UNIVERSITÀ DEGLI STUDI DI GENOVA

Production Systems

Henri Lefebvre

Contents

Introduction

Any system can be regarded as some sort of "black box" or "machine" that turns a certain number of inputs into one or multiple output. Production systems do not escape that definition and can be viewed as a system transforming raw materials into final products. Actually, the inputs are more general and complex than just raw materials, for example, one can consider energy, workforce, decisions and many others as inputs of our system. As well, the outputs are generally not only a final product but also scraps or waste. Scrap corresponds to the portion of raw material that we loose with respect to the technology we utilize for transforming the raw materials into final products. Wastes, on the other hand, are immaterial. The figure (1) sums up what have been said.

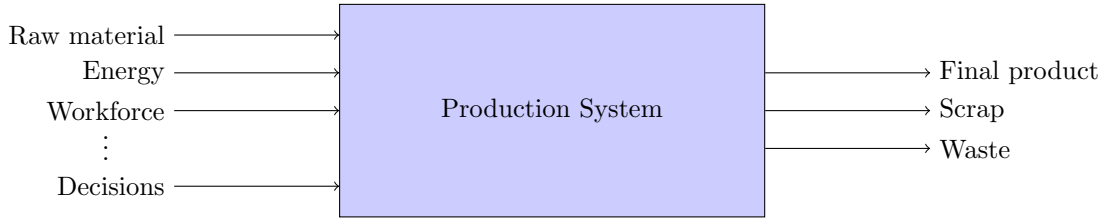


Figure 1: General production system

The objective of this course is to analyse how we can make our production system more efficient with respect to some criteria (time, money, ...). Of course, our production depends on the methods and on the technolgies we use to manufacture our goods, but these are rather concerns for mechnanical engineers or more specific fields (chemistry, process engineering, ...). Actually, this course focuses on the only input we can, at our level, act uppon, which is : decisions.

To understand how decisions can improve (or worsen) our productivity, let's take a very simple example. Consider the following task : "Drilling four holes in a piece of wood". And let X be the rate of production. Consider as well that the "drilling" operation needs some set-up time in order to prepare the right borer and to turn on the machine. Let's denote by T_S this set-up time and by T_H the operation time which is necessary to actually drill the wooden piece. There are plenty of ways to perform such task, one idea is to drill one hole at a time, stopping the machine after each drill. But this wouldn't be a very clever way since we would have to wait for the set-up time for each hole we want to make. Another idea is to perform a batch of holes in a row so that we only account for the set-up time once for all the batch. The two situations are presented in the GANT diagram in figure (2).

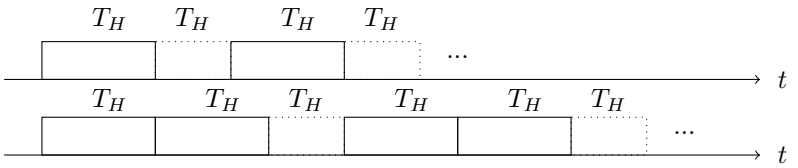


Figure 2: Two GANT diagrams showing how decisions can influence productivity

In the first situation, the rate of production can be expressed as

$$X_1 = \frac{1}{T_H + T_S}$$

meaning that we produce one item every $T_H + T_S$ unit of times. For the second situation however, the rate can be computed as

$$X_2 = \frac{2}{2T_H + T_S} = \frac{1}{T_H + \frac{T_S}{2}}$$

which means that we produce one item every $T_H + \frac{T_S}{2}$ unit of time, which is smaller. In fact, as long as the batch size increases, the set-up time becomes less and less influent on the production rate since

$$X_k = \frac{k}{kT_H + T_S} \overset{k\to\infty}{\longrightarrow} \frac{1}{T_H}$$

which clearly demonstrates that we can influence our production by making some decisions rather than others.

The first part of this course will deal with the inventory management in which we'll try to answer to questions like "How and when should we order goods for our production system ?". In fact, we will deal with the different stakeholders to establish good strategies to make our orders. These stakeholders are represented in figure (3). Firstly, the suppliers provides us with some raw materials which can then be stored in a warehouse. When needed, the production system will take some of these materials to transform them into final products. Once done, he will then leave them in another warehouse which will be used to fulfill the demand of such final goods.

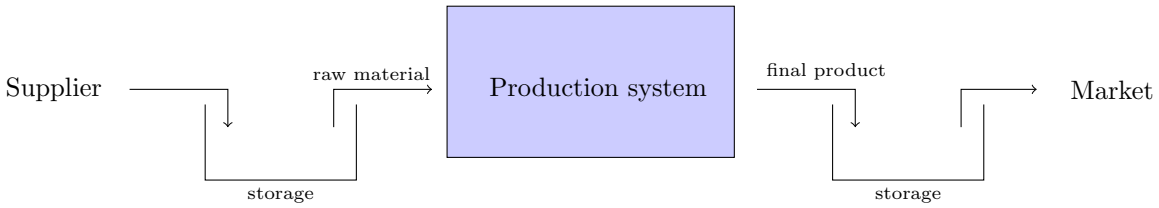


Figure 3: Production system agents

That being said, it is clear now that any production system aims to fulfill a certain demand taking into account the supply constraints it may encounter. This will be largely discussed futher on.

The figure (4) depicts the different production system types we can encounter :

- Job shops : very specialized entities where, often, solely one product can be produced at the same time. Car repairers are a good example of such production systems. This kind of system is not easily formallized because they are too specific. We will not discuss them in this course.
- Mass production : produces an average number of varieties of goods at average to high rate.
- Continuous flow process : produces continuously a given product at very high rate but with very small variety of products.

We will focus on both mass production and Continuous flow processes.

The second part of this course will take interest in the bill of materials (which will be further defined) in which we'll look at how to optimize production with respect to the production rate taking into account bottleneck machines in a production site.

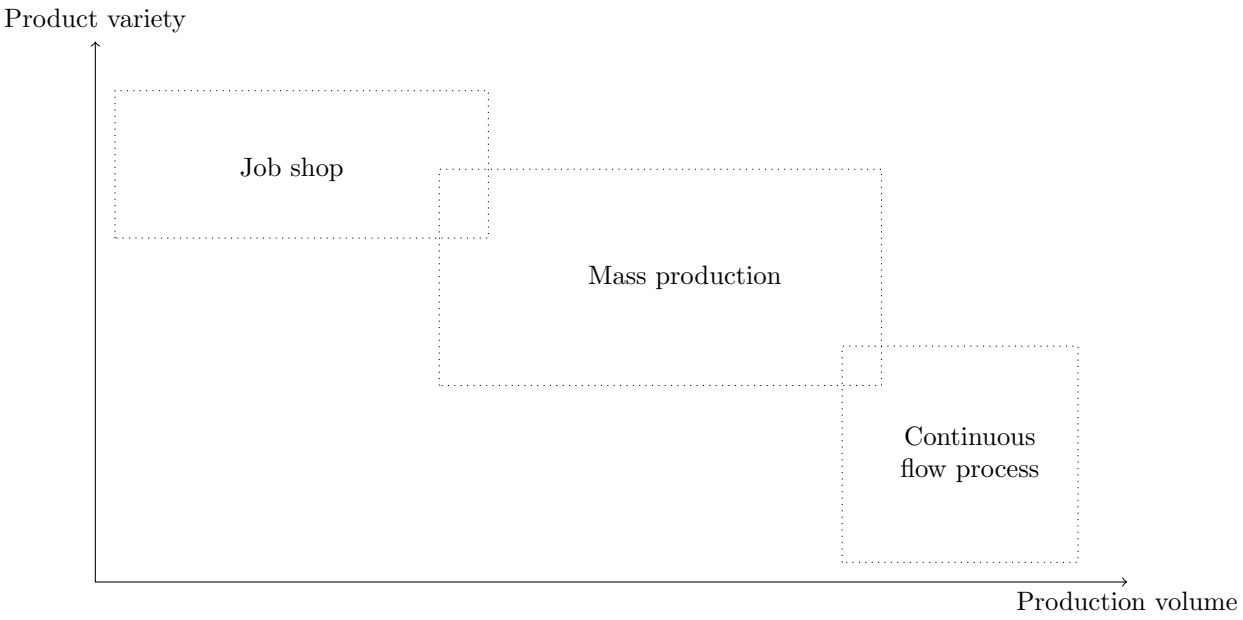


Figure 4: Production system types

Part I

Inventory management

Chapter 1

Continuous review

Inventory management, or storage management, can be qualified as the art of storing and ordering goods in a clever way with respect to some criteria (most often the cost). It can be applied at both the beginning and the end of the production system (even in between actually). At the beginning of this chain, raw materials are delivered by the suppliers and are stored in a warehouse. The storage is emptied when the production site requires some raw materials to be used for production. On the other side of the chain, at the end, the storage is filled with final products and is emptied by the market demand. Though the two situation may appear different in some kind, we can jointly study them considering some sort of "abstract demand" (either production demand or market demand) and some sort of "abstract supplier" (either a real supplier which provides raw material or the production site itself). See figure (1.1) for graphical representation.

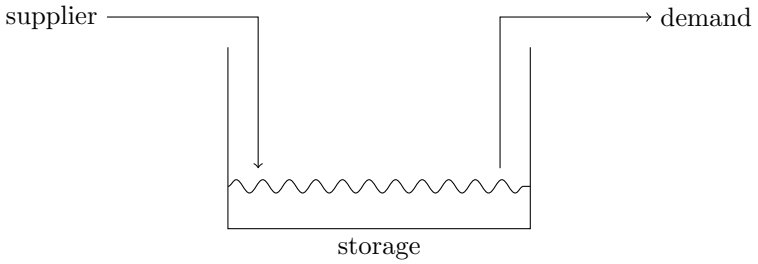


Figure 1.1: Storage management parameters

To study the inventory management, we introduce models that tend to represent real life cases. Making different assumptions on the demand and supplier, we end with different models. The simplest model is the so called "deterministic model" where all the input data for our problem are supposed to be perfectly known in advance (both the supply and the demand). We will start by studying this model before focusing on a "stochastic model" where the supplier and the demand are modeled by random variables.

In this chapter, we will study what is called "continuous inventory management". What we mean by this is that the time is modeled by a continuous variable ($t \in \mathbb{R}$). In the next chapter, we'll deep into "periodic inventory management" where the time is modeled by a discrete variable ($t \in \mathbb{N}$).

1.1 Deterministic model

The deterministic model supposes that we exactely know both the demand and the characteristics of our supplier. We denote by λ the rate at which the storage is emptied. In the case where our warehouse is at the end of our chain,

just after the production site, λ represents how many items (final products) we sell by unit of time. If considered at the beginning of the chain, it corresponds to how many items (raw materials) are needed by our production site. This coefficient can be computed as

$$\lambda = \frac{\text{nb of items leaving the warehouse}}{\text{observed time}}$$

To fullfill that demand, we need to order a certain quantity of good, which will cost us some amount of money. In fact, the costs are of three kinds :

- A **fixed cost** which we have to pay for each order we make to our supplier. Let K denote this cost, expressed as $euros/order$
- An **inventory cost** which we pay for storing goods in our warehouse, expressed as $euros/(time \times item)$. Let h be the inventory cost.
- A **unitary cost** which corresponds to the price of each item per se. We will refer to this cost as c, expressed as $euros/item$

For sure, it is easy to see that choosing the right time and quantity to order goods is a matter of balancing between the fixed cost we pay for ordering an independent amount of product and the inventory cost which we pay for storing the goods. If storing products in our warehouse is very expensive for us, we might prefer to order often and pay the fixed cost multiple times rather than paying it just once and be left with a lot of goods we pay a fortune for storing. If, however, the inventory cost is very low (which means that we can store many items for a small amount of money), we'd rather order a lot of products and store them for a long time if needed. So, how do we compute the total cost ?

Let T be the period of time between the moment we order some goods and the moment our storage goes empty. Note that since the demand is deterministic and known in advance there is no risk of "unpredicted" events which could lead us to be unable to fullill the demand nor to order too many items in such a way that some would remain after some time. Finally, let Q be the number of items we will buy at the beginning of that period. The figure (1.2) sums up some of the introduced variables.

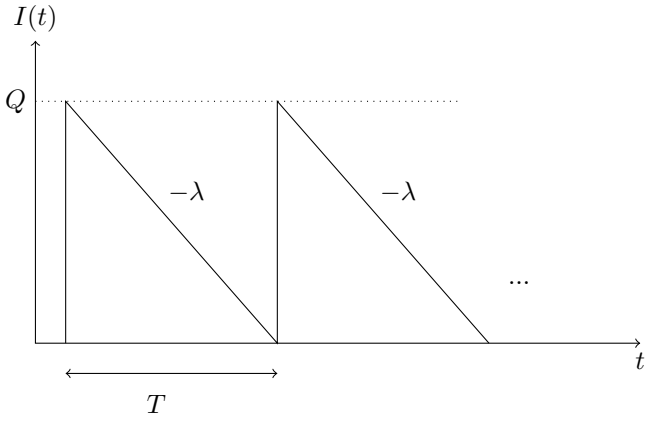


Figure 1.2: Inventory level in the deterministic model

For a given period of time T, we can then compute the total cost as

$$J_T = \underbrace{K}_{\text{fixed cost}} + \underbrace{cQ}_{\text{price for } Q \text{ items}} + \underbrace{\int_0^T hI(t)dt}_{\text{storage cost}}$$

where the integrated part is simply given by $\frac{QT}{2}$ (area of a triangle[1]).

[1] $\int_0^T hIt(t)dt = \int_0^T h \times (Q - \lambda t)dt = h[Qt - \lambda\frac{t^2}{2}]_0^T = h\frac{2QT - \lambda T^2}{2}$ and since $Q = \lambda T$ we get $\frac{QT}{2}$ by substituting λ

For sure, we want to minimize the cost for a given period of time. That cost is given by

$$J = \frac{1}{T}\left(K + cQ + \frac{hQT}{2}\right) = \frac{K\lambda}{Q} + c\lambda + \frac{hQ}{2}$$

Note that we used the fact that $Q = \lambda T$ holds. Regarded as a function of Q, we can analyse its variations by derivation (since $Q = 0$ does not make any sense for our purpose, it is for sure differentiable) :

$$\begin{cases} \frac{dJ}{dQ} & = -\frac{K\lambda}{Q^2} + \frac{h}{2} \\ \frac{d^2J}{dQ^2} & = \frac{K\lambda}{Q^3} \end{cases}$$

Solving the optimality condition $\frac{dJ}{dQ} = 0$ we get that

$$Q^* = \sqrt{\frac{2K\lambda}{h}}$$

can be regarded as a critical point of J. And, since the second derivative of J is positive, the value Q^* corresponds to a minimum of our cost function. This quantity is known as the **Economic Order Quantity** (EOQ) and corresponds to the number of items we should order every Q/λ unit of time in order to minimize the total cost.

We may discuss the obtained formula noting that : (1) if the fixed price for order is very large with respect to the inventory cost, we will by a large amount in once and store it for a long period of time ; (2) if the demand is high, we buy more ; (3) if the inventory cost is large with respect to the fixed cost per order, we will prefer making multiple orders in order to store as less as possible.

