} + \alpha_{1,3,0,2}^{1,3,0,1} + \alpha_{1,1,0,1}^{1,3,0,2} + \beta_{1,2,1,1}^{1,1,0,1} + \beta_{1,2,2,1}^{1,1,0,1} + \beta_{1,2,1,2}^{1,2,1,1} + \beta_{1,2,1,3}^{1,2,1,2} + \beta_{1,2,2,2}^{1,2,2,1} + \beta_{1,2,2,3}^{1,2,2,2} + \beta_{1,3,0,1}^{1,2,2,3} + \beta_{1,3,0,1}^{1,2,1,3} + \beta_{1,3,0,2}^{1,3,0,1} + \beta_{1,1,0,1}^{1,3,0,2} = 2$$

All the variables here are null except: $\alpha_{1,2,1,3}^{1,2,1,2} = \alpha_{1,1,0,1}^{1,3,0,2} = 1$. We can verify these two values from "Fig.8," while $t_{1,2}^{1,2} + d_{1,2}^{1,2} > t_{1,3}^{1,2}$ and $t_{0,2}^{1,3} + d_{0,2}^{1,3} > t_{0,1}^{1,1}$.

The mathematical approach computes the WIP needed for one path and the extra WIP required by the other branches. Hence, to find out the WIP needed for the whole schedule, we must add *(n–1)* pallets (section 3.C) to the WIP level found by the resolution of the mathematical model.

In this case, the WIP computed using mathematical model is equal to 2 and we have two branches in the second stage. Hence, the WIP of the schedule is equal to $2 + (2 - 1) = 3$.

We notice here that the optimal WIP computed with our approach is equal to 3 and that the lower bound of the WIP is equal to 2. In fact, this theoretical value cannot be reached. Indeed, we mentioned that we have two possible paths from the first task to the last one. If we consider that we will perform each path separately, which means that we consider that machine M_3 will be available at any time. With this relaxation, we need 12 t.u. to perform each path apart, which means 2 WIP. However, "Fig.1" shows that M_3 is shared by the two paths. Hence, there will be, necessarily, an extra time while processing one of these two paths, which means that there will be a need for at least one more WIP. Then, the level of WIP found by our approach (3) is thus optimal.

V. CONCLUSION

This paper deals with cyclic scheduling problems with assembly/disassembly tasks and Work-In-Process minimization. The main contribution here is to propose a mathematical model of the scheduling issue of such systems.

First, we have presented systems with assembly/disassembly tasks and we have shown the interest of using cyclic scheduling approach to solve these problems. Secondly, we have clearly defined the concept of WIP in these systems. Afterwards, we have proposed a mathematical model which deals with the specificity of assembly systems, i.e. synchronization of multiple tasks. Then, we have used an illustrative example that has been previously used by Trouillet [20] to explain our model and our resolution method.

This study shows that one can solve optimally problems with assembly/disassembly tasks. We have shown that we can find an optimal scheduling (cycle time) for assembly/disassembly systems like Fournier in [9]. However, our approach allows, in addition, to find the optimal WIP in the system.

Future works will consider extended assembly/disassembly systems: with several tasks and imbricated stages. In addition, we can add extra constraints to the problems like working with a limited WIP. Moreover, we aim to substitute the CPLEX solver for an algorithm especially fitted to the mathematical model, in order to improve the resolution time.

REFERENCES

[1] Bellmann R., Dynamic Programming, *Princeton University Press,* Princeton, New Jersey, 1965.

[2] Ben Amar M.A., Bourdeaud'huy T. and Korbaa O., Cyclic Scheduling MIP Implementation: Cutting Tetchniques, *ICPR'07* 2007, Valparaiso, Chili.

[3] Bourdeaud'huy T. and Korbaa O., A Mathematical Model For Cyclic Scheduling With Work-In-Progress Minimization, INCOM'06 (2006), Saint Etienne, France.

