Design of a period batch control planning system for cellular manufacturing

Riezebos, J.

Chapter 5 Models and methods for determining a period length P

Chapter Four has shown that both stage definition and period length determination are essential factors in the design of a period batch control system. These factors should be determined with care. However, literature gives little support for an adequate decision about these factors. We have presented a framework that provides some assistance in designing stages. With respect to period length determination, we concluded that there is a lack of understanding and insight on the effect of period length variation in a period batch control system. For our study, we need to gain more insight in this design factor. We will start to fill this gap in the current chapter.

In order to improve our understanding of the effect of period length variation, we will first develop mathematical models for period length determination. These models will also include the effect of several measures that are enabled by a change in the production structure, as discussed in Chapter Four. One of these measures is overlapping production, which splits a batch into several subbatches. Overlapping production is easier applicable in a cellular manufacturing system compared with a functional organized system. We will discuss the effect of several strategies to determine these subbatches on period length and costs. The effect of varying the period length and applying various subbatch strategies on system performance will be addressed in Chapters Six and Seven.

We describe two deterministic approaches for period length determination. Section § 5.1 presents an approach that uses detailed information on operating characteristics of the products. It assumes that the allocation of operations to the stages is known in advance. Next, Section § 5.2 discusses classical economic period determination methods.

Section § 5.3 relaxes the assumption that the contents of the stages are known before the period length is being determined. It combines the two approaches that we discussed. The result is a detailed economic period determination approach for an equal number of subbatches strategy, which is presented and discussed.

Section § 5.4 relaxes the assumption that each operation should apply the same number of subbatches. It models the situation with different numbers of subbatches per operation and distinguishes between nested and non-nested batching policies. We present the required modifications in the model of Section § 5.3 for determining a suitable period length that allows various numbers of subbatches per operation.

Section § 5.5 describes solution methods for this model. It explores the cost structure of the model, and develops three related search heuristics: an enumerative search heuristic that finds a suitable variable batching strategy given a period length P, a progressive search heuristic that finds both a period length P and a suitable variable batching strategy, and finally an exhaustive search heuristic that aims at improving the solutions of the other two heuristics.

Section § 5.6 describes the performance of these three heuristics. Finally, Section § 5.7 concludes this chapter on mathematical models and solution methods for the determination of a period length P.

§ 5.1 Detailed decoupled period determination

This section develops a deterministic mathematical model for period length determination according to the principles of Burbidge that we presented in Section § 4.2.1. These principles assume that the period length has to be determined, given a stage definition. Then, the number of stages N and the contents of the stages are known in advance.

The general restriction on P is described with respect to the throughput time of the products that have to be produced within a stage (e.g., Burbidge, 1975a, New, 1977). The length of the period has to be such that the demand for all products can be fulfilled and hence that all required operations in the stage can be finished within one period. We call this a longest-path oriented lowerbound. We present it in Formula (1), but first we introduce some notation:

Longest-path oriented lowerbound on period length P

Define:

j : index of stage, $j = 1..\mathrm{N}$

h : index of product, $h = 1..\mathrm{H}$

i : index of operation, $i = 1..n_j^h$

n_j^h : number of operations of product h performed successively in stage j

p_{hi} : processing time of i^{th} operation of product h

m_{hi} : number of machines that perform the i^{th} operation of product h

q_h : batch size of product h

s_{hi} : setup time required for i^{th} operation of product h

P : Period length

$\lceil x \rceil^+$: nearest integer greater than or equal to x

THEN:

$$\mathrm{P} \geq \sum_{\mathrm{i}=1}^{n_j^h} \left\{ s_{hi} + p_{hi} \cdot \left\lceil \frac{q_h}{m_{hi}} \right\rceil^+ \right\} \quad \forall \mathrm{j} = 1..\mathrm{N}, \mathrm{h} = 1..\mathrm{H} \qquad (1)$$

We can apply this formula for all sequences of operations that are performed within a stage, i.e. for each path of operations. However, for sake of simplicity we will focus our attention on a single path, the longest path or critical path of operations for a product in a stage.

Formula (1) describes the total time required for producing a batch of q_h items in stage j, and the period length P has to exceed the required time in each stage and for each product. The formula assumes that set-up activities for the next operation cannot start before the preceding operation has finished. If this assumption is not valid, the next machine can prepare for being set up while the former operation of the batch at another machine has not yet finished. This means we have to allow parallel (set-up) activities at different machines for the same batch. We can formulate this in Formula (2) as:

$$\begin{aligned} P &\geq \sum_{i=1}^{n_j^h} \left\{ d_{hi} + p_{hi} \cdot \left\lceil \frac{q_h}{m_{hi}} \right\rceil^+ \right\} && \forall j = 1..N, h = 1..H \\ d_{hi} &= max\left[0, s_{hi} - \sum_{k=1}^{i-1} \left\{ d_{hk} + p_{hk} \cdot \left\lceil \frac{q_h}{m_{hk}} \right\rceil^+ \right\} \right] && \forall i = 1..n_j^h \end{aligned} \qquad (2)$$

We have introduced a new variable d_{hi}: the delay imposed by the set-up of operation i not being ready when the batch arrives from the preceding operation. d_{hi} depends on the internal set-up time of operation i (s_{hi}) and the set-up and processing times of preceding operations. If parallel set-up activities at different machines are not allowed, as in (1), we have $d_{hi}=s_{hi}$.

It is important to note that if P is smaller than the right hand side expression of the second formula, a feasible schedule can still exist. However, such a schedule would require further close-scheduling: subsequent *operations* at a product in a stage will have to be performed in parallel. In (2), we only required that -if possible- set-ups at the subsequent machines are performed before the batch arrives. If overlapping production is applied for operation i of product h, the total batch q_h can be divided into nb_h equal subbatches $[q_h/nb_h]^+$, $nb_h \in [1,q_h]$. After finishing the first subbatch at machine 1, these parts are transferred to the second machine. The last subbatch might contain less items due to rounding differences. Only products with the longest throughput time in a stage need to apply overlapping production in order to reduce P.