The moment at which we should order that quantity Q^* of goods is given by the characteristics of our supplier. Let τ be the lead-time of our supplier (difference between the moment we make an order and the moment we actually recieve the products). We define the so called **reorder point** as $R = \lambda\tau$ which corresponds to the inventory level at which we should call the supplier to make an order[2].

1.2 Stochastic model

In the previous model, we supposed that we could both know perfectly the demand and the lead-time of our supplier. In real life however, such quantities are often subject to variations and are found by estimation. That being said, we can not afford to trust a deterministic formula since we know that life is full of unpredicted and unpredicatable events. Let x be a random variable such that $x \sim \mathcal{N}(0,1)$. We now model the demand (denoted d by convention in the stochastic model) by a random variable given by $d = \bar{d} + \sigma_d x$ and the lead-time by $\tau = \bar{\tau} + \sigma_\tau x$.

Actually, the stochastic model is a generalisation of the deterministic model and a lot of common aspects will remain true in this model (in the sense that we can derive the deterministic case from the stochastic case). But, since the demand now variates with time, we can no longer rely on an exact formula giving us *exactely* how many goods we have to order in order to have a 0-inventory-level when we re-order some other goods. Instead, we'd rather save some goods to prevent the fact of being short on stock. That amount of goods is called the **safety stock** and will be denoted by the variable $s.stock$. Given these informations, what is reasonnable to expect ?

The expected value of the demand (resp. the leadtime) is $\bar{d}$ (resp. τ). The reorder point is thus expected to be given by $\bar{d}\bar{\tau}$ which, in average, would bring the inventory level to 0 before re-ordering. To this quantity, we add the safety stock to prevent us from delays of our supplier.

$$R = \bar{d}\bar{\tau} + s.stock$$

[2]Note that in the deterministic case, directely using the lead-time is strictly equivalent to using the reorder point, however, three arguments to use the re-order point rather than the lead-time : (1) the reorder point is linked to a quantity that the warehouse can directly control, (2) the deterministic case is a model, which can or can not reflect the reality, using the reorder point assures us that if, for unexpected reasons, we get to the reorder point sooner than planned, we can still make an order at the right time for the new period of time, (3) we will keep using that quantity in the stochastic case.

In average, the stochastic model should be exactely the deterministic model shifted up by precisely *s.stock* goods in the inventory. The figure (1.3) represents two realisations of the demand while figure (1.4) represents the expected value of our inventory level.

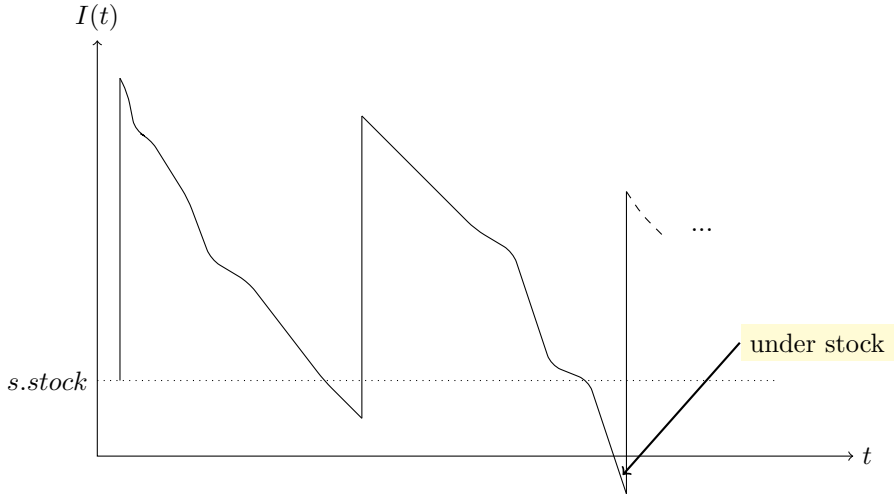


Figure 1.3: Two realisations of the demand

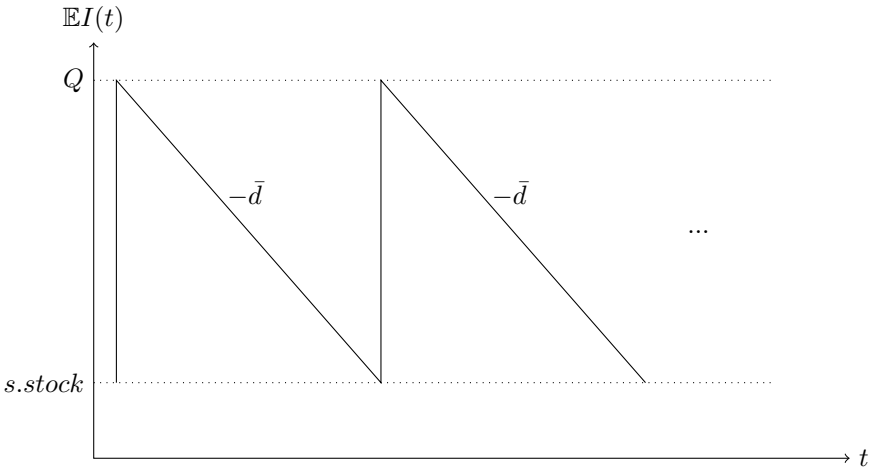


Figure 1.4: Expected inventory level

The expression of the cost must now be updated since we no longer only pay for ordering and storing the goods which are used to fulfill the demand but also for storing some goods as a safety-stock. The total cost is now given by :

$$J = \frac{K\bar{d}}{Q} + c\bar{d} + \frac{hQ}{2} + h \times s.stock$$

This function of two variables (Q and *s.stock*) is actually well splittable into two functions of one variable denoted by $f(Q)$ and $Q(s.stock)$. It is clear that minimizing the total cost can now be done by minimizing both functions seperately. Regarding the first function f, the computations remains the same as in the previous section (deterministic model) and we still have $Q^* = \sqrt{2K\bar{d}h}$. Minimizing g how ever, requires some more analysis. Of course, g is linear with respect to *s.stock* so the global minimum is 0 since it cannot be negative. But setting it to 0 would mean risking to be short on stock at some point. We, somehow, want to control the risks we take and two strategies can be opted for :

1. Let d_τ be the demand over the lead-time τ. It is clear that d_τ is *linked* to both d and τ but it is a rather strong assemption to say that these two random variables are *dependent*. Indeed, the realisation of the leadtime of a supplier is not dependent on the global demand[3]. d_τ is thus given by $d_\tau = \bar{d_\tau} + \sigma_{d_\tau} x$ where $\bar{d_\tau} = \bar{d}\bar{\tau}$ and $\sigma_{d_\tau} = \sqrt{\sigma_d^2 \bar{\tau} + \sigma_\tau^2 \bar{d}^2}$. One idea to select effectively the safety stock is to make sure that the probability of being short on stock is lesser than a certain number which we control. Being short on stock is realised when the demand over the lead-time is greater than the reorder point we have choosen. This strategy implies choosing *s.stock* so that

$$\mathbb{P}(d_\tau \leq R) \geq \alpha = \mathbb{P}\left(x \leq \frac{R - \bar{d_\tau}}{\sigma_{d_\tau}}\right) \geq \alpha$$

where α is called the "fill rate". Using this strategy can be done as follow :

   (a) Choose an acceptable probability α
   (b) Solve the eqution for X and let $\bar{X}$ be the solution
   (c) Subsitute X by $\bar{X}$ in $s.stock = \bar{d_\tau} + \sigma_{d_\tau} X$

2. As it has been previously said, it holds that we are short on stock when $d_\tau > R$ (more demand during the lead-time than the reorder point). Let $n(R)$ be the average number of unused items (items remaining in the warehouse when we re-order again). We can compute this quantity with the following formula[4] :

$$n(R) = \int_R^\infty (d_\tau - R) f(d_\tau) d(d_\tau)$$

   However, it is analytically hard to compute. To deal with that problem, we introduce the function L which is called the loss function expressed as

$$L(X) = \int_X^\infty (t - X) f(t) dt \text{ where } f \text{ is the density of the normal law } \mathcal{N}(0, 1)$$

   so that

$$n(R) = \sigma_{d_\tau} L(X)$$

   Finally, by defining the average percentage of unused items with respect to the total number of goods we buy, denoting it by $\beta = \frac{n(R)}{Q^*}$ we can find *s.stock* so that

$$L(X) = \frac{\beta Q^*}{\sigma_{d_\tau}}$$

   where β should of course be defined *a priori* depending of the market requirements, of course, the lowest β is, the greatest is the service.

   Using this second strategy can be done like so :

   (a) Compute σ_{d_τ}
   (b) Compute $\frac{\beta Q^*}{\sigma_{d_\tau}}$ and use the table to find the corresponding X denoted $\bar{X}$
   (c) Subsitute X by $\bar{X}$ in $s.stock = \bar{d_\tau} + \sigma_{d_\tau} X$

[3] it is surely linked, for example, during peak demands of a given good, one supplier may have difficulties to satisfy all of his clients, but we cannot consider that a truck getting lost, or breaking down depends on the demand

[4] For any random variable X with density function f, the average value of X is given by $\int_0^\infty x f(x) dx$

Chapter 2

Periodic review

In the previous section, time was supposed to be continuous. In this new chapter however, we will regard time as a discrete variable meaning that we consider fixed size time intervals for our inventory management study. To each of these periods corresponds a certain demand, denoted d_i for period i, and a decision (called "action") that we take in order to satisfy the demand, denoted a_i. We represent this situation with plots similar to figure (2.1) where each box stands for a period of time. The question we have to answer in this chapter is : "What actions do I have to take in order to fulfill the demand minimizing the total cost ?", or similarly : "How many items a_i should I buy in period i to minimize the cost ?"

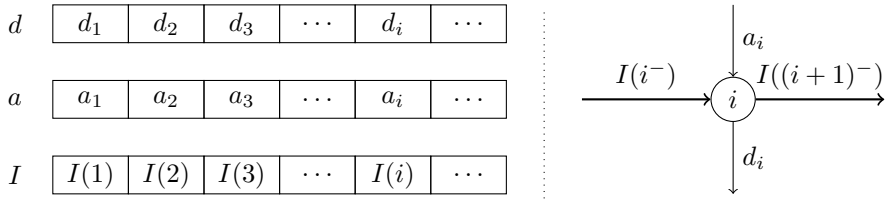


Figure 2.1: A representation of the periodic review model

With this representation, it becomes clear that, at the end of the i-th period, the inventory level, denoted by $I((i+1)^-)$, corresponds to the inventory level we had at the beginning of that period, $I(i^-)$, to which we should add the number of items we have ordered, a_i, minus the demand, d_i, which corresponds to goods which have been sold out. Hence,

$$I((i+1)^-) = I(i^-) + a_i - d_i$$

Thus, by denoting by h the inventory cost and by K the fixed cost per order as we have been doing so far, the total cost can be computed as :

$$J = \sum_{i=1}^{N} \left(K\delta(i) + hI((i+1)^-) \right) \text{ where } \delta(i) = \begin{cases} 1 & \text{if } a_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

Our goal is to minimize this function by choosing relevant a_is. For example, we need to discuss whether it is more profitable to order goods for two periods in a row instead or ordering period per period with respect to the demand. This can be solved by mixed integer linear programming models in an exact way. However, the cost of these methods are not worth it since efficient Heuristics have been developed which are very simple to implement. In this chapter, we will focus on the "Silver-Mill Heuristic".

Even though it appears clear, note that, if the inventory level in period 0 is 0 and if we suppose knowing in

advance the exact demand for each period, then it holds that

$$\sum_{i=0}^{N} a_i \geq \sum_{i=0}^{N} d_i$$

This can easily be demonstrated by recursivity. Let's suppose that $I_0 = 0$. Then, and because $I_i \geq 0, \forall i$ in the deterministic model, $I_1 = a_1 - d_1 \geq 0$ which means that $a_1 \geq d_1$. So the property is true for the first rank. Let's now suppose, that, for a given rank k, we have $\sum^k a_i \geq \sum^k d_i$. Considering I_{k+1}, it holds that $I_{k+1} = I_k + a_{k+1} - d_{k+1}$ with $I_k = \sum^k a_i - \sum^k d_i$ so that in definitive $\sum^{k+1} a_i \geq \sum^{k+1} d_i$.

2.1 The Wagner-Within Theorem

The Wagner-Within Theorem tells us about the shape of the solution to our problem. It stipulates that the optimal solution is either that (1) we buy exactely the demand of multiple time periods or (2) we buy exactely the current period demand, but in no case we would buy a fraction of the demand for one time period and buy the rest in another period. The optimal solution cannot be splitted into periods of time. For instance, the optimal solution cannot be to buy 70% of the forecasted demand of November in September and 30% of that same demand in October, but rather buy all the corresponding demand for November in September or all the corresponding demand for Novemeber in October. We do not split the orders for one time period. So that, considering a_1 for example, we now know that

$$a_1 \in \{d_1, d_1 + d_2, d_1 + d_2 + d_3, \ldots\}$$

In this section we will demonstrate this theorem for the case of two time periods **WLOG**.

Let's consider a time horizon of two periods and, since the demand is supposed to be known in a deterministic way, we can impose that the inventory level at the end of the horizon is zero, $I_2 = 0$. As well, we'll assume that I_0 is zero (no initial inventory level). The inventory level at the end of this time horizon is given by $I_2 = a_1 + a_2 - d_1 - d_2 = 0$ which leads to the equality

$$a_1 + a_2 = d_1 + d_2 \tag{2.1}$$

In the first time period, we at least have to order enough goods to fulfill the demand of the first period, so that $a_1 \geq d_1$ which we'll rather write as $a_1 = d_1 + \epsilon$ where ϵ is a positive number (which in any case should be less than d_2 because it would violate our will to end with an inventory level of zero). From equation (2.1), it holds that $a_2 = d_2 - \epsilon$, and so, at the end of each period we end with :

Inventory level at the end of each periods :

ϵ	0

This allows to compute the total cost like so :

$$J = K + h\epsilon + \begin{cases} K & \text{if } \epsilon = 0 \\ 0 & \text{if } \epsilon = d_2 \\ K & \text{if } \epsilon < d_2 \end{cases} = \begin{cases} 2K + h\epsilon & \text{if } \epsilon < d_2 \\ K + hd_2 & \text{if } \epsilon = d_2 \end{cases}$$

Indeed, we always pay at least one order (which corresponds to the first order in period one) and the inventory cost for the remaining items after the first periods (which is ϵ). In the case where $\epsilon = d_2$, we already have ordered enough products to fulfill both the demand of the first period and the demand of the second period, so we don't have to make another order and we don't pay for ordering nothing. Otherwise, we have to order again, so a new K cost has to be paid. We then have two cases :

- if $\epsilon < d_2$, then clearly, the minimum is reached by choosing $\epsilon = 0$
- if $\epsilon = d_2$, then the minimum is $K + hd_2$

What this discussion means, is that, in order to choose a value for ϵ in order to minimize our cost, the only "good" decisions we can make are those of buying once for two periods or buying once at each period but in no case should we buy more in period one and buy again a smaller amount of goods in period two. That's the meaning of the Wagner-Within theorem.