[4] Camus H., Ohl H., Korbaa O. and Gentina J-C., Cyclic Schedules in Flexible Manufacturing Systems with Flexibilities in operating sequences, *Proceedings of the 17th International Conference on Application and Theory of Petri Nets (ICATPN)*, Osaka, Japan (1996), pp. 97-116.

[5] Chretienne P., Coffman E.G., Lenstra J.K., and Liu Z., *Scheduling: Theory and its applications* (1997), chapter 3, pages 33-64, John Wiley & Sons.

[6] Driss, O.B., Korbaa O.,Ghedira K. and Yim P., A distributed transient inter-production scheduling for flexible manufacturing systems , Journal Europeen des Systemes Automatises, JESA 2007, Volume 41, n°1.

[7] Dupas R., Cavory G. and Goncalves G., Optimising the throughput of a manufacturing production line using a genetic algorithm. *Real-World Applications GECCO 1999*: 1775.

[8] Field F.R. and Clark J.P., Recycling of USA automobile materials: a conundrum for advanced materials, *ATA 1991*, Vol. 44 (8/9), pp. 541-555.

[9] Fournier O., Lopez P. and Lan Sun Luk J.D., Cyclic scheduling following the social behavior of ant colonies. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics 2002*. Volume 3, Pages 450-454.

[10] Gupta S.M., and McLean C.R., Disassembly of products. *Computers and Industrial Engineering (1996)*. 31 1-2, pp. 225-228.

[11] Hanen C. and Munier Kordon A., Periodic schedules for linear precedence constraints. *Discrete Applied Mathematics* (2009), 157 (2), pp. 280-291.

[12] Hsu T., Korbaa O., Dupas R. and Goncalves G, Cyclic scheduling for F.M.S.: Modelling and evolutionary solving approach, *European Journal of Operational Research, EJOR* (2008), pp 463-483, Vol. 191, No. 2.

[13] Jianzhong Mo, Zhang Q. and Gadh R., Virtual Disassembly, *International Journal of CAD/CAM (2002)*, 2(1), 2002, 29-37.

[14] Korbaa O. Camus H. and Gentina J.-C., A New Cyclic Scheduling Algorithm for Flexible Manufacturing Systems, *International Journal of Flexible Manufacturing Systems (IJFMS)* 2002, Vol. 14, N. 2, pp. 173-187.

[15] Lambert A. J. D., Disassembly sequencing: a survey. *International Journal of Production Research*, Volume 41, Issue 16 November 2003 , pages 3721 - 3759.

[16] Mane A. , Nahavandi S. and Zhang Jingxin, Sequencing production on an assembly line using goal chasing and user defined algorithm, *Winter Simulation Conference (WSC'02) 2002* -, vol. 2, pp.1269-1273.

[17] Mascle, C. and Balasoiu B. A., Disassembly-assembly sequencing using feature-based life-cycle model. *Proceedings of 2001 IEEE International Symposium on Assembly and Task Planning*, pp. 31-36.

[18] Sundaram S., Remmler I. and Amato N. M., Disassembly sequencing using a motion planning approach. *Proceedings of 2001 IEEE International Conference on Robotics and Automation*, 1475-1480.

[19] Trouillet B., Benasser A. and Gentina J-C, Transformation of the Cyclic Scheduling Problem of a Large Class of FMS into the Search of an Optimized Initial Marking of a Linearizable Weighted T-System, *Sixth International Workshop on Discrete Event Systems (WODES'02) 2002*, pp.83.

[20] Trouillet B., Dupas R., Goncalves G. and Hsu Tiente, Two approaches to the cyclic scheduling with assembly, *12th IFAC Symposium on Information Control Problems in Manufacturing, INCOM 2006*, Saint-Etienne (France).

[21] Trouillet B., Korbaa O, Gentina J-C, Formal Approach for FMS Cyclic Scheduling , IEEE SMC Transactions, 2007, Part C, Vol. 37, Issue 1, pp. 126-137.