Expression (3) results for P when overlapping production is allowed and the same number of subbatches is used for all operations of a product h. We assume that each operation requires a different machine and that m_{hi} machines are used for operation i of product h. The batch of product h at operation i is therefore divided over the available number of machines m_{hi} for this operation.

The subbatch size per machine is then: $\left\lceil \frac{q_h}{m_{hi} \cdot nb_h} \right\rceil^+$

$$P \geq \max_{i=1}^{n_j^h}\left[r_{hi} + p_{hi}\cdot\left\lceil\frac{q_h}{m_{hi}}\right\rceil^+ + \sum_{t=i+1}^{n_j^h}\left\{ p_{ht}\cdot\left\lceil\frac{q_h}{m_{ht}\cdot nb_h}\right\rceil^+\right\}\right] \quad \forall j = 1..N, h = 1..H$$

$$r_{hi} = \max_{l=1}^{i}\left[s_{hl} + \sum_{t=l}^{i-1}\left\{ p_{ht}\cdot\left\lceil\frac{q_h}{m_{ht}\cdot nb_h}\right\rceil^+\right\}\right] \qquad (3)$$

r_{hi} is the earliest starting time of the first subbatch of the i^{th}operation of product *h.*

If r_{hi-1} exists, $r_{hi} = max\left[s_{hi}, r_{hi-1} + p_{hi-1}\cdot\left\lceil\frac{q_h}{m_{hi-1}\cdot nb_h}\right\rceil^+\right]$ which is easier to compute.

Lemma If $nb_h = 1$ (overlapping production not allowed for product *h*), then Formula (2) and (3) are equivalent:

$$nb_h = 1 \Rightarrow \max_{i=1}^{n_j^h}\left[r_{hi} + p_{hi}\cdot\left\lceil\frac{q_h}{m_{hi}}\right\rceil^+ + \sum_{t=i+1}^{n_j^h}\left\{ p_{ht}\cdot\left\lceil\frac{q_h}{m_{ht}\cdot 1}\right\rceil^+\right\}\right] = \sum_{i=1}^{n_j^h}\left\{ d_{hi} + p_{hi}\cdot\left\lceil\frac{q_h}{m_{hi}}\right\rceil^+\right\}$$

Proof Appendix C.

We can express r_{hi} in terms of a start delay d'_{hi}, which is a modification of d_{hi}: This makes it easier to relate the earliest starting time at *i* with the delays imposed at earlier operations.

Let $$d'_{hi} = max\left[0, s_{hi} - \sum_{k=1}^{i-1}\left\{ d_{hk} + p_{hk}\cdot\left\lceil\frac{q_h}{m_{hk}\cdot nb_h}\right\rceil^+\right\}\right] \quad \forall h = 1..H,\ i = 1..n_j^h$$

then: $$r_{hi} = d'_{hi} + \sum_{k=1}^{i-1}\left\{ d'_{hk} + p_{hk}\cdot\left\lceil\frac{q_h}{m_{hk}\cdot nb_h}\right\rceil^+\right\}$$

Expression (3) describes the maximum of all minimal required throughput times for each product in each stage. It is therefore a *longest-path oriented lowerbound* on the period length P for a given stage definition.

Load oriented lowerbound on period length P

Expression (3) does not take into account the maximum capacity of each machine in a period. We therefore introduce a second restriction on the period length P that pays attention to the principle of a *load-oriented lowerbound.*

Let the i^{th} operation of product h be performed at machine k. We denote this by $i^h \in k$. The following expression results (Formula (4)):

$$P \geq \sum_{h=1}^{H} \sum_{i^h \in k} \left\{ s_{hi} + p_{hi} \cdot \left\lceil \frac{q_h}{m_{hi}} \right\rceil^+ \right\} \qquad \forall k = 1..K \text{ machines} \quad (4)$$

Note that the set-up time s_{hi} is the internal set-up time, i.e., the machine cannot process any job during this set-up time. The various operations $i^h \in k$ that have to be performed at this machine k may be allocated to different stages according to this capacity restriction. For the mean work load of the machine in a period, the stage to which operations are allocated does not matter in a basic unicycle PBC system, as we have discussed in Section § 4.4.3.

Relationship between batch size and period length

The two Formulas (3) and (4) assume that the batch sizes q_h are given, while P still has to be determined. However, the batch size of a part directly depends on the expected demand during the sales period for the end products in which it is used. The length of this sales period equals P and hence P influences the batch size. We already used the formula $q_h = D_h \cdot P$ in our description of the set-up effect in Chapter Four, where D_h is the demand for product h during a standard time unit (e.g., a year) and P is also expressed in this standard time unit. If we assume that demand for h is constant and evenly distributed over time, we can rewrite our problem as in Expressions (5) and (6).

Determine P and $nb_h, h = 1..H,$ such that

$$P \geq \sum_{h=1}^{H} \sum_{i^h \in k} \left\{ s_{hi} + p_{hi} \cdot \left\lceil \frac{P \cdot D_h}{m_{hi}} \right\rceil^+ \right\} \qquad \forall k = 1..K \text{ machines} \quad (5)$$

$$P \geq \max_{i=1}^{n_j^h} \left[r_{hi} + p_{hi} \cdot \left\lceil \frac{P \cdot D_h}{m_{hi}} \right\rceil^+ + \sum_{t=i+1}^{n_j^h} \left\{ p_{ht} \cdot \left\lceil \frac{P \cdot D_h}{m_{ht} \cdot nb_h} \right\rceil^+ \right\} \right] \qquad \forall j = 1..N, h = 1..H \quad (6)$$

Equation 5 can be rewritten to 5′ under the assumption that $\left\lceil \frac{P \cdot D_h}{m_{hi}} \right\rceil^+$ is integer:

$$P \geq \max_{k=1}^{K} \frac{\sum_{h=1}^{H} \sum_{i^h \in k} s_{hi}}{1 - \sum_{h=1}^{H} \sum_{i^h \in k} \frac{p_{hi} \cdot D_h}{m_{hi}}} \qquad (5')$$

The approach described above assumes that the allocation of operations to machines as well as the allocation of operations to the various stages is known in advance and does not depend on P. Therefore the approach does not influence the distribution of work load over the cells during a period, nor does it change the decomposition of the production system.