2.2 The Silver & Mill Heuristic

The Silver-Mill heuristic is a very simple algorithm which relies on the Wagner-Within theorem. It often drives a solution near the real solution which in most cases is acceptable. In fact, the algorithm finds a local minimum of the cost function (instead of a global minimum). Simply sumed up, this heuristic considers incrementing periods of time and looks at whether or not it would be a good idea to buy more than what is needed for that specific period of time. We know from the Wagner-Within theorem that if we buy more than needed, then whe should buy exactely the amount of good sufficient to satisfy the demand of the next period (otherwise it's not worth it). The algorithm computes a series of costs making this kind of assumptions until a local minimum is reached. Here is how it works : let $c(i)$ be the cost we would pay if, starting from the first period, we were to buy as many products as needed to statisfy up to the i-th period's demand. Clearly, the cost $c(1)$ is simply K since we only make one order and don't pay for inventory since all our order is consumed by the demand in the first period ($a_1 = d_1$). For the second period, the cost $c(2)$ is now the fixed price K, since we order once for two periods, to which we have to add the inventory cost we pay for the remaining inventory level after period one which corresponds exactely to hd_2. In order to compare $c(1)$ and $c(2)$, we'll focus on the *average* cost that we pay per period of time instead of the absolute value. It holds then that $c(2) = \frac{K+hd_2}{2}$. Similarly, we can compute $c(3)$ as $c(3) = \frac{K+h(d_2+d_3)+hd_3}{3} = \frac{K+hd_2+2hd_3}{3}$ since we order once for the three subsequent periods of time and that we have to pay for inventory in period one (whose inventory level is then $d_2 + d_3$) and in period two (whose inventory level is then d_3). Continuing like so, and searching for a local minimum, we get the Silver-Mill heuristic for choosing our actions.

The general formula (which can be recursively demonstrated) for computing the coefficients $c(i)$s is given by

$$c(i) = \frac{K + hd_2 + 2hd_3 + \ldots + (i-1)d_i}{i}$$

Indeed, we assume paying once in period one, so we pay the order price K just once, then we have to deal with inventory costs. The quantity d_k will remain in the inventory untill period k is reached. So the associated inventory cost is given by $(k-1)hd_k$ since we pay for having it in store in every preceding periods of time.

The figure (2.2) shows how the heuristic works. The algorithm continues to compute total costs (incrementing the number of periods considered) until a local minimum is found. In this example, our action will be to order in the first period for three periods of time since $c(3)$ is a local minimum of our function J. Note that, if we had continued to search for lower values we may have eventually found that (in this case), $c(5)$ was more advantageous from an economical point of view and we would have better done to buy all the products for 5 periods instead of four. But in general, this approach gives a really good approximation of the optimal value for the problem and, with respect to the simplicity of the algorithm, it often doesn't worth it to use more advanced and complex models.

2.3 Analysis of the method

In this last section of this chapter, we would like to discuss on the accuracy, or rather plausibility, of this method. In particular, we will have a look at what happens if the demand for each period of time is homogeneous. Let d be

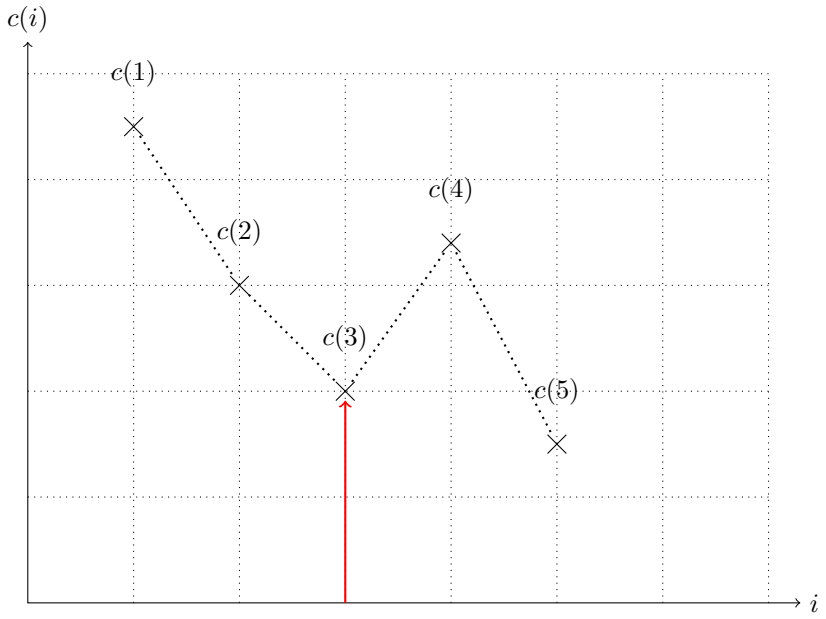


Figure 2.2: Silver-Mill heuristic example

that demand. Using the previously established formula, it holds that :

$$\begin{aligned} c(n) &= \frac{K + hd + 2hd + \ldots + (n-1)hd}{n} \\ &= \frac{K + \sum_{i=1}^{n-1} ihd}{n} \\ &= \frac{K + \frac{n(n-1)}{2}}{n} \\ &= \frac{K}{n} + \frac{nhd}{2} - \frac{hd}{2} \end{aligned}$$

and let us relax our problem considering the function $c(x)$ where x is no longer discrete but continuous. Derivating this function, we get

$$\begin{cases} \frac{dc}{dx} & = -\frac{K}{n^2} + \frac{hd}{2} \\ \frac{d^2c}{dx^2} & = \frac{K}{n^3} > 0, \forall x \in \mathbb{R} \end{cases}$$

The second derivative being greater than zero for any x means that the function c is convex. Knowing that, it is legit to compute the minimum of this function by truncating the solution of $\nabla c = 0$ to find the actual minimum of the non-relaxed problem (where $x \in \mathbb{N}$). Solving this equation we get :

$$\frac{dc}{dx} = 0 \Leftrightarrow x^* = \sqrt{\frac{2K}{hd}}$$

And since $Q = dn^*$, it holds that

$$Q = dx^* = \sqrt{\frac{2kd}{h}}$$

which is the exact same formula which we found in the first part of this course in the deterministic model.

Part II

Bill of materials

Chapter 3

Model without set-up times

Looking back at figure (1), we realise that we have only been focusing on one part of the global schema. Indeed, we have only dealt with inventory management so far. This part of the course will now focus on what is called "Bill of materials" in which we look at how we can manage the relations between raw materials and final product production.

3.1 Model representation and explosion of bill of materials

The Bill of materials uses a standard representation in terms of graph. The nodes are a set of products : raw materials, final products or temporary products. Temporary products are products which are used to make either other temporary products or final products from either other temporary products or raw materials. The arcs of the graph represents transformations. In figure (3.1), the first case where a node i is linked to a node j represents the fact that, somehow, product i is transformed into product j, for instance, if we paint a piece of wood (product i) it becomes a painted piece of wood (product j), or again, if we cut a large piece of iron (product i) we transforme it into *several* smaller pieces of iron. The number n_{ij} valuating the arc tells us how many product i we need to produce a single product j. If we consider the painting operation, then $n_{ij} = 1$, whereas if we consider cutting an iron piece, $n_{ij} \geq 2$ (depending on the situations). The second case represents the fact of assembling products i and j in order to make product k. Again, n_{ik} represents the number of product i needed to make one product k.

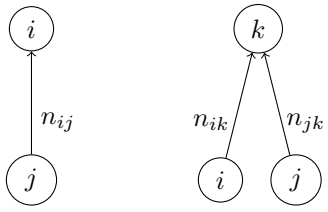


Figure 3.1: Bill of materials representation in terms of graph

Let's now consider figure (3.2) which shows you a first example of bill of materials and let's extend the notation n_{ij} to any nodes i and j as long as there exists a path from i to j in the graph. Then, n_{2f} represents the number product 2 needed to produce 1 final product f. How can we compute that value ? It is rather simple : we need 3 units of product 1 to produce 1 unit of product f, but we need 2 unit of product 2 to produce 1 unit of product 1 ; the quantity n_{2f} is then given by $n_{21} \times n_{1f} = 2 \times 3 = 6$, we need 6 units of procut 2 to produce 1 unit of product f. In general, the following formula holds :

$$n_{kf} = \prod_{(i,j) \in Path(k \to f)} n_{ij}$$

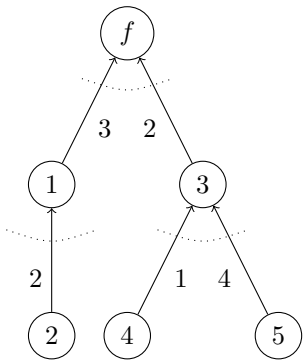


Figure 3.2: First example of Bill of Materials

It is clear now that if we want to produce a certain number of product f denoted X_f, we need to produce at least $X_f n_{if}$ product i so that $X_i \geq X_f n_{if}$. In the following discussion, we will always choose $X_i = X_f n_{if}$ which corresponds to the "conservation of flow" in our system (even though it is not a necessary condition). An other assumption we will make in this chapter, as well as in chapter 2, is that every machine can produce only one kind of product. We will then denote by i both the product and the machine producing it.

Considering machine i, let μ_i be the maximum speed for producing one product, expressed as items/unit of time, and we define u_i as the "utilization of the resource i" which equals :

$$u_i = \frac{X_f n_{if}}{\mu_i} \leq 1, \forall i$$

Using that definition, it holds that

$$X_f \leq \frac{\mu_i}{n_{if}}, \forall i \Leftrightarrow X_f \leq \min_i \left(\frac{\mu_i}{n_{if}} \right) \triangleq X_f^{max}$$

where X_f^{max} is the maximum rate at which we can produce product f with respect to the machines we use to produce it. The bottleneck machine(s) can be computed as $\text{argmin}_i \left(\frac{\mu_i}{n_{if}} \right)$. This (or these) machine(s) is (are) the one(s) which prevents us from producing more, at a higher rate.

3.2 Production time

Given a machine i, we can compute the operation time needed to produce one item of product i which is simple

$$T_{oi} = \frac{1}{\mu_i}$$

We then enhance figure (3.2) with those operation times as a valuation of the arcs. Figure (3.3) shows such a representation. The operation time has been added as the second element of a couple. The first element will be dealt with in chapter two of this section. The maximum rate X_f^{max} at which we can produce our final product is then given by

$$X_f^{max} = \min_i \left(\frac{1}{T_{oi} n_{if}} \right)$$

Let's start by computing the n_{if} quantities for all i, we get :

$$\begin{array}{lll} n_{1f} = 3 & n_{3f} = 2 & n_{5f} = 4 \times 2 = 8 \\ n_{2f} = 2 \times 3 = 6 & n_{4f} = 1 \times 2 = 2 & \end{array}$$

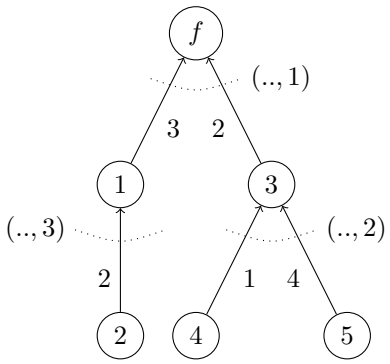


Figure 3.3: First example of Bill of Materials

then we can find X_f^{max} as well as the bottleneck machine by doing the following

$$\begin{aligned} X_f^{max} &= \min_i \left(\frac{1}{T_{oi} n_{if}} \right) \\ &= \min \left(\frac{1}{1 \times 3}, \frac{1}{3 \times 6}, \frac{1}{1 \times 2}, \frac{1}{2 \times 2}, \frac{1}{2 \times 4} \right) \\ &= \min \left(\frac{1}{3}, \frac{1}{18}, \frac{1}{2}, \frac{1}{4}, \frac{1}{8} \right) \\ &= \frac{1}{18} \end{aligned}$$

$$bottleneck = \arg\min_i \left(\frac{1}{T_{oi} n_{if}} \right) = 2$$

Note that, in this case, the bottleneck machine is the one with the highest operation time. This is NOT a general rule, and it's easy to find a counter example : consider a final product f which is made by assembling a hundred small pieces with one big one. Clearly, even if the machine which produces the small pieces is efficient, it is *possible* that making a hundred of these pieces may be longer than producing one other piece, even bigger. But again, this isn't always the case. The only way to solve for the bottleneck machine problem is to look at both the operation time and the number of products required to produce the final product.

3.3 Batch production

A batch b_i can be defined as the number of items i the machine produces without interuption before releasing it to other machines. Let's suppose that we want to produce a batch of final product b_f, how much time do we need to do it ? In this section, we will try to answer that question considering that our batch sizes are $b_i = b_f n_{if}$, which is not necessary, but simpler. In general however, you often have external constraints (linked to the technology you use for example) which imposes a minimum batch size or a maximum batch size, so that :

$$b_i^{min} \leq b_i \leq b_i^{max}, \forall i$$

From that constraint, we can deduce the following

$$\begin{aligned} &\Rightarrow b_i^{min} \leq b_f n_{if} \leq b_i^{max}, \forall i \\ &\Rightarrow \frac{b_i^{min}}{n_{if}} \leq b_i \leq \frac{b_i^{max}}{n_{if}}, \forall i \\ &\Rightarrow \max_i \left(\frac{b_i^{min}}{n_{if}} \right) \leq b_f \leq \min_i \left(\frac{b_i^{max}}{n_{if}} \right) \end{aligned}$$

Since the batch size b_f has to be an integer, we have to make sure that this interval contains at least on integer value, otherwise, we cannot use $b_i = b_f n_{if}$ as a batch size.

Considering that relation to hold however, we can compute the production time for a given batch as

$$T_{i,prod}(b_i) = T_{oi} b_i = T_{oi} b_f n_{if}$$

which now allows us to compute the time it takes to go from one raw material (a leaf of the tree in our graph representation) to the final product (root of tree). Considering $\mathcal{P}_k$ as the path going from the raw material k to the final product in our graph, it holds that

$$T_{\mathcal{P}_k} = \sum_{j \in \mathcal{P}_k} T_{j,prod}(b_j) = b_f \sum_{j \in \mathcal{P}_k} T_{oj} n_{jf}$$

This relation tells us about how much time we would need to produce a batch of final product b_f since

$$T_{prod}(b_f) = \max_{\mathcal{P}_k} \left(T_{\mathcal{P}_k}(b_f) \right)$$

Up to now, we have considered choosing a batch size b_f and compute the time needed to produce it. But we can as well use the established formulas in the other way to compute the maximum batch size allowed in order respect a certain constraint on production time. For instance, let's say we want to respect the condition $T_{prod}(b_f) \leq T^{max}$ meaning that we do not want to produce more than T^{max} units of time. Then it still holds that

$$\begin{aligned} T_{prod}(b_f) \leq T^{max} &\Leftrightarrow b_f \max_{\mathcal{P}_k} \left(b_f \sum_{j \in \mathcal{P}_k} T_{oj} n_{jk} \right) \leq T^{max} \\ &\Leftrightarrow b_f \leq \frac{T^{max}}{\max_{\mathcal{P}_k} \left(\sum_{j \in \mathcal{P}_k} T_{oj} n_{jk} \right)} \end{aligned}$$

which tells us about the maxumum size we can use for a batch in order to fullfill that constraint.