The model to determine a suitable value for the period length uses detailed information on the processing times and set-up times for making the products. Therefore, we denote this approach as '*detailed decoupled period determination*'. *Detailed* because of the type of information needed. *Decoupled* because the decision about the period length does not influence stage definition.

§ 5.2 Classical economic period determination

The problem of period length determination can also be approached from an economic lot sizing point of view. Such an approach makes a trade-off between costs related to the number of ordering periods in a year and costs related to the length of the period. Such an approach is still a decoupled period determination approach, but it requires less detailed information on the operating characteristics, such as processing times and set-up times per operation. Several contributions can be found in literature on production cycle determination.

The economic order interval approach applies the economic batch size formula to determine the optimal order interval for each product. Theory says that if this optimal order interval is used for a product, the minimal total costs of ordering and holding inventory will result. This standard economic ordering approach can be criticised, mainly because it determines an individual order interval per product without taking notice of the relationship with other products that are being produced in the same production system. PBC requires a single period length and hence single order interval for all products. Therefore, it requires an approach that takes into account the various products in the system when determining the economic order interval.

Since the 1960s, several mathematical models are developed that face this problem of determining a single ordering period for a combination of products. Goyal and Satir (1989) present an overview. These models basically minimize a total cost function consisting of (1) a joint cost component C that is involved each time a new period starts, (2) an extra individual ordering cost component sc_h if product h is ordered in a period, and (3) a holding cost component hc_h for holding one item of product h during one time unit on stock. The models assume that an item is on average half a period P on stock. The amount ordered per period is $q_h=D_h \cdot P$, where P is the length of a period (or production cycle) and D_h the demand per time unit. The total cost function is then assumed to be TC, which is minimized with respect to P.

$$TC = \frac{\left(C + \sum_{h=1}^{H} sc_h\right)}{P} + P \cdot \sum_{h=1}^{H} \left\{\tfrac{1}{2} \cdot D_h \cdot hc_h\right\} \qquad \rightarrow P = \sqrt{\frac{2\left(C + \sum_{h=1}^{H} sc_h\right)}{\sum_{h=1}^{H} \{D_h \cdot hc_h\}}}$$

Note that for the total cost function *TC* this interval P need not result in the lowest cost solution, as we have optimized it with respect to the condition that all products are ordered exactly once per period.

The above presented problem formulation is an example of the continuous time replenishment case. Several variants of this problem on the total cost function and the corresponding optimal solution are available in literature. For example, Boucher (1984) introduced a set-up time component in the cost function. Goyal and Satir (1989) presented an overview of deterministic and stochastic inventory models that allow for joint replenishment. Many of these models try to find different replenishment cycles for sets of inventory items. Within the set, each item has the same cycle length, but between the sets different cycles exists.

The discrete period part replenishment problem is a special variant of this problem, as it allows replenishment only within a single phase context. This work originates from the work of Wagner and Whitin (1958). In these models, the length of the base period (time between the phase moments) is fixed and not discussed. A cycle may consist of several periods.

Shtub (1990) discusses the use of these models in a group technology production situation. He states that a single cycle approach such as PBC can lead to an inefficient use of the production cells due to the possibility that major set-ups will be repeated each period. He presents an improvement heuristic based on the savings algorithm of Clarke and Wright (1963) for the capacitated lot sizing problem. Rachamadugu and Tu (1997) applied dynamic programming to obtain solutions for the uncapacitated problem. Jamshidi and Browne (1993) give an indication of the losses incurred if fixed production cycles are used. Muckstadt and Roundy (1993) describe the joint replenishment problem as a special case of ordering in a two stage serial assembly system. They apply a stationary nested power-of-two policy to obtain a near optimal solution with possibly different cycle lengths for the various part families.

Both approaches can be viewed as originating from a classical total cost perspective. Such a perspective pays no attention to the costs that result if the work load varies in subsequent periods. Literature on aggregate production planning does use such cost functions with respect to hiring/firing policies. Kaku and Krajewski (1995) have applied some of these insights in the determination of ordering cycles and period length.

The classical economic period determination models neither pay attention to the length of the throughput time of the batch in the production system. They are mainly focussed on the length of the production interval, assuming that no (holding) costs are incurred before or after this interval. Therefore, the models have to be revised in order to be suitable for period length determination in a PBC context. Section § 5.3 will develop such a model.

§ 5.3 Modelling detailed economic period determination

We propose to combine the economic approach to determine a suitable order interval with the detailed period determination approach, i.e., we want to include detailed information on the operating characteristics of the system when determining the period length. We have not found such an approach in literature. We denote the procedure as a *'detailed economic period determination'* approach. Note that it is not a decoupled approach. In other words, we relax the assumption that the definition of stages is known in advance before determining a suitable period length. The decisions on the number of stages and the allocation of operations to these stages depends on the length of the period, as has been shown in Chapter Four. Therefore, besides a period length P we will have to determine a stage definition. Stage definition will in our modelling approach be restricted to finding a suitable number of stages N, as this primarily influences the total throughput time in the system. We will also determine if overlapping production can effectively be used between successive operations, as this has an important effect on throughput time, length of period and required number of stages. The modelling approach will not determine a suitable allocation of operations to the stages. It therefore assumes that overlapping production can be applied between any two successive operations unless decided otherwise. The basic idea is to determine the effect of period size P, number of stages N, and number of subbatches nb_h on three relevant cost factors: holding costs in the system, ordering or set-up costs, and transfer costs:

- Holding costs, HC_h , are related to the average echelon inventory for product *h*
- Set-up costs per time unit, SC_{hi} , are involved each time a set-up of operation *i* for product *h* is required
- Transfer costs, TC_{hi} , are related to the number of subbatches of product *h* transferred at operation *i* (i.e., nb_h>1)

Before we present the model, we will explicate the relationship between these cost factors and the total cost of a PBC system that operates with period length P and number of subbatches nb_h. The notation that we use is introduced within the text and summarized at page 124.

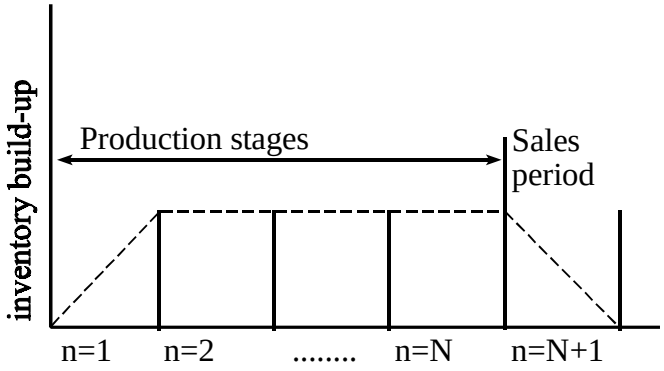


Figure 5.1 Build up of inventory (WIP and finished goods) during stages

holding costs

First, the cost of holding inventory. We relate these costs to the average amount of inventory in the system, i.e. we use echelon holding costs. In the modelling approach in this chapter, we assume that during the first production stage of PBC, all raw material is introduced into the system when the production activities start. During the next stages, the material is further processed, but no new material is added to the system inventory. When production has finished, the product is available for sale or delivery in stage N+1. The reason for this assumption is that we can no longer use information on the allocation of operations to the stages, as stage definition has not yet been performed.

Our new period determination approach assumes that on average the raw material is being introduced halfway the first production stage and the final product is sold or delivered halfway stage N+1, as seen in Figure 5.1. The material that is worked into each product is therefore on average N stages present in the system, irrespectible of the actual progress in making the product (parts, components, sub-assembly, or final product). The average time a batch q_h that is produced for sales of product h in stage N+1 is in the system is N periods of length P. The average total inventory in the system is then $\sum_h \mathrm{N} \cdot q_h$.

We compute the holding costs of a single product over the average time the product is in the system. The total holding costs for a product depends on both the average time it is in the system and the cost of holding one item of this product in stock during a standard time unit. This holding cost HC_h may vary per product h, to reflect differences in work content, amount and value of material, and so on. However, it is assumed that a good estimation of HC_h can be obtained without explicit knowledge of the distribution of operations over the various stages. In the next chapter, we will return to this assumption. The total cost of holding this inventory during a standard time unit is *TCH*, with

$$TCH = \sum_{\mathrm{h}=1}^{\mathrm{H}} \{\mathrm{N} \cdot q_h \cdot HC_h\} = \mathrm{N} \cdot \mathrm{P} \cdot \sum_{\mathrm{h}=1}^{\mathrm{H}} \{D_h \cdot HC_h\}$$

The number of stages N is in our approach a function of the decision variables P and nb_h. They directly influence the total time needed for producing a batch of product h. This total time is denoted as $\mathrm{TT_h}$, and we can reformulate Formula (3) to determine the length of $\mathrm{TT_h}$. Note that the number of operations for a product n_h no longer refers to a specific stage j.

$$TT_h = \max_{i=1}^{n_h} \left[r_{hi} + p_{hi} \cdot \left[\frac{\mathrm{P} \cdot D_h}{m_{hi}} \right]^+ + \sum_{t=i+1}^{n_h} \left\{ p_{ht} \cdot \left[\frac{\mathrm{P} \cdot D_h}{m_{ht} \cdot nb_h} \right]^+ \right\} \right]$$

N is an integer. We assume that the PBC system uses N stages for all products h:

$$\mathrm{N} \geq \max_{\mathrm{h}=1}^{\mathrm{H}} \left[\frac{TT_h}{\mathrm{P}} \right]^+ \qquad (7)$$

set-up costs

The second cost factor is the sum of set-up costs, ordering costs, and other costs that are incurred every time a new cycle starts. These costs will be involved irrespectible of the size of the orders or the length of the period. The costs of the frequency of operating the planning system (forecasting effort, program meetings, and so on) are included in these set-up costs. We calculate the set-up cost per standard time unit SC_{hi} and allocate the costs to the products h according to the time required per set-up at an operation i.