One final consideration is to be taken regarding the rotation of batches. Let's say we want to produce $X_f^* \leq X_f^{max}$ final products. We introduce the variable D as the difference in time between the moment a batch is produced and the moment a subsequent batch is produced so that

$$X_f^* = \frac{b_f}{D}$$

(number of items of the final product over the time of a "period" of production) The figure (3.4) illustrates this notion.

Figure 3.4: GANT diagram with production cycle

Chapter 4

Model with set-up time considerations

Up to now, our considerations were only focused on the operation times for making products and the reciepee of the final product. In this chapter, we will start complexifying our model by adding the setup time for machines. The setup time corresponds to the time in which the machine isn't used to actually produce one good but rather to "get ready" to do so. For instance, one machine has a "cold start" or may need some preparation like changing drills or configurations. We will see in this chapter that the setup time is actually a critical aspect of the production plant and that it drives the batch sizes in a very precise way. The figure (4.1) shows how the setup time impacts the production of a given batch.

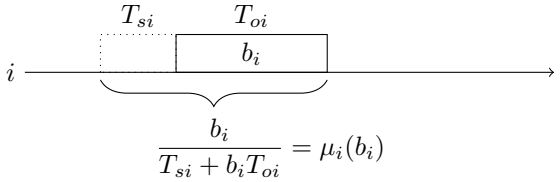


Figure 4.1: Simple example of batch production with setup time in GANT representation

Concerning the rate of production of mahcine i as depicted in figure (4.1), a few considerations are to be made : (1) of course, if we set the setup time to be zero we get back with our previous formula $\mu_i = 1/T_{oi}$; (2) when $b_i \to \infty$, we also get that $\mu_i(b_i) \to \frac{1}{T_{oi}}$, this can be simply explain as the fact that, if we produce more and more products in one batch, the setup time (which is paid just once), becomes less and less significant with respect to the production time.

As far as the representation in terms of graph (bill of materials) is concerned, we will now fill the first element of the couple valuating transformations with the setup times for each machine as you can see in figure (4.2).

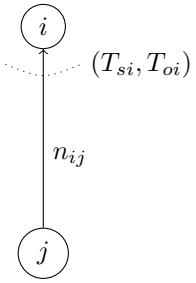


Figure 4.2: Simple example of batch production with setup time in GANT representation

4.1 Finding the bottleneck machine(s)

But let's get back to our prior concerns : production rates. Of course the following fomrula still holds :

$$X_f^{max} = \min_i \left(\frac{\mu_i}{n_{if}} \right)$$

As we did in the last chapter, we will consider setting the batch size as exactely $b_i = n_{if} b_f$ so that

$$X_f^{max} = \min_i \left(\frac{b_f}{T_{si} + T_{oi} b_f n_{if}} \right)$$

which in trun is no longer a constant but rather a function of b_f. We will denote it as $X_f^{max}(b_f)$ not to forget it. The meaning of this is that now the maximum rate of production depends on the batch size we use with respect to the characteristics of the machines we use. And it is somehow logical since we know that producing batches of 1 item would mean "paying" for the setup time for 1 operation time whereas using batches of 10 items would mean paying once the setup time for 10 operation times ! As well, the overall production time is influenced by the setup times in a very simple way :

$$T_{prod}(b_f) = \max_{\mathcal{P}_k} \left(\sum_{i \in \mathcal{P}_k} (T_{si} + T_{oi} b_f n_{if}) \right)$$

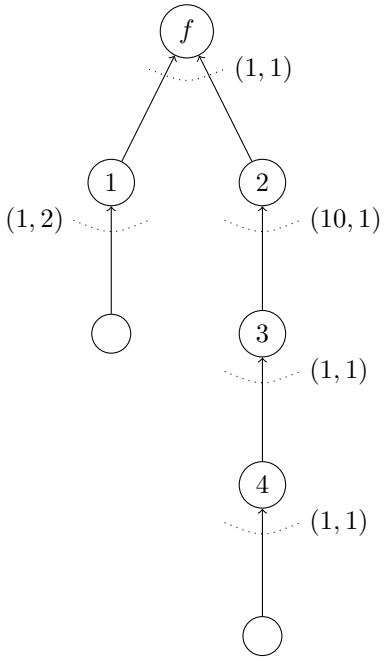


Figure 4.3: Bill of material with setup time example

Let's have an example to better understand what it means to have a maximum production rate which depends on the batch size. Let's consider the bill of materials of figure (4.3) and let's compute X_f^{max}. Note that when no n_{if} coefficient is written, it is assumed to be 1.

Applying the given formula, we gen that

$$\begin{aligned} X_f^{max} &= \min_i \left(\frac{b_f}{1 + 1.b_f}; \frac{b_f}{1 + 2.b_f}; \frac{b_f}{10 + 1.b_f}; \frac{b_f}{1 + 1.b_f}; \frac{b_f}{1 + 1.b_f} \right) \\ &= \min_i \left(\frac{b_f}{1 + 1.b_f}; \frac{b_f}{1 + 2.b_f}; \frac{b_f}{10 + 1.b_f} \right) \\ &= \min_i \left(\frac{1}{1 + 2.b_f}; \frac{1}{10 + 1.b_f} \right) \end{aligned}$$

You may notice that in line 2 of these equations, we got rid of the first value because, in any case, we knew it was lower than the other ones since $1 + 1.b_f < 1 + 2.b_f$ and $1 + 1.b_f < 10 + 1.b_f$. The discussion now continues with respect to the machine 1 and 2 and we should look at figure (4.4) which plots the functions $b_f \mapsto 1 + 2b_f$ and $b_f \mapsto 10 + b_f$. The bottleneck machine is then given by $\arg\min_i \left(\frac{1}{1+2.b_f}; \frac{1}{10+1.b_f} \right) = \arg\max_i (1 + 2.b_f; 10 + 1.b_f)$ which corresponds to the second machine (machine 2) as long as the bacth size is less than 9 and corresponds to the first machine (machine 1) when $b_f > 9$.

Similarly, the production time for producing a batch of size b_f can be computed as

$$\begin{aligned} T_{prod}(b_f) = \max((1 + 2.b_f) + (1 + 1.b_f); (1 + 1.b_f) + (1 + 1.b_f) \\ + (10 + 1.b_f) + (1 + 1.b_f)) \\ = \max(2 + 3.b_f; 13 + 4.b_f) \end{aligned}$$

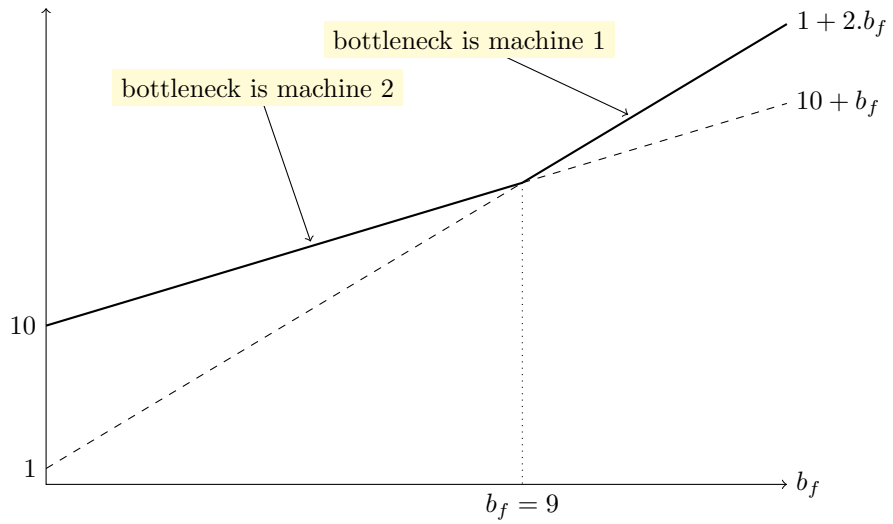


Figure 4.4: Bottleneck machine depending on the batch size

4.2 Controlling the production rate

Let's get deeper with the interpretation of what have been said by looking at both figure (4.8) and (4.9). The first figure shows a situation in which we would like compute the batch size of our production by imposing a certain production rate X_f^* so that $X_f \geq X_f^*$. Solving this problem is, in fact, rather simple and can be done like so

$$\begin{aligned} X_f \geq X_f^* &\Leftrightarrow \min_i \left(\frac{b_f}{T_{si} + b_f T_{oi} n_{if}} \right) \geq X_f^* \\ &\Leftrightarrow \frac{b_f}{T_{si} + b_f T_{oi} n_{if}} \geq X_f^*, \forall i \\ &\Leftrightarrow b_f \geq X_f^* (T_{si} + T_{oi} b_f n_{if}) \\ &\Leftrightarrow b_f \geq \frac{X_f^* T_{si}}{1 - X_f^* T_{oi} n_{if}}, \forall i \\ &\Leftrightarrow b_f \geq \max_i \left(\frac{X_f^* T_{si}}{1 - X_f^* T_{oi} n_{if}} \right) = b_f^* \end{aligned}$$

Notice that we used the fact that $1 - X_f^* T_{oi} n_{if} \geq 0$ to go from the third line to the fourth one. This is true because it is equivalent to $X_f^* \geq \frac{1}{T_{oi} n_{if}}$ which has to hold, otherwise, imposing X_f^* would not be feasible.

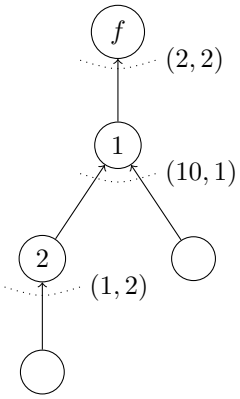


Figure 4.5: Bill of material

Let's take an example in order to see how we can play with these equations. Let's consider the bill of materials presented in figure (4.5) and let us assume that we want to achieve a production rate of at least $X_f^* = 0.9 X_f^{max}(b_f^{max})$. For this example, we will assume that there is no upper bound for batch sizing so that $b_f^{max} = \infty$. It holds that

$$X_f^{max} = \min_i \left(\frac{1}{T_{oi} n_{if}} \right) = \min \left(\frac{1}{2}; \frac{1}{1}; \frac{1}{2} \right) = \frac{1}{2}$$

which tells us that the bottleneck machines are the one denoted by f and by 2, and that the production we actually want to achieve is $X_f^* = .45$. That being said, we can now compute the minimum batch size we can use in order to fullfill this constraint. This is done lke so :

$$b_f^* = \max_i \left(\frac{.45 \times 2}{1 - .45 \times 2}; \frac{.45 \times 10}{1 - .45 \times 1}; \frac{.45 \times 1}{1 - .45 \times 2} \right) = 9$$

Let's assume that we choose to use exactely b_f^* as our batch size (any greater size is actually also valid) to produce at rate X_f^*, exactely. Then, one cycle should lasts

$$D^* = \frac{b_f}{X_f^*} = \frac{9}{.45} = 20$$

And let's have a look at the GANT diagram given by these parameters in figure (4.6). We can see that the bottleneck machine is the machine f and that the machine 2 is no more a bottleneck as we had originally planned. That is because the bottleneck machine actually depends on the batch size we use. When we firstly computed the bottleneck machines (when we computed X_f^{max}), we did it assuming that the batch size was the maximum one, which is no longer true. To make this even more clear, let's deal with the case where we want, with the same production plant, produce at a rate of $X_f^* = .3$ instead of 0.45. But let us still use batches of 9.

First, let's make sure that it is possible to use $b_f = 9$ by computing the lowest batch size we can use with respect to the constraint $X_f \geq .3$:

$$\begin{aligned} b_f^* &= \max\left(\frac{.3 \times 2}{1 - .3 \times 2}; \frac{.3 \times 10}{1 - .3 \times 1}; \frac{.3 \times 1}{1 - .3 \times 2}\right) \\ &= \max\left(\frac{3}{2}; \frac{30}{7}; \frac{3}{4}\right) \\ &= \frac{30}{7} \approx 4 \end{aligned}$$

This tells us two things : (1) that the batch size has to be greater than 4 (so we can use 9 while respecting our constraints) ; (2) that if we use $b_f = b_f^*$, the bottleneck machine will be machine 1. However, and as discussed, the GANT diagram in figure (4.7), the bottleneck machine is actually still the machine f if we use 9 as our batch size. The production cycle is computed as always by $D^* = \frac{9}{.3} = 30$.

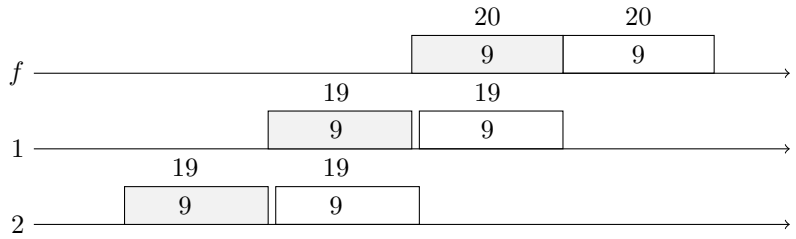


Figure 4.6: GANT diagram when $X_f^* = 0.45$ and batch size is $b_f = b_f^* = 9$

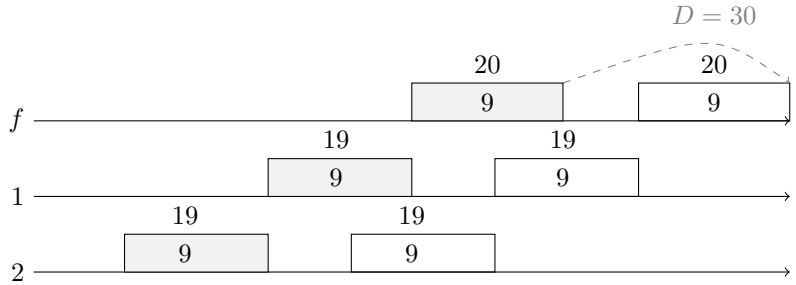


Figure 4.7: GANT diagram when $X_f^* = 0.3$ and batch size is $b_f = 9$

4.3 Controlling the production time

We now can satisfy constraints on the production rate, which is a good thing. However, production plants often encounter another kind of constraints which is linked to the production time. In this section, we will focus on what