In a PBC system, the number of set-ups is 1/P per standard time unit. The set-up costs for an operation are the product of the set-up time for this operation s_{hi} and its set-up costs per standard time unit SC_{hi}. In a basic unicycle PBC system, each operation i of product h is performed once per period, and this frequency is independent of the stage it is allocated to. We therefore sum the set up times for all n_h operations of product h. The total set-up costs *per period* are $\sum_{h=1..H} \sum_{i=1..nh} s_{hi} \cdot SC_{hi}$. Total set-up costs *per standard time unit* is *TCS*, with:

$$TCS = \frac{\sum_{h=1}^{H} \sum_{i=1}^{n_h} \{s_{hi} \cdot SC_{hi}\}}{P}$$

transfer batch costs

Finally, we consider the costs of transferring batches between the subsequent operations. These costs depend on the number of subbatches of a product h: nb_h. If there is only one batch (nb_h=1) then TC_{hi}' reflects the cost of transportation and administration effort required between the subsequent operations. If the batch of product h is split into several subbatches (nb_h>1), the total transfer costs increase. The transportation and administration activities are performed more often and the utilization of the various operations may decrease as machines have to wait more frequent on the arrival of smaller batches. The extra transfer costs TC_{hi}" may vary per product and operation. We assume that the extra transfer cost is linear with the number of extra subbatches. Therefore, the transfer costs per standard time unit are *TCT*:

$$TCT = \frac{\sum_{h=1}^{H} \sum_{i=1}^{n_h} \{TC'_{hi} + (nb_h - 1) \cdot TC''_{hi}\}}{P}$$

We obtain the following model, which is a non-linear optimization problem with (H+1) decision variables.

$$\min_{P,\, nb_h(h=1..H)} \quad N \cdot P \cdot \sum_{h=1}^{H} \left\{ D_h \cdot HC_h \right\} + \frac{\sum_{h=1}^{H} \sum_{i=1}^{n_h} \left\{ s_{hi} \cdot SC_{hi} + TC'_{hi} + \left(nb_h - 1 \right) \cdot TC''_{hi} \right\}}{P} \tag{8}$$

$$s.t. \quad \{1\} \quad N \geq \max_{h=1}^{H} \left\lceil \frac{TT_h}{P} \right\rceil^{+} \tag{7}$$

$$\{2\} \quad P \geq \max_{k=1}^{K} \left\lceil \frac{\sum_{h=1}^{H} \sum_{i^h \in k} s_{hi}}{1 - \sum_{h=1}^{H} \sum_{i^h \in k} \frac{p_{hi} \cdot D_h}{m_{hi}}} \right\rceil \tag{4}$$

$$TT_h = \max_{i=1}^{n_h} \left\lceil r_{hi} + p_{hi} \cdot \left\lceil \frac{P \cdot D_h}{m_{hi}} \right\rceil^{+} + \sum_{t=i+1}^{n_h} \left\{ p_{ht} \cdot \left\lceil \frac{P \cdot D_h}{m_{ht} \cdot nb_h} \right\rceil^{+} \right\} \right\rceil \tag{9}$$

$$r_{hi} = \max_{l=1}^{i} \left\lceil s_{hl} + \sum_{t=l}^{i-1} \left\{ p_{ht} \cdot \left\lceil \frac{q_h}{m_{ht} \cdot nb_h} \right\rceil^{+} \right\} \right\rceil = \max \left\lceil s_{hi}, r_{hi-1} + p_{hi-1} \cdot \left\lceil \frac{q_h}{m_{hi-1} \cdot nb_h} \right\rceil^{+} \right\rceil \tag{3'}$$

Restriction{1} and {2} are both lowerbounds, either on the required actual capacity or on the required total time in the system. Restriction{1} neglects the possible product waiting time due to a machine not being ready when the subbatch arrives. At the other hand, Restriction{2} neglects the loss of capacity at machine *k* due to overlapping production some products at this machine[1]. Both restrictions assume that the waiting time factor is incorporated in the other restriction, i.e., they are both a lowerbound. We therefore used in both restrictions the greater than or equal to sign.

In a practical optimization approach, one could introduce a correction factor MI in the restrictions, with MI≤1. For example, introducing MI=0.95 in the left side of restriction {2} (MI·P ≥ ..) indicates that the period length P has to be determined such that the utilization of any machine *k* will not exceed 95%. If we introduce this factor in Restriction{1}, we obtain:

$$N \geq \max_{h=1}^{H} \left\lceil \frac{TT_h}{MI \cdot P} \right\rceil^{+} \tag{10}$$

[1] The waiting time of either the product or the machine is difficult to estimate, as it is not strictly necessary to start processing a subbatch as soon as it has arrived at the machine. If a machine has lack of capacity, the subbatch may have to wait until the machine is able to process all subbatches subsequently without intermediate waiting times. The machine therefore may introduce an extra product waiting time factor in the total time TT_h that the product is in the system, in order to reduce its own machine waiting time factor.

Let us summarize the notation that we used in the formulas in this section:

N	Number of stages	
P	Period length	
nb_h	Number of (equal) subbatches of product h	$h = 1..\mathrm{H}$
$\mathrm{TT_h}$	Total time required for processing q_h units of product h	$h = 1..\mathrm{H}$
n_h	Number of operations required for product h	$h = 1..\mathrm{H}$
D_h	Demand for product h per standard time unit	$h = 1..\mathrm{H}$
s_{hi}	Set - up time for i^{th} operation of product h in standard time units	$h = 1..\mathrm{H},\ i = 1..n_h$
m_{hi}	Number of parallel machines that perform i^{th} operation of product h	$h = 1..\mathrm{H},\ i = 1..n_h$
p_{hi}	Processing time for i^{th} operation of product h in standard time units	$h = 1..\mathrm{H},\ i = 1..n_h$
q_h	Number of units of product h required per period	$h = 1..\mathrm{H}$
r_{hi}	Earliest starting time of i^{th} operation of product h	$h = 1..\mathrm{H},\ i = 1..n_h$
HC_h	Holding cost for one unit of product h during a standard time unit	$h = 1..\mathrm{H}$
SC_{hi}	Set - up cost for i^{th} operation of product h per unit of time	$h = 1..\mathrm{H},\ i = 1..n_h$
TC'_{hi}	Transfer cost for the first batch at the i^{th} operation of product h	$h = 1..\mathrm{H},\ i = 1..n_h$
TC''_{hi}	Transfer cost for an extra subbatch at the i^{th} operation of product h	$h = 1..\mathrm{H},\ i = 1..n_h$
MI	Machine interference correction factor	

The approach has resulted in an optimization model for determining a period length and a subbatch strategy per product that minimizes the sum of three cost factors. The required number of stages is being determined as the minimal number of periods of length P that is required in order to cover the longest product throughput time $\mathrm{TT_h}$.