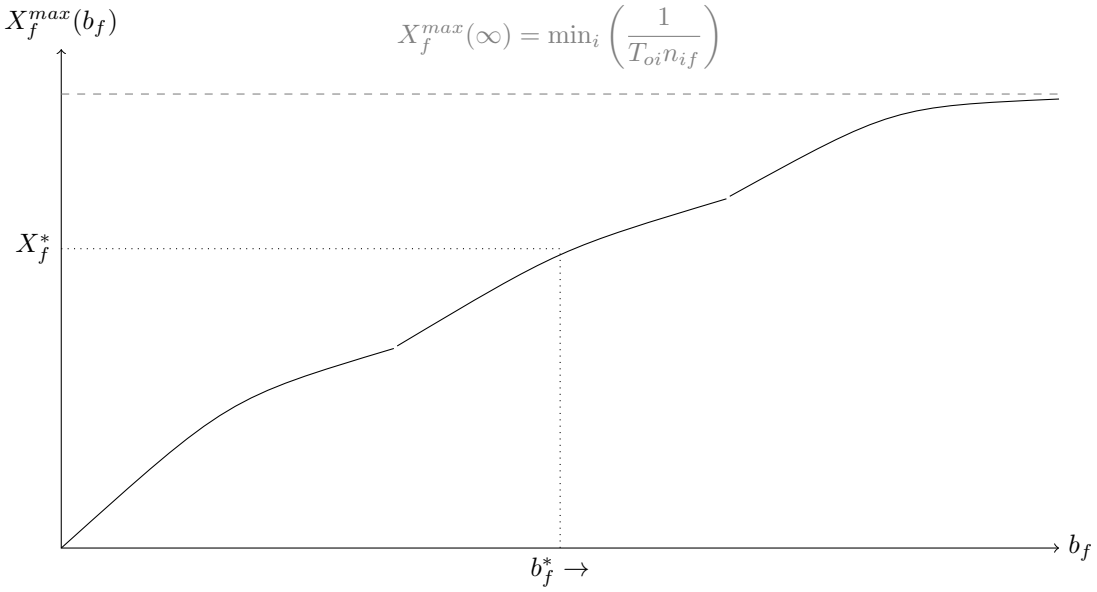


Figure 4.8: How the production rate depends on the batch size

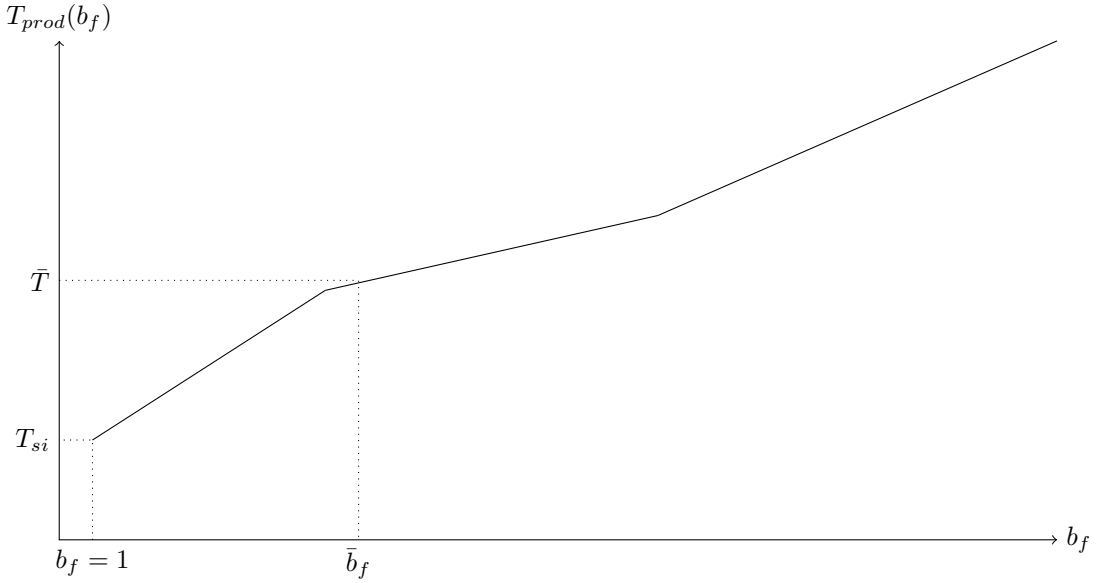


Figure 4.9: How the production time depends on the batch size

happens when we impose that the overall production time must be lower than a certain duration. More formally, we impose that $T_{prod} \leq \bar{T}$. The computation is similar to what have been done so far and is given by :

$$\begin{aligned} T_{prod} \leq \bar{T} &\Leftrightarrow \max_{\mathcal{P}_k} \left(\sum_{i \in \mathcal{P}_k} (T_{si} + T_{oi} b_f n_{if}) \right) \leq \bar{T} \\ &\Leftrightarrow \sum_{i \in \mathcal{P}_k} T_{si} + b_f \sum_{i \in \mathcal{P}_k} T_{oi} n_{if} \leq \bar{T}, \forall \mathcal{P}_k \\ &\Leftrightarrow b_f \leq \frac{\bar{T} - \sum_{i \in \mathcal{P}_k} T_{si}}{\sum_{i \in \mathcal{P}_k} T_{oi} n_{if}}, \forall \mathcal{P}_k \\ &\Leftrightarrow b_f \leq \min_{\mathcal{P}_k} \left(\frac{\bar{T} - \sum_{i \in \mathcal{P}_k} T_{si}}{\sum_{i \in \mathcal{P}_k} T_{oi} n_{if}} \right) = \bar{b}_f \end{aligned}$$

Let's take another example considering figure (4.10) to introduce how we can handle situations in which we both have a condition on production time and on production rate. To do that, consider the following problem : finding b_f subject to the constraints :

$$\begin{cases} X_f \geq X_f^* \\ T_{prod} \leq \bar{T} \end{cases} \quad \text{with } X_f^* = 0.8 \times X_f^{max}(b_f^{max}) \text{ and } \bar{T} = 10 \times T_{prod}(1)$$

Let's first compute the inputs of our problem which are $X_f^{max}(b_f^{max})$ and $T_{prod}(1)$. Since no maximum batch size has been given, we will consider that $b_f^{max} = \infty$ so that

$$X_f^{max}(b_f^{max}) = \min_i \left(\frac{1}{T_{oi} n_{if}} \right) = \min \left(\frac{1}{1}; \frac{1}{2} \right) = \frac{1}{2}$$

and

$$T_{prod}(1) = \max_{\mathcal{P}_k}(25; 5) = 25$$

which gives us the values we wanted : $X_f^* = 0.4$ and $T_{prod} = 250$.

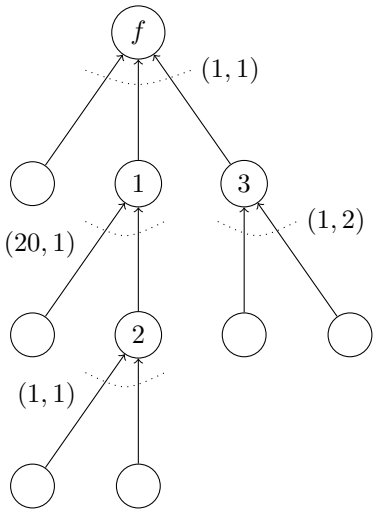


Figure 4.10: Bill of materials

Considering the first constraint (on the production rate), we can compute a lower bound for our batch size :

$$\begin{aligned} b_f^* &= \max_i \left(\frac{X_f^* T_{si}}{1 - X_f^* T_{oi} n_{if}} \right) = \max \left(\frac{.4 \times 1}{1 - .4 \times 1 \times 1}; \frac{.4 \times 20}{1 - .4 \times 1 \times 1}; \frac{.4 \times 1}{1 - .4 \times 1 \times 1}; \frac{.4 \times 1}{1 - .4 \times 2 \times 1} \right) \\ &= \max \left(\frac{2}{3}, \frac{40}{3}; \frac{2}{3}; 2 \right) \approx 13.33 \end{aligned}$$

Using the other constraint (on the production time), we can then compute an upper bound for the batch size :

$$\bar{b}_f = \min_{\mathcal{P}_k} \left(\frac{\bar{T} - \sum_{i \in \mathcal{P}_k} T_{si}}{\sum_{i \in \mathcal{P}_k} T_{oi} n_{if}} \right) = \min \left(\frac{250 - 22}{3}; \frac{250 - 2}{3} \right) = 76$$

Which gives us the final solution :

$$b_f^* = 14 \leq b_f \leq 76 = \bar{b}_f$$

Of course, it is possible to write infeasible constraints which would lead to having $\bar{b}_f < b_f^*$. This situation is depicted in figure (4.11). If such a situation were to happen, the way of solving it is to decrease the setup time which means either (1) numerically decreasing the setup time (buying new machines for example) or by masking the setup time.

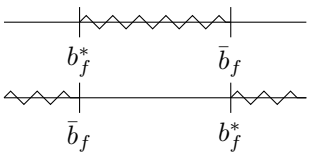


Figure 4.11: Batch size

Chapter 5

Arbitrary batch sizing ($b_i \neq b_f n_{if}$)

So far, we have built a model considering operation time and setup times to for each machine. Doing that, we made an assumption that for any i we would choose the batch production size of product i following the rule $b_i = b_f n_{if}$. We saw that (1) this is not mandatory (but simpler from the point of view of calculations) and that (2) it is not always feasible. In this chapter, we will discuss the case when $b_i \neq b_f n_{if}$. In the first chapter of this part, we established the following formula :

$$X_f^{max}(\underline{b}) = \min_i \left(\frac{\mu_i}{n_{n_{if}}} \right) = \min_i \left(\frac{b_i}{T_{si} + T_{oi} b_i} \Big/ n_{if} \right)$$

If we want to produce at a minimum production rate X_f^* we can now compute the minimum batch size on machine i like so

$$\begin{aligned} X_f^{max}(\underline{b}) \geq X_f^* &\Leftrightarrow \frac{b_i}{T_{si} + T_{oi} b_i} \geq X_f^* n_{if}, \forall i \\ &\Leftrightarrow b_i \geq X_f^* n_{if} T_{si} + X_f^* n_{if} T_{oi} b_i, \forall i \\ &\Leftrightarrow b_i \geq \frac{X_f^* n_{if} T_{si}}{1 - X_f^* n_{if} T_{oi}}, \forall i \end{aligned}$$

Futhermore, we know that the production rate for machine i can be computed as $X_i^* = X_f^* n_{if}$ and that the "production" cycle is given by $D_i^* = \frac{b_i}{X_i^*}$. However, how can we draw a GANT diagram of such a production plan ?

5.1 The Kanban organisation

In this section, we will try to compute the overall "production cycle". That is, from which point in time can we repeat the production of a batch of size b_i for machine i. Since every machine can now choose an arbitrary batch size, how can we plan our production ? Well, for it to be repeatable, the overall production cycle D has to be a multiple of each particular machine production cycle D_i^*. The minimum one being the *minimum common multiple* of each D_i of the machines. Thus, we have a lower bound for the production cycle which is

$$D^* = \operatorname*{mcm}_i (D_i^*)$$

Note that the *minimum common multiple* doesn't have to be an integer, for example : $\text{mcm}\left(\frac{1}{2}; \frac{1}{4}\right) = \frac{1}{2}$.

Given these numbers, we can easily input how many items of product i are produced during a whole period D^* since D^*/D_i^* (which is integer) equals the number of batches produced in the time period D^*, we get that

$$\#items(i) = \left(\frac{D^*}{D_i^*} \right) b_i$$

which can be written in a similar way by substituting D_i^* by its expresion in terms of b_i, n_{if} and X_f^* as previously introduced :

$$\#items(i) = \frac{D^*}{\frac{b_i}{X_f^* n_{if}}} b_i = (D^* X_f^*) n_{if} \tag{5.1}$$

Finally, since $n_{ff} = 1$, it holds that $\#items(f) = D^* X_f^*$, which makes relation (5.1) somehow analogous to our initial relation $b_i = b_f n_{if}$.

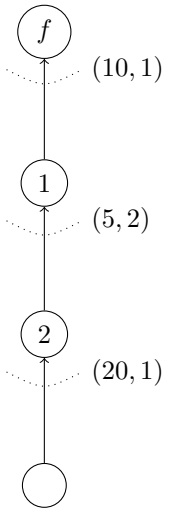


Figure 5.1: Example

But let's have an example in order to understand better how we can draw our gant. Consider figure (5.1) and a minimum production rate $X_f^* = \frac{1}{3}$. We can easily compute the minimum batch size resulting of that constraint using the previously cited formula like so :

$$b_f \geq \frac{\frac{1}{3}.1.10}{1 - \frac{1}{3}.1.1} = 5$$

$$b_1 \geq \frac{\frac{1}{3}.1.5}{1 - \frac{1}{3.1.2}} = 5$$

$$b_2 \geq \frac{\frac{1}{3}.1.20}{1 - \frac{1}{3}.1.1} = 10$$

These lower bounds tell us about the minimum batch size we have to use in order to produce at the minimum rate $X_f^* = \frac{1}{3}$. Thus, we can use greater batch sizes like the following : $b_f = 6$, $b_1 = 5$ and $b_2 = 10$. Using this choice, we can compute each machine's production cycle like so

$$D_f^* = \frac{6}{\frac{1}{3}.1} = 18 = 2.3^2 \; ; \; D_1^* = \frac{5}{\frac{1}{3}.1} = 15 = 3.5 \; ; \; D_2^* = \frac{10}{\frac{1}{3}} = 30 = 2.3.5$$

From which we can now deduce the minimum common multiple which is the minimum production cycle for the overall plan : $D^* = 2.3^2.5 = 90$. Assuming these results, you will find in figure (5.2) the GANT representation of two production cycles.

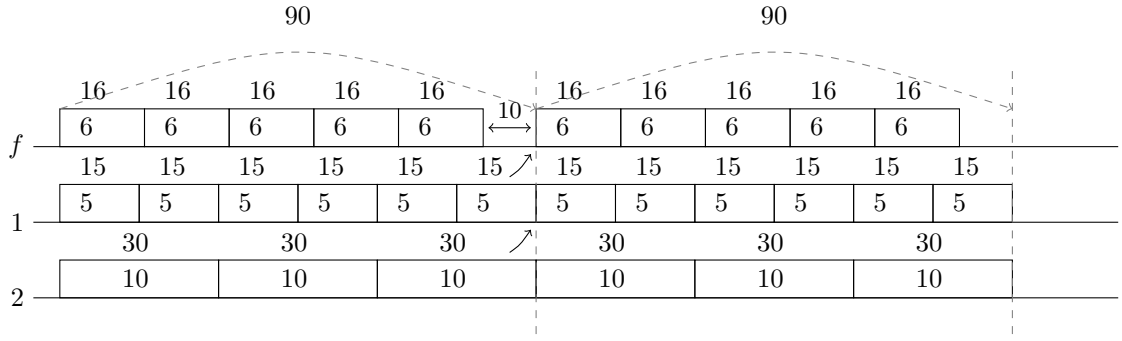


Figure 5.2: Resulting GANT diagram

However, as depicted in figure (5.2), one may have noticed that we did not fully respect the dependency of product 1 over product 2, nor of f over 1. In fact, it is not possible to draw such a GANT. To avoid this, we assume the existence of queues in front of each machine and that from now on, the objective for every machine is to fill the queue in front of it when it is emptied. This way of reasonning is called "pull production" since the information flow goes from right to left, whereas in the previous chapters we were referencing to "push production" since it was the raw material which determined when the following tasks could begin. In this view of the production line, every machine starts producing a batch as soon as the queue in front of him is empty. It is called as well KANBAN organisation.