§ 5.4 Modelling different number of subbatches per operation

We have introduced the decision variables nb_h in the above developed model without paying attention to the possibility of having a different number of subbatches per operation. However, it may not be optimal to use the same number of subbatches for all operations of a product. An operation at a bottleneck may benefit more from processing a complete batch at a time, while the total throughput time of a product may benefit most if more subbatches are applied at an operation that takes a long processing time.

Within a context of a manufacturing system that is planned and controlled with a PBC system, it may also be better to distinguish between operations that operate with different numbers of subbatches. It will often be more difficult to apply a large number of subbatches for the transfer of material between cells than for transfer within a cell. As the costs will generally be higher for between cell transfer, a manufacturing system would probably prefer only one subbatch at the last operation in a cell in order to transfer the product only once per period to

another cell or stage decoupling stock. However, between other operations within the cell it may apply more subbatches in order to reduce the total throughput time of the batch. Furthermore, the products for which overlapping production should be applied (nb_h>1) may differ per stage. In one stage, product *h* may determine the longest throughput time, while in another stage a different product may benefit most from overlapping production. We should therefore allow a different number of subbatches per operation in our model of period length determination.

In order to introduce this possibility in our model, we have to modify the formula for TT_h. In this formula, we use the number of subbatches to determine the earliest starting time of the first subbatch of the product at the next operation. Let nb_{hi} denote the number of subbatches at operation *i* of product *h*. A different number of subbatches per operation means that the number of subbatches at the next operation might differ from the number of subbatches at the former operation. It is then not required that $nb_{hi} = nb_{hj}$ for $i \neq j$. To formulate this situation, we need to modify the determination of the earliest starting time at the next operation.

If we no longer require that $nb_{hi} = nb_{hj} \; \forall \; i \neq j$, we can distinguish the following cases:

(a) Non-nested case $nb_{hi-1} \geq nb_{hi}$

(b) Nested case $nb_{hi-1} \leq nb_{hi}$

§ 5.4.1 Non-nested batching policy $nb_{hi-1} \geq nb_{hi}$

A non-nested batching policy results in the next operation having the total batch divided into a smaller number of subbatches than used at the preceding operation. Between these operations, some waiting time may occur in order to compose the new subbatches, which are now larger than before. If this waiting time is ignored and processing at operation *i* starts as soon as the first subbatch of operation *i-1* arrives at *i*, then not all input material for the first subbatch at *i* will be available. This might result in waiting time for both machine and product at operation *i* instead of waiting time for the product only, especially if $p_{hi-1} > p_{hi}$.

In case of a non-nested batching policy provides the arrival time of the first subbatch of operation *i-1* at operation *i* not sufficient information for determining the finishing time of this subbatch at operation *i*. We need to know this finishing time to determine the earliest starting time at following operations. Therefore, we have to modify the expression that determines the earliest starting time of product *h* at operation *i*: r_{hi}.

Let

ra_{hi} = earliest effective starting time of i^{th} operation of product *h*

rs_{hi} = earliest arrival time of the first subbatch of operation *i - 1*

re_{hi} = earliest starting time to produce the first subbatch at operation *i* without intermediate idle time

We have:

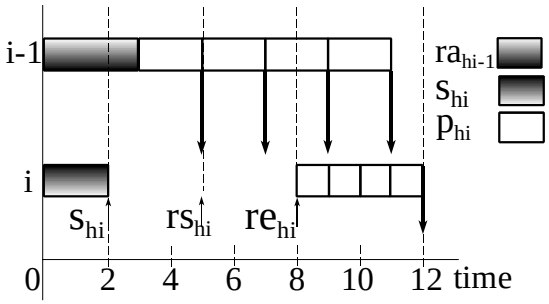
$$ra_{hi} = \max\,[s_{hi},\ rs_{hi},\ re_{hi}] \text{ with}$$

$$rs_{hi} = ra_{h\,i-1} + p_{h\,i-1} \cdot \left[\frac{q_h}{m_{hi-1} \cdot nb_{hi-1}} \right]^+$$

$$re_{hi} = ra_{h\,i-1} + \left(p_{hi} + \{p_{h\,i-1} - p_{hi}\} \cdot \left[\frac{nb_{hi-1}}{nb_{hi}} \right]^+ \right) \cdot \left[\frac{q_h}{m_{hi-1} \cdot nb_{hi-1}} \right]^+$$

If rs_{hi} exceeds the set-up time s_{hi} of operation i, this operation is allowed to start. However, to determine the earliest effective starting time that can be used for computing the total time needed for product h, we have to recognize that at operation i a complete subbatch has to be produced before it can be transferred to operation i+1. This complete subbatch at i may be composed of several subbatches of operation i-1[2].

The variable re_{hi} reflects that the arrival time of the last items for the first subbatch at i determines the finishing time at i. The earliest starting time at i is computed using backward scheduling from the arrival time of the last items of the first subbatch at operation i.



$ra_{hi}=\max[s_{hi},rs_{hi},re_{hi}]=$
$=\max[2,\ 3+2,\ 3+(1+\{2-1\}\cdot 4)\cdot 1]=$
$=\max[2,\ 5,\ 8] = 8$

$nb_{hi-1}=4$
$p_{hi-1}=2$
$nb_{hi}=1$
$p_{hi}=1$
$ra_{hi-1}=3$
$q_h=4$
$m_{hi-1}=1$
$m_{hi}=1$

Figure 5.2 Computing earliest starting time at operation i in the non-nested case

Figure 5.2 shows an example for which ra_{hi} is computed. Operation i-1 transfers each unit of product h as soon as it is ready to operation i, hence $nb_{hi}=q_h$. At operation i, no transfer of subbatches takes place: the complete batch has to be finished before it is transferred to

---

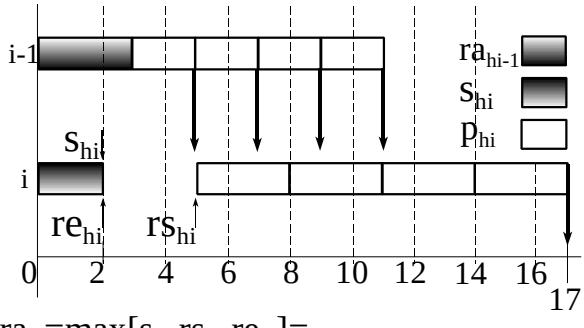
[2] The factor $z=[nb_{hi-1}/nb_{hi}]^+$ denotes the number of subbatches at operation i-1 that have to be completed to obtain sufficient items for the first subbatch at operation i. From the finishing time of these z subbatches at i-1 we have to withdraw the processing time of the items that were earlier made available to operation i through the transfer of subbatches (z-1). These (z-1) subbatches make it possible to start processing the first subbatch at i at time re_{hi} although not all required items are available at this time. The variable re_{hi} assumes that all items of these (z-1) subbatches of operation i-1 have been processed at i when the z^{th} subbatch arrives at i, so $p_{hi}\cdot(z\text{-}1) \cdot [q_h/(m_{hi-1}\cdot nb_{hi-1})]^+$ (their total processing time) is withdrawn from $ra_{hi-1} + p_{hi-1}\cdot z \cdot [q_h/(m_{hi-1}\cdot nb_{hi-1})]^+$, the earliest finishing time of the z^{th} subbatch at operation i-1.

operation *i*+1. Consequence of this batching policy is that, although operation *i* can start at time 5, it cannot finish before time 12, so starting at time 8 will make it possible to produce the first subbatch without interruption.

Note that a reduction in the number of subbatches at operation *i*-1 will lead to an increase of the total time needed on the two operations for this product. For example, if nb_{hi-1}=2, rs_{hi}=7, re_{hi}=3+(1+1·2)·2=9. The earliest finishing time of operation *i* is hence 9+4=13, and the earliest starting time on *i*-1 is in both cases 3.

The use of nb_{hi}=1 results in less waiting time for the machine that performs operation *i*. This is generally preferred at bottleneck machines. However, the introduction of several subbatches at a bottleneck results not necessarily in a lower utilization, as can be seen from operation *i*-1 in Figure 5.2. If more subbatches are distinguished at an operation, the utilization of the machine that performs this operation depends on the transfer times of the preceding operation and the actual starting time of the operation at the machine. If the actual starting time exceeds the earliest starting time, utilization might increase. Hence, these machines often use a time buffer between the earliest starting time and the actual starting time. The time buffer increases the utilization of these machines, but does also increase the total throughput time of the batch.

Note also that in an actual schedule for machine *i* the start of operation *i* of product *h* may be scheduled before re_{hi} but not before rs_{hi}: rs_{hi} is a strict lowerbound on the actual start of *i*.



ra_{hi}=max[s_{hi},rs_{hi},re_{hi}]=
=max[2, 3+2, 3+(3+{2-3}·4)·1]
=max[2,5,2]=5

nb_{hi-1}=4
p_{hi-1} =2
nb_{hi} =1
p_{hi} =3
ra_{hi-1} =3
q_h =4
m_{hi-1} =1
m_{hi} =1

Figure 5.3 ra_{hi}=rs_{hi}: operation i starts without interruption if first subbatch at i arrives

Figure 5.3 shows a situation where we have $nb_{hi-1} > nb_{hi}$ but $re_{hi} < rs_{hi}$. The situation is identical to the example in Figure 5.2, except the processing time at operation *i:* p_{hi} being 3 instead of 1. We see that the increase in processing time at operation *i* results in a different value for the earliest starting time that would make it possible to produce the complete subbatch at *i* without interruption (re_{hi}) and the earliest starting time due to the arrival of the first subbatch from *i*-1: ra_{hi}. This shows that it is necessary to use both earliest time estimators rs_{hi} and re_{hi} when determining ra_{hi} in the non-nested case $nb_{h\,i-1} \geq nb_{hi}$.

§ 5.4.2 Nested batching policy $nb_{hi-1} \le nb_{hi}$

The question arises if both earliest time estimators will also be necessary in the other situation that we distinguished, the nested batching policy where $nb_{hi-1} \le nb_{hi}$. The following result is obtained:

Lemma: If $\left\{\left(nb_{hi-1} \le nb_{hi}\right) \wedge \left(nb_{hi} \in \left[1, q_h\right] \ \forall i = 1..n_h\right)\right\} \Rightarrow \left\lceil \frac{nb_{hi-1}}{nb_{hi}} \right\rceil^{+} = 1 \Rightarrow re_{hi} = rs_{hi}$

Proof:
$$re_{hi} = ra_{h\,i-1} + \left(p_{hi} + \left\{p_{h\,i-1} - p_{hi}\right\} \cdot \left\lceil \frac{nb_{hi-1}}{nb_{hi}} \right\rceil^{+} \right) \cdot \left\lceil \frac{q_h}{m_{hi-1} \cdot nb_{hi-1}} \right\rceil^{+} =$$

$$= ra_{h\,i-1} + \left(p_{hi} + \left\{p_{h\,i-1} - p_{hi}\right\} \cdot 1 \right) \cdot \left\lceil \frac{q_h}{m_{hi-1} \cdot nb_{hi-1}} \right\rceil^{+} =$$

$$= ra_{h\,i-1} + \left(p_{hi} + p_{h\,i-1} - p_{hi} \right) \cdot \left\lceil \frac{q_h}{m_{hi-1} \cdot nb_{hi-1}} \right\rceil^{+} =$$

$$= ra_{hi-1} + p_{h\,i-1} \cdot \left\lceil \frac{q_h}{m_{hi-1} \cdot nb_{hi-1}} \right\rceil^{+} =$$

$$= rs_{hi}$$

The reason for this equality is that all parts for the first subbatch at i are included in the first delivery of a subbatch from i-1. Therefore, ra_{hi}=max[s_{hi}, rs_{hi}] and the time estimator re_{hi} is no longer required in case of nested batching strategies where $nb_{h\,i-1} \le nb_{hi}$.