5.2 Using Kanban with $b_i = b_f n_{if}$

Studying this new kind of systems, we have discovered a new pattern for production scheduling. In this section, we will have a look at what happens if we apply the KANBAN organisation to the situations in which $b_i = b_f n_{if}$. Of course, we have $X_f^* = b_f^*$ which gives us, for machine i, a production cycle of

$$D_i^* = \frac{b_f n_{if}}{X_f^* n_{if}} = \frac{b_f}{X_f^*}$$

which does not depend on i. Computing the overall production rate is trivial since the *minimum common multiple* of a set of identical numbers D_i^* is simply given by D_i^*. So that $D^* = D_i^*, \forall i$ since they are the same.

Using figure (5.1) again with the same production rate ($\frac{1}{3}$), we can compute b_f^* easily like before. This yields $b_f^* = 10$. Let's say we decide to use $b_f = 12$. Then we can compute the production cycle as $D^* = \frac{12}{\frac{1}{3}} = 36$ (and of course $D^* = D_i^*$) and we can draw, as in figure (5.3), the GANT diagram of such a plan.

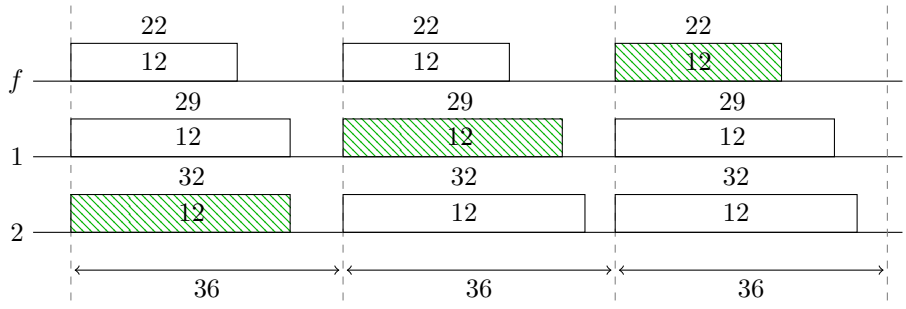


Figure 5.3: GANT diagram using batch size $b_f = 12$

If we assume to start the production with empty queues in front of each machine, what would be the time needed to produce the first batch ? Well, as represented by the green rectangles in figure (5.3), we would roughly need to wait as many production cycles as there are levels in the bill of materials (which means, product dependencies). Thus, the following formula

Production time of the first batch $\approx$ production cycle $\times$ number of levels of BoM

In our example, this formula gives us $3 \times 36 = 108$, when the real result is 94.

Chapter 6

Fullfillement of an order

Up to this chapter, our focus has been on the fullfillment of a constraint given in terms of production rate and production time. In this part of the course, we will change our main objective by adding a constraint on the number of items to be produced. Still we will have a minimum rate to be performed, stil we will have a production time limit not to be crossed, but the objective now becomes to produce exactely N final products and to plan the production accordingly.

6.1 Given that $b_i = b_f n_{if}$

6.1.1 Push systems

We know, from the previous chapters, that the production time can be computed as

$$T_{prod}(b_f) = \max_{\mathcal{P}_k} \left(\sum_{i \in \mathcal{P}_k} (T_{si} + b_f T_{oi} n_{if}) \right)$$

but it is worthy to notice that the time to seperately produce two batches of size b_f does not equal the double time we would need to produce one batch of size b_f. More formally, we are saying that $T_{prod}(2.b_f)|_{b_f} \neq 2T_{prod}(b_f)$ in the sense that figure (6.1) represents. In fact, we do not need for a whole "production cycle" to be finished to begin with another one. Indeed, we can begin a new cycle as soon as the "longest" machine has done its work (i.e. the bottleneck machine). The following formula holds instead :

$$T_{prod}(2.b_f)|_{b_f} = T_{prod}(b_f) + T_b(b_f)$$

where $T_{prod}(2.b_f)|_{b_f}$ represents the time needed to produce $2.b_f$ items knowing that the batch size is b_f and where $T_b(b_f)$ is the time which is needed, by the bottleneck machine (hence T_b for bottleneck) to produce a bacth of size b_f. This formula can, for sure, be generalized as

$$T_{prod}(K.b_f)|_{b_f} = T_{prod}(b_f) + (K-1)T_b(b_f) \tag{6.1}$$

A first example

However, the above formula holds for a given bottleneck machine. But we do know that the bottleneck machine depends on the batch size which has been chosen *a priori* : given a batch size b_f, we can comptue the production

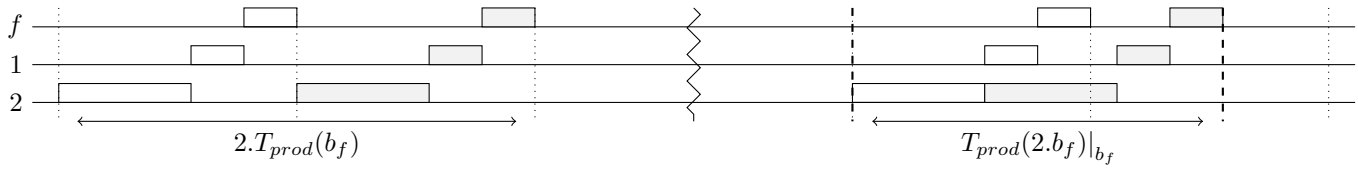


Figure 6.1:

time $T_{prod}(K.b_f)|_{b_f}$. But how can we choose b_f with constraints on the production rate, on the overall production time, and, as well, on the number of items to be produced ? This is not an easy question actually, so let's take an example in order to better understand where the difficulty lies.

Let's consider the Bill of Material depicted in figure (6.2) with the following constraints :

1. Minimum production rate of $X_f^* = \frac{1}{5}$
2. Maximum production time for a batch $\bar{T} = 80$
3. Number of items to be produced $N = 200$

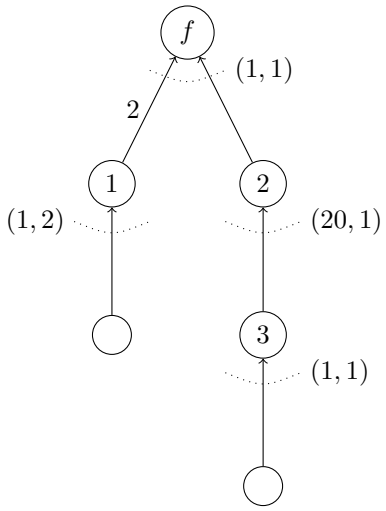


Figure 6.2: Example BoM

We are very used to the first two constraints. The first one gives us a lower bound for the batch size in order for the production rate to be feasible. The second one gives us an upper bound so that the production time of a batch is not greater than the time limit. Computing these bounds is rather easy and we will not discuss them :

$$
\begin{aligned}
b_f^* &= \max_i \left(\frac{X_f^* T_{si}}{1 - X_f^* T_{oi} n_{if}} \right) \\
&= \max \left(\frac{\frac{1}{5}.1}{1 - \frac{1}{5}.1}; \frac{\frac{1}{5}.1}{1 - \frac{1}{5}.2.2}; \frac{\frac{1}{5}.20}{1 - \frac{1}{5}.1.1}; \frac{\frac{1}{5}.1}{1 - \frac{1}{5}.1.1} \right) \\
&= 5 \\
\bar{b}_f &= \min_{\mathcal{P}_k} \left(\frac{\bar{T} - \sum_{i \in \mathcal{P}_k} T_{si}}{\sum_{i \in \mathcal{P}_k} T_{oi} n_{if}} \right) \\
&= \min \left(\frac{80 - (1+1)}{2.2 + 1.1}; \frac{80 - (1 + 20 + 1)}{1.1 + 1.1 + 1.1}; \right) \\
&= 15.6
\end{aligned}
$$

Which tells us that we need to choose our batch size so that $b_f \in [5, 15]$. Since we want to produce $N = 200$ items, we should repeat a certain number of times the production of such bacthes untill we reach 200 items. On possible way would be to produce 40 times a batch of 5 items (since $40 \times 5 = 200$), or produce 25 times batches of 8 (since again $25 \times 8 = 200$). In fact, the only multiples of 200 witin the interval $[5, 15]$ are $\{5, 8, 10\}$. Let's study each one of these cases !

For each cases, we can write the equation (6.1) as

$$
\begin{aligned}
T_{prod}(200)|_{bf=5} &= T_{prod}(5) + 39.T_b(5) \\
T_{prod}(200)|_{bf=8} &= T_{prod}(8) + 39.T_b(10) \\
T_{prod}(200)|_{bf=10} &= T_{prod}(8) + 39.T_b(10)
\end{aligned}
$$

where $T_{prod}(b_f)$ can be computed as

$$T_{prod}(b_f) = \max_{\mathcal{P}_k} \left(\sum_{i \in \mathcal{P}_k} (T_{si} + T_{oi} n_{if}) \right) = \max(2 + 5.b_f; 22 + 3.b_f)$$

and

$$T_b(b_f) = \max(1 + 1.b_f; 1 + 2.2.b_f; 20 + 1.b_f; 1 + 1.b_f) = \max(1 + 4b_f; 20 + b_f)$$

Using these formulas, we easily get the following results :

b_f	$T_{prod}(b_f)$	$T_b(b_f)$	$T_{prod}(200)\|_{b_f}$
5	37	25	1012
8	46	33	838
10	52	41	831

Which means that, for this way of producing 200 items, using a batch size of 10 items is the quicker (it turns out to be the biggest batch size but this is just a particular case). However, there are other ways of getting to 200 ! For example, we may want to produce 13 batches of 15 and a batch of 5, since $200 = 13 \times 15 + 5$. Looking at figure (6.3), we realise that we can compute the time needed to produce 200 items in that way by first computing the time to produce 13 bacthes of 15 and then add the difference between the needed time to produce a batch of 5 and the time at which the bottleneck machine can start.

We get that :

$$\begin{aligned} T_{prod}(15) &= 77 \\ T_b(15) &= 61 \\ T_{prod}(195)|_{bf=15} &= 77 + 12.61 = 809 \\ T_{prod}(5) &= 37 \\ T_b(5) &= 25 \end{aligned}$$

And, it holds that the delay represented by the red arrow in figure (6.3) can be computed as the production time of the bottleneck machine $T_b(5)$ minus the production time of machine f, which yields $25 - 16 = 9$. We can now compute the final, overall, production time which is : $809 + 9 + 6 = 824$! Which is less than what we had previously found.

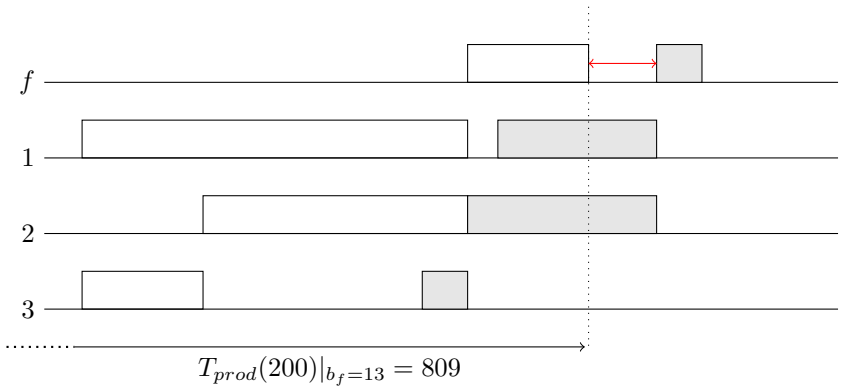


Figure 6.3: GANT diagram corresponding to the production of $200 = 13 \times 15 + 5$ items

Finding the optimal batch size is not easy to be done by hand and we usually use approximation techniques to find a solution.

Approximate solution

Let us define our coefficient K as $\frac{N}{b_f}$ and let's assume that this number is an integer number. Then, assuming that we know *a priori* the bottleneck machine b, it holds that

$$\begin{aligned} T_{prod}(N)|_{b_f} &= \sum_{i\in\mathcal{P}_k}(T_{si}+T_{oi}n_{if}b_f)+\left(\frac{N}{b_f}-1\right)(T_{sb}+b_fT_{ob}n_{bf}) \\ &= \sum_{i\in\mathcal{P}_k}T_{si}+b_f\left(\sum_{i\in\mathcal{P}_k}T_{oi}n_{if}-T_{ob}n_{bf}\right)+NT_{ob}n_{bf}+\frac{N}{b_f}T_{sb} \end{aligned}$$

which is just a re-writing of the previously introduced equations. This function of b_f is convex, we can find its global minimum by first relaxing the "integer" constraint and derivating this function and solving $\nabla. = 0$. We get

$$\frac{\partial.}{\partial b_f} = \sum_{i\in\mathcal{P}_k}T_{oi}n_{if}-T_{ob}n_{bf}-\frac{N}{b_f^2}T_{sb}$$

and

$$\nabla. = 0 \Leftrightarrow b_f^\circ = \sqrt{\frac{T_{sb}}{\sum_{i\in\mathcal{P}_k}T_{oi}n_{if}-T_{ob}n_{bf}}}$$

This results gives us the optimal batch size we should use **given the bottleneck machine considered**. However, we know that the bottleneck changes with the batch size. In fact, imagine a situation in which the bottleneck machine changes for a given batch size, say $\tilde{b}_f$. And imagine a situation in which, for $b_f > \tilde{b}_f$, we have a convex function like the red one in figure (6.4), and that, on the other hand, for $b_f < \tilde{b}_f$, we have a convex function like the black one in the same figure. Using that formula will yield one of the two extremum depending on which machine we assume to be the bottleneck machine...

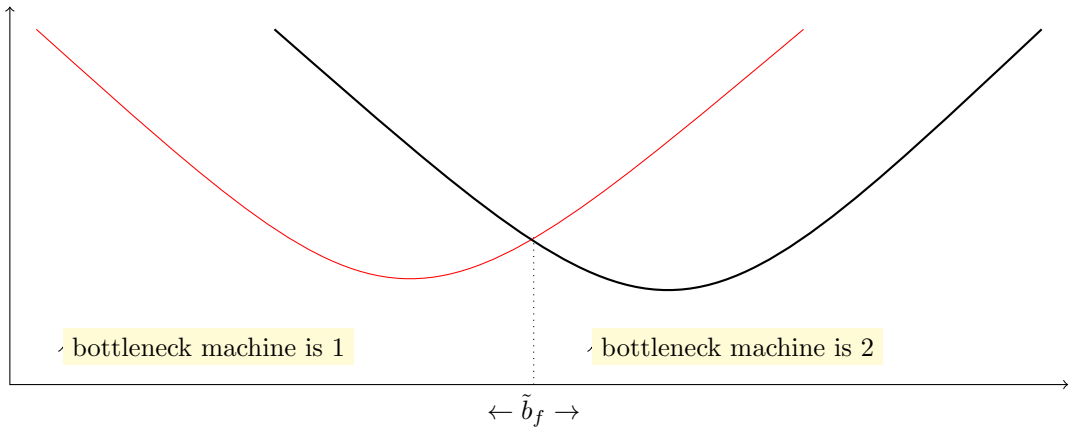


Figure 6.4: Two different convex functions for finding "optimal" batch size

Upper bound

As previously said, computing the exact production time needed to produce N items isn't an easy thing to do. However, a good upper bound for the production time, given a batch size b_f, can be computed as

$$T_{prod}(N)|_{b_f} \leq T_{prod}(b_f)+\left(\left\lceil\frac{N}{b_f}\right\rceil-1\right)T_b(b_f)$$

The error we're making in our estimate is then proportionnal to $\left\lceil\frac{N}{b_f}\right\rceil b_f - N$ where $\left\lceil\frac{N}{b_f}\right\rceil b_f$ represents the number of items for which we computed the production time and N the aimed number of items.

6.1.2 Pull systems

Concerning production systems which use the Kanban approach (also refered as pull systems), we can find a somehow analogous formula to compute an upper bound for the production time of N items.

From the previous sections, we know that the ratio D/D_i gives us the number of batches that machine i delivers during a time window D. Thus, the working time of machine i can be computed as

$$T_{wi} = \frac{D}{D_i}(T_{si} + T_{oi}b_i)$$

Which equals, if we consider the batch sizing as $b_i = b_f n_{if}$,

$$\frac{D}{\frac{b_f}{X_f^{max}(b_f)}}(T_{si} + T_{oi}n_{if}) = D\frac{X_f^{max}(b_f)}{\frac{b_f}{T_{si}+T_{oi}n_{if}b_f}}$$

since $D_i = D, \forall i$ and $D = b_f/X_f^{max}(b_f)$. Finally, noting that $X_f^{max} = \min\left(\frac{b_f}{T_{si}+T_{oi}b_f n_{if}}\right)$, we conclude that in any case :

$$T_{wi} \leq D$$

Thus, it stands that, since the time to produce the first batch is given by $L.D$ where L is the level of the Bill of Material, an upper bound for the production time of N items using batch sized as $b_i = b_f n_{if}$ in the pull production system is :

$$T_{prod}^{pull}(N)|_{b_f} = LD + \left\lceil \frac{N}{b_f} - 1 \right\rceil D$$

This is very analogous to the previous formula obtained when dealing with push systems since here, the production cycle D is determined by the bottleneck machine ($D = T_b(b_f)$).

6.1.3 Conclusion

We end up with two estimates for an upper bound of the time to produce N items which are :

$$T_{prod}^{push}(N)|_{b_f} \leq T_{prod}(b_f) + \left(\left\lceil \frac{N}{b_f} \right\rceil - 1\right) T_b(b_f)$$

$$T_{prod}^{pull}(N)|_{b_f} \leq LD + \left(\left\lceil \frac{N}{b_f} \right\rceil - 1\right) T_b(b_f)$$

And it can be easily seen that $T_{prod}^{push}(b_f) \leq LD$ which imples the following relation between our two upper bounds :

$$T_{prod}^{push}(N)|_{b_f} \leq T_{prod}^{pull}(N)|_{b_f}$$

But since our goal is to find a "reasonable" time to be considered for the production of N items, for instance to tell our consumer that the production will be ready by that time and that he should come pick up our final products, one may always use the second formula even for *push* systems since (1) it is an upper bound of the production time and (2) we end with a uniform formula.

6.1.4 Computing upper bounds : one final example

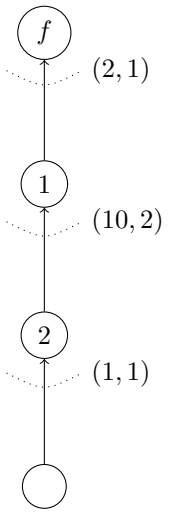


Figure 6.5: Example

Consider figure (6.5) which represents a very simple bill of material. Let's assume that one client orders $N = 70$ final products. When should we tell him to come back to pick up his goods ? We will do the computations studying various batch sizes among $10, 7$ and 6.

Using push systems

Using the previously established formula, we can compute an upper bound of the time needed to produce $N = 70$ items like so :

$$T^{push}_{prod}(70)|_{b_f=10} = (\underline{1 + 1.10} + \underline{10 + 2.10} + \underline{2 + 1.10}) + (7 - 1)(10 + 2.10) = 233$$

since with $b_f = 10$, machine number 1 is the bottleneck machine. Note that, in this situation, the computed upper bound is actually the "supremum" (the best upper bound we can found) which coincides with the exact value of T^{push}_{prod} since 70 is a multiple of 10.

Similar computations gives the following results :

$$T^{push}_{prod}(70)|_{b_f=7} = 257$$
$$T^{push}_{prod}(70)|_{b_f=6} \leq 279$$

And note that, in the latter case, the result is an upper bound which differs from the exact value of the production time needed since 6 is not a divisor of 70. Actually, we are estimating our upper bound by computing the time needed to produce $6 \times 12 = 72$ items instead of 70 since $\lceil 70/6 \rceil = 12$.

Using pull systems

Using the corresponding formula for the pull systems, we can compute upper bounds in the same way. (Note that the formula can be rewritten as $T^{pull}_{prod}(N)_{b_f} = \left(L - 1 + \left\lceil \frac{N}{b_f} \right\rceil\right) T_b(b_f)$ which is handier to compute by hand...). Doing so, we get the following results :

$$T^{pull}_{prod}(70)|_{b_f=10} \leq (3 - 1 + 7).30 = 270$$
$$T^{pull}_{prod}(70)|_{b_f=7} \leq 288$$
$$T^{pull}_{prod}(70)|_{b_f=6} \leq 308$$

Which gives as expected greater upper bounds but sufficiently good ones.

Optimal batch sizing ?

Just like we did in the section dedicated to push systems, and since we know that we can use one uniformed formula, let's suppose N/b_f to be integer and let's assume knowing the bottleneck machine corresponding to the optimal

batch size, our upper bound can be given by

$$\begin{aligned} T_{prod}(N)_{b_f} &= \left(L - 1\frac{N}{b_f}\right) T_b(b_f) \\ &= \left(L - 1\frac{N}{b_f}\right) (T_{sb} + T_{ob} b_f n_{bf}) \\ &= \frac{N}{b_f} T_{sb} + (L-1) T_{ob} b_f n_{b_f} + (L-1) T_{sb} + N T_{ob} n_{bf} \end{aligned}$$

And let's take the derivative of this function with respect to b_f, it yields

$$\frac{dT_{prod}}{db_f} = -\frac{N}{b_f^2} + (L-1) T_{ob} b_f n_{bf}$$

Which is convex and yields the optimal solution

$$b_f^\circ = \sqrt{\frac{N T_{sb}}{(L-1) T_{ob} n_{bf}}}$$

But keep in mind that, similarly to what have been said in the previous section, this formula holds only if the bottleneck machine was well the one we had supposed it would be. That is, we have to check *a posteriori* that the bottleneck machine we considered for our computation is well the one corresponding to the batch size b_f°.

Looking back at our example, we obtain the "optimal" batch size (supposing that 1 will be the bottleneck machine) as $b_f^\circ = \sqrt{\frac{70.10}{2.2}} \approx 14$. Using such a batch size[1] will result in the "smallest" production time estimate using the established formula which are

$$\begin{aligned} T_{prod}^{push}(70)|_{b_f=14} &= 221 \\ T_{prod}^{push}(70)|_{b_f=14} &= 266 \end{aligned}$$

6.2 Arbitrary batch sizing

Our final consideration concerning the computation of the time needed to produce a fixed number of items N is the one related to the situation in which $b_i \neq b_f n_{if}$. We have seen that the only way of producing using an arbitrary batch size is to use the Kanban system. In this case, we are dealing with different batch sizes for each machine b_i which are arbitrary choosen. We have already established that $\frac{D}{D_i} = K_i$ gives us the number of batches of product i produced during the time window D. Thus, $K_i b_i$ returns the number of items i produced during one production cycle D and $\left\lceil \frac{N}{K_f b_f} \right\rceil$ equals the number of production cycle D needed in order to produce N items. Thus, the following formula holds:

$$T_{prod}(N)|_{\underline{b}} = LD + \left(\left\lceil \frac{N}{K_f b_f} \right\rceil - 1 \right) D$$

Which is a generalization of the previous formula when $b_i = b_f n_{if}$ since in this case we have $K_f = 1$.

[1] which is relevant since the bottleneck machine associated to that batch size is well machine 1

Chapter 7

Production scheduling

In the previous part, we dealt with the optimization of a production plant with respect to the constraints of the bill of material in order to produce and fulfill orders from our customers. If we take a step backward however, we can regard our production plant as a sort of "black box" which turns orders into available final products. By considering only upper bounds for the production time for each order, we can study how and when to produce items without taking into accounts the details of the bill of material. This new part of the course will be dedicated to the study of production scheduling problems. The first part of this chapter will present the basic concepts of scheduling while the second part will introduce very basic rules to optimize certain objective functions which we will discuss.

7.1 General scheduling problems

Scheduling problems are a combination of three decisions to be made at the same time, which are :

Assignment : associating each and every task to one resource (some resources can remain unused). $\forall$ task τ_i, associate one resource R_j : $\tau_i \mapsto (\tau_i, R_j)$

Sequencing : deciding a permutation among the set of tasks to be performed in that specific order. $T = \{\tau_1, ..., \tau_N\} \mapsto (\tau_1, \tau_5, \tau_3, ..., \tau_2)$. $\forall (\tau_i, \tau_j) \in T, t_{\tau_i} < t_{\tau_j}$ or $t_{\tau_j} < t_{\tau_i}$.

Time tabling : choosing explicitely the time at which each task should begin its processing.

Clearly, an optimal solution for the scheduling problem has to simultaneously take into account these three characteristics. However, heuristic approaches often build up a solution incrementaly taking these "steps" seperately.

In our case, the number of resources (number of machies) is just one since the only "resource" we have is the abstracted production plant.

When dealing with scheduling problem, we often introduce variables representing the completion time C_i of one task τ_i which is the moment in time at which the task has been performed. That moment come, if possible, before a given due date dd_i. To characterize the fact that a task is performed too late with respect to the due date, we introduce the "lateness" function as $L_i = C_i - dd_i$ which is negative if the task is early, 0 if the task is performed just in time and positive if the task is late. Sometimes, we also consider "tardiness" which is analogous to lateness except that early tasks are valued with 0, it can be expressed as $T_i = \max(0, C_i - dd_i)$. Finally, we may consider as well the "tardy job" indicator U_i which equals 1 if the job τ_i is tardy and 0 otherwise.

In general, scheduling is an attempt to minimize an objective function taking into account every possible sequencing and every possible timetabling. We say that a cost function J is "regular" if and only if $\frac{\partial J}{\partial C_i} \geq 0, \forall i$. In other words, the value of J decreases when each C_i decreases : the sooner the better. Thus, for that kind of cost functions, only the sequencing of the task is necessary since, following the "the sooner the better" rule, timetabling is performed by following the established sequencing without idle time between the task.

7.2 Basic rules for scheduling

In this section, we will introduce very basic rules to solve some of the simplest scheduling problems. The examples given will all refer to figure (7.1) which shows a set of tasks with both their processing time and due dates.

	τ_1	τ_2	τ_3	τ_4	τ_5	τ_6	τ_7
t_i	5	10	7	6	8	10	5
dd_i	20	100	50	40	30	35	55

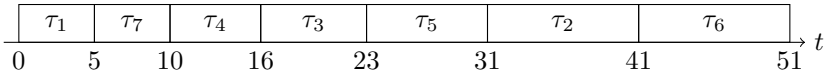
Figure 7.1: A set of tasks with their processing times t_i and due dates dd_i

7.2.1 Shortest Processing Time first (SPT)

The SPT rule consists of ordering the tasks by their processing time to define the sequencing of tasks. This rule is optimal for any regular cost function. For instance, the following cost function :

$$\text{minimize } J = \frac{1}{N}\sum_{i=1}^{N} C_i$$

where N is the number of tasks is a regular function and the corresponding solution is represented in figure (7.2).



$$J = \tfrac{1}{7}(5 + 10 + 16 + 23 + 31 + 41 + 51) = \tfrac{177}{7} \approx 25$$

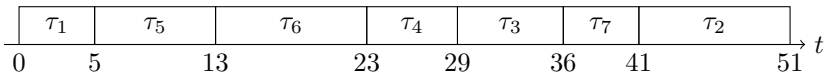
Figure 7.2: Solution using the SPT rule for minimizing $J = \frac{1}{N}\sum_i C_i$

7.2.2 Earliest Due-Date first (EDD)

The earliest due date rule is used to minimize the maximum lateness defined as

$$\text{minimize } J = \max_i L_i$$

Note that this does not necessary means that one task will be late, it simply says that even the task with the maximum lateness (which can be negative in case of an early task) should be put as small as possible (which means, in case of an early task, as soon as possible). The solution of this problem with the data from figure (7.1) is represented in figure (7.3). In this case, you will notice that no tasks are late with respect to their due date and that the EDD does not allow idle time between the tasks.



$$J = \max(5-20, 13-30, 23-35, 29-40, 36-50, 41-55, 51-100) = -11$$

Figure 7.3: Solution using the EDD rule for minimizing $J = \max L_i$

7.2.3 Moore algorithm

The Moore algorithm is used to minimize the number of tardy jobs ($\sum_i U_i$) and is formalized as follows :

step 0 : Initialize $\mathcal{L}$ using the EDD rule and $\mathcal{R} = \emptyset$

step 1 : Compute the lateness $L_i, \forall \tau_i \in \mathcal{L}$

step 2 : If there exists a tardy task in $\mathcal{L}$, focus on the subset $\mathcal{L}' = \{\tau^1, ..., \tau^k\}$ where k is the number of the first tardy task found. Else, **STOP**, the solution is optimal.

step 3 : Extract from $\mathcal{L}'$ the task with the greatest processing time from $\mathcal{L}$ and move it to $\mathcal{R}$. Go to **step 1**.

Let's execute the algorithm with the following input tasks :

	τ_1	τ_2	τ_3	τ_4	τ_5	τ_6	τ_7	τ_8
t_1	5	7	10	9	10	8	15	3
dd_i	30	45	20	15	32	50	80	22

The first step is to initialize the $\mathcal{L}$ list with the tasks ordered following the EDD rule. Such an operation yields the following :

$$\mathcal{L} = (\tau_4, \tau_3, \tau_8, \tau_1, \tau_5, \tau_2, \tau_6, \tau_7)$$

which is represented in figure (7.4). We can then compute the lateness of each task $L_i = C_i - dd_i$ as

$$L = (-6, -1, 0, -3, 5, -1, 2, -13)$$

in which we can see that both τ_5 and τ_6 are tardy.

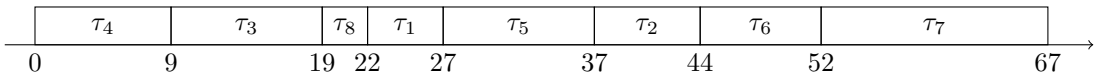


Figure 7.4: Ordered tasks by the EDD rule

Step 2 of the algorithm tells us to focus on the sublist $\mathcal{L}' = (\tau_4, \tau_3, \tau_8, \tau_1, \tau_5)$ composed of all the tasks scheduled before the first tardy task. Among these tasks, the one with the gratest processing time is τ_3 ($t_3 = 10$), thus, we move τ_3 from $\mathcal{L}$ to $\mathcal{R}$. We then go back to step 1.

At this point, we have removed τ_3 from $\mathcal{L}$ so that now

$$\mathcal{L} = (\tau_4, \tau_8, \tau_1, \tau_5, \tau_2, \tau_6, \tau_7)$$

on which we can compute the lateness of each tasks :

$$L = (-6, -10, -13, -15, -11, -8, -23)$$

We can see, as a result, that no tasks are tardy. The algorithm stops and the solution is given by $(\mathcal{L}|\mathcal{R})$ where $\mathcal{R}$ contains the only task which will be tardy.

7.3 Choosing the right rule

Since different approaches have been developed in the former section, one may wonder how and when to choose one technique rather than another. The answer to that specific question is, of course, dependent of the context. However, some words can be said regarding to the cost constraints and how they may influence one's choice.

For instance, let's consider a case where the contract on which you agreed with the customer stipulates that the cost curve is of the same shape as figure (7.5). Such a curve says that the cost associated to being late is small if one order is delivered tardy but gets more and more elevated as the lateness increases. In such a case, the EDD rule is the right approaches since we "don't care" about being late as long as we are not "too late".

Considering figure (7.6) however, we can see that we pay a fixed cost for being tardy. In this case, one should try to minimize the number of tardy jobs and should use the Moore's algorithm.

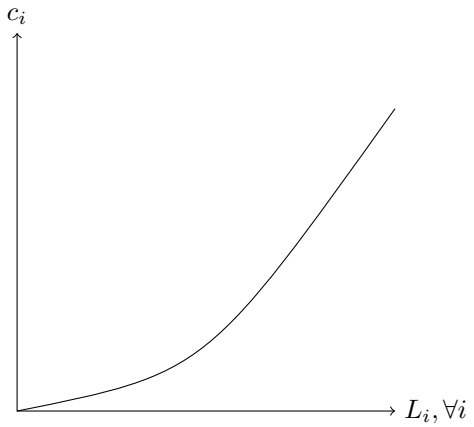


Figure 7.5: Cost function likely to be used with the EDD rule

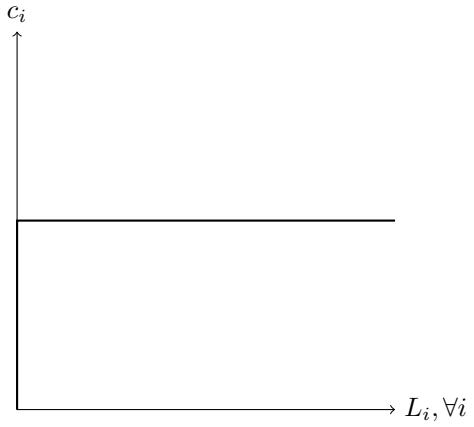


Figure 7.6: Cost function likely to be used with the Moore algorithm

Chapter 8

Shared temporary products

Up to this point, we have always made the assumption that one product (either raw material or temporary product) can only be used to build one, and only one, other product. In terms of BoM, this meant that every node (product) had at most one successor. In this new chapter, we will drop this assumption and examine how we can deal with this situation. From now on, we allow one node to have more than one successor.

8.1 Adapting the n_{if} coefficient

Let's consider figure (8.1) in which we have a product i required to make three other temporary products. Up to now, we always considered that there was one path from i to the final product f so that

$$n_{if} = \prod_{(k,j)\in Path_{i\to f}} n_{kj}$$

However, now, the notiton of $Path_{i\to f}$ becomes ambiguous.

But we can still reason about the bill of material and notice that the number of items i needed to produce one item f must take into account the number of items i needed to produce the three temporary produce needed for f. Thus, the number of items i needed is the sum between all the paths from i to f available in the bill of material. This property can be thought as a "superposition effect" typical in phisical systems. Finally, the following formula holds:

$$n_{if} = \sum_{p\in\mathcal{P}(i,f)} n_{ij}^{p}$$

where $\mathcal{P}(i,f)$ denotes the set of paths going from i to f and $n_{if}^{p} = \prod_{(k,j)\in p} n_{kj}$.

Concerning figure (8.1), we can see that there are three different paths from i to f. Thus, it holds that

$$\begin{cases} n_{if}^{red} & = 1.1 = 1 \\ n_{if}^{blue} & = 1.2.1 = 2 \Rightarrow n_{if} = 2+1+9 = 12 \\ n_{if}^{green} & = 1.3.3 = 9 \end{cases}$$

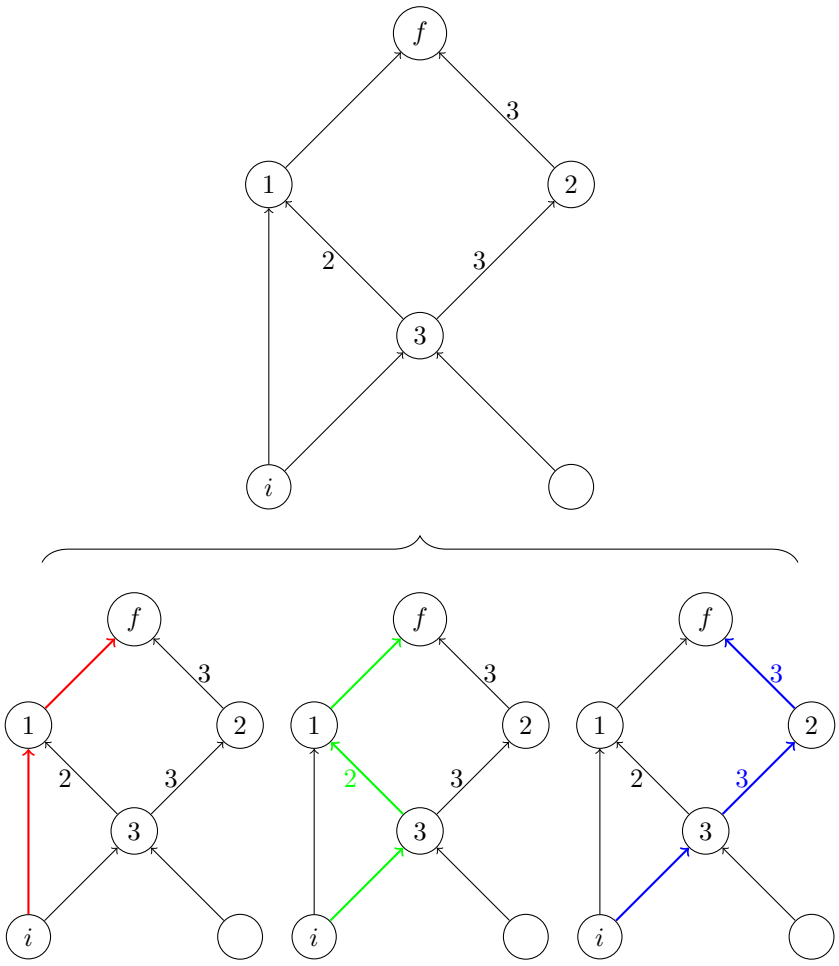


Figure 8.1: Dropping the unique path assumption

8.2 A fully worked example

In this section, we will treat a whole example using this new assumption. The example considered is associated to the bill of material represented in figure (8.2).

Let's first begin by computing the different $n_{if}, \forall i$ taking into account every possible paths. Remember that when no n_{ij} coefficient is written in the bill of material, it implies $n_{ij} = 1$. Hence the following results which can be found easily :

$$n_{ff} = 1 \qquad n_{1f} = 1 \qquad n_{2f} = 1 \qquad n_{3f} = 1$$
$$n_{4f} = 1 + 1 = 2 \qquad n_{5f} = 1 + 1 + 1 = 3 \qquad n_{6f} = 1 + 1 = 2 \qquad n_{7f} = 1 + 1 = 2$$

As usual, the maximum production rate is given by the following

$$\begin{aligned} X_f^{max}(b_f) &= \min_i \left(\frac{b_f}{T_{si} + T_{oi} b_f n_{if}} \right) \\ &= \min \left(\frac{b_f}{1 + b_f}; \frac{b_f}{4 + 2.b_f}; \frac{b_f}{2 + 2.b_f}; \frac{b_f}{1 + b_f}; \frac{b_f}{2 + 5.2.b_f}; \frac{b_f}{3 + 3.3.b_f}; \frac{b_f}{1 + 1.2.b_f}; \frac{b_f}{1 + 2.b_f} \right) \\ &\overset{b_f \to \infty}{=} \frac{1}{10} \end{aligned}$$

Let's assume to produce at a rate $X_f^* = \frac{1}{11}$, what is the minimum batch size we can use ? Again, as usual, we

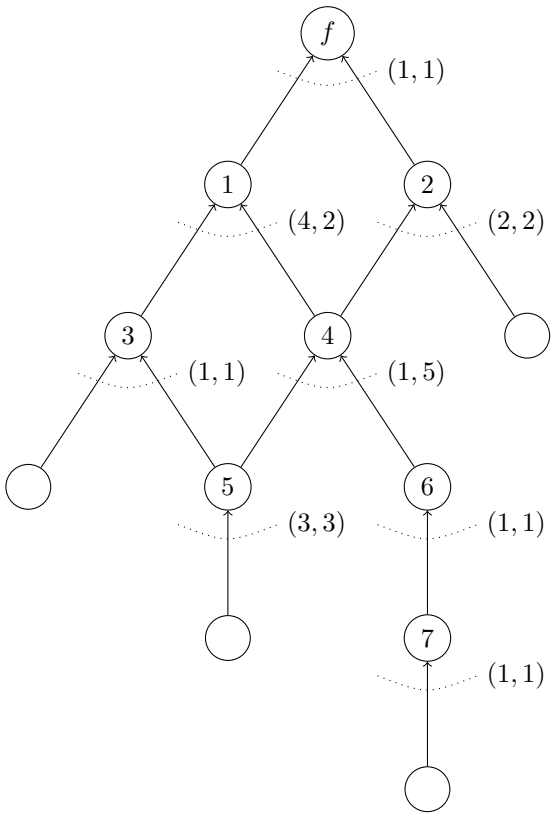


Figure 8.2: Bill of material for the worked example

compute it as the following

$$
\begin{aligned}
b_f^* &= \max_i \left(\frac{X_f^* T_{si}}{1 - X_f^* T_{oi} n_{if}} \right) \\
&= \max \left(\frac{\frac{1}{11}.1}{1 - \frac{1}{11}.1}; \frac{\frac{1}{11}.4}{1 - \frac{1}{11}.2.1}; \frac{\frac{1}{11}.2}{1 - \frac{1}{11}.2.1}; \frac{\frac{1}{11}.1}{1 - \frac{1}{11}.1.1}; \frac{\frac{1}{11}.2}{1 - \frac{1}{11}.5.2}; \frac{\frac{1}{11}.3}{1 - \frac{1}{11}.3.3}; \frac{\frac{1}{11}.1}{1 - \frac{1}{11}.1.2}; \frac{\frac{1}{11}.1}{1 - \frac{1}{11}.1.2} \right) \\
&= 2
\end{aligned}
$$

Which means that producing at rate $\frac{1}{11}$ implies using a batch of at least 2 items f. If we use such a batch size, the bottleneck machine would be machine 4. Let's try to depict the GANT diagram.

The way of drawing the GANT diagram is the following :

1. Draw a repeatable GANT diagram taking into account the production time of each batch
2. Identify the "schedule" for producing the first batch
   (a) Color the "boxes" associated to the machines without predecessors in the bill of material (these machines are the first (and only) machines to work during the first time window)
   (b) In the subsequent time window, color the "boxes" associated to successors of machines for which associated box is already colored in the previous time window.

Note that, still like before, it takes $L.D$ (where L is the number of levels of bill of material) to produce the first batch. In our case, $D = b_f / X_f = 22$. The associated GANT diagram is represented in figure (8.3).

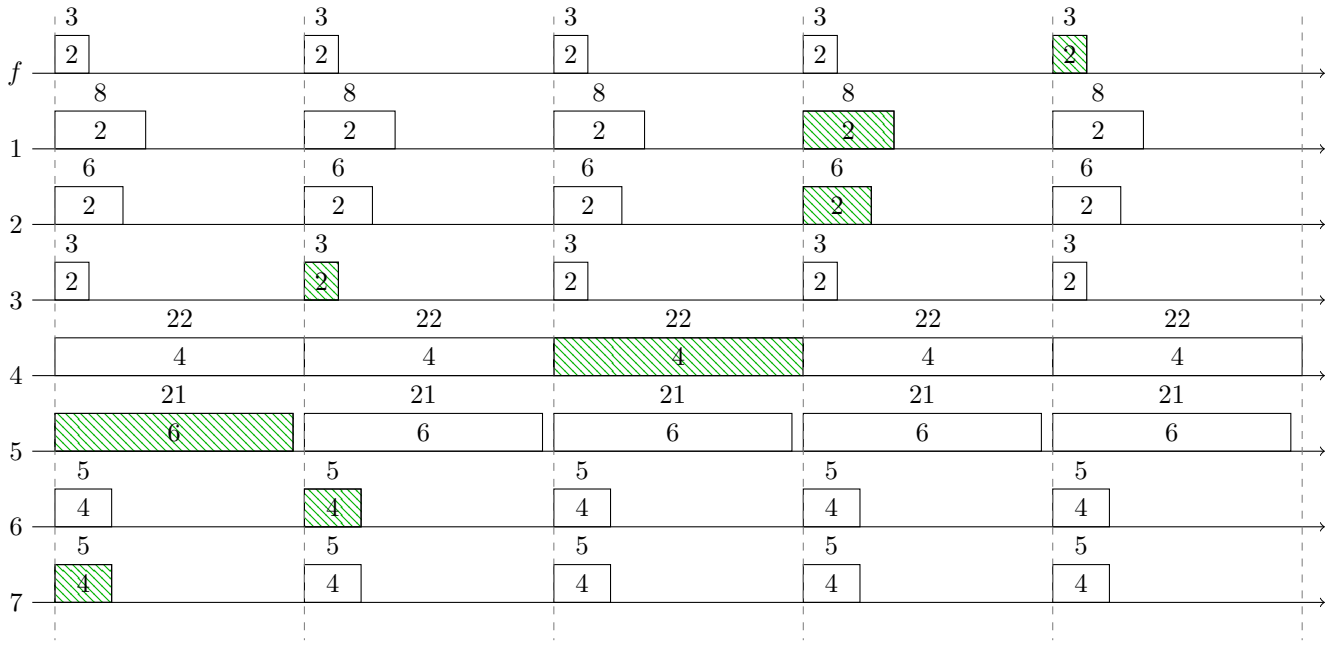


Figure 8.3: GANT diagram

Chapter 9

Multiple final products

Chapter 10

Shared resources generalisation