The relationship between the number of subbatches at subsequent operations does influence the completion time of the whole batch for this set of operations. Within the set of nested batching strategies, a subset can be defined that will result in smaller intermediate machine waiting times during the processing of the batch. A strategy within this subset will be denoted as a *powered nested batching policy* , which is defined as follows:

A batching policy nb_{hi} is powered nested if and only if

$$\exists \quad a, c, y \in \mathrm{N}, \qquad a \ge 1, c \ge 2$$

such that
$$nb_{hi} = \frac{q_h}{a \cdot m_{hi}}$$

$$nb_{hi} = c^{y} \cdot nb_{hi-1}$$

$$nb_{hi} \in \left[nb_{hi-1}, \ldots, \frac{q_h}{m_{hi}} \right] \subseteq \left[1, .. q_h\right] \subset \mathrm{N}$$

For nb_{hi} to be powered nested, the next condition must hold: $\frac{q_h}{m_{hi}} \in \mathrm{N}$
This is a necessary but not a sufficient condition. It is necessary for nb_{hi} to become integer, but not sufficient for the powered nestedness of nb_{hi}.

Remark: A powered nested batching policy is a nested policy, as $nb_{hi} = c^y \cdot nb_{h\,i-1} \geq nb_{h\,i-1}$.

Suitable nested batching policies may a priori be restricted to a specific value of *c*. We will denote this by a *power-of-c nested batching policy*, e.g., a power-of two nested batching policy has *c*=2.

The main contribution of powered nested batching policies is in the logistical organization of the batching process. With a powered nested batching strategy, subbatches from earlier operations may be combined into larger subbatches, but this will never require a further division of one of the received subbatches. The received subbatches remain consistent during processing at i.

The logistical organization of non-powered nested batching policies is more complex. Splitting a batch that just arrived into two[3] subsets results in temporally stocks between operation *i*-1 and *i*, as shown in Figure 5.4. A non-powered nested batching policy may also result in unequal subbatches at operation *i*. Unequal subbatches cause extra logistical complexity at this and following operations. Powered nested batching policies do not result in these effects.

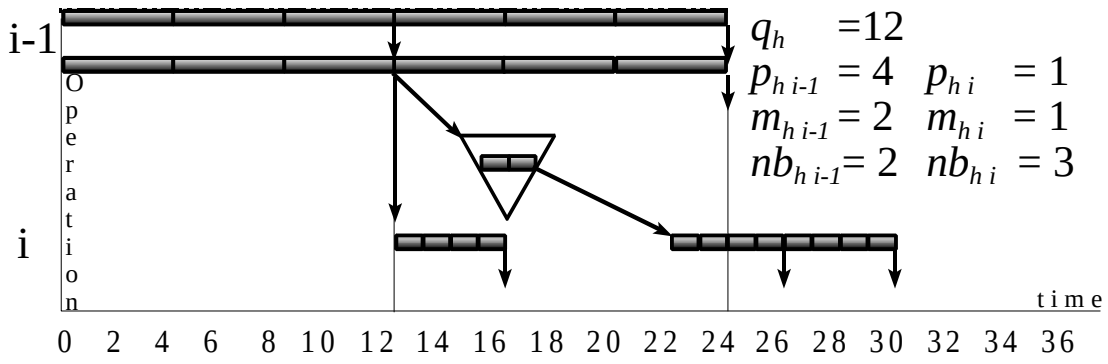


Figure 5.4 Non-powered nested batching policies result in temporally stock

If we want to determine a powered nested batching policy for operation *i*, we have to find appropriate values for *a*, *c*, and *y*. We will determine such values for the data that we used in Figure 5.4. The data in this figure describe a problem with a batch size of q_h=12 items. At operation *i*-1, m_{hi-1} =2 machines are available, each using a processing time of p_{hi-1}=4 time units per item. The whole batch is divided into nb_{hi-1}=2 subbatches of six items each. At operation *i*, we only have one machine available (m_{hi} =1), while the processing time per item p_{hi} is now equal to 1 time unit.

[3] For example, one subbatch that immediately may start processing and another that has to await processing until the next subbatch from the preceding operation has arrived.

What number of subbatches at operation *i* would result in a powered nested batching strategy and what parameters *a*, *c*, and *y* are necessary to accomplish this?

It is easy to see that nb_{hi}=1 would result in a non-nested policy. For nested policies, nb_{hi} needs to be ≥ 2. If nb_{hi}=nb_{hi-1}=2, we require that c^y=1 with $c \geq 2$. Hence y=0 and a=6.

nb_{hi}	a≥1	c≥2	y≥0	batching strategy
1	-	-	-	non-nested
2	6	2	0	powered nested
3	4	-	-	non-powered nested
4	-	2	1	powered nested
5	-	-	-	non-powered nested
6	2	3	1	powered nested
7	-	-	-	non-powered nested
8	-	2	2	non-powered nested
9	-	-	-	non-powered nested
10	-	5	1	non-powered nested
11	-	-	-	non-powered nested
12	1	6	1	powered nested

Table 5.1 Suitable values of *a*, *c*, and *y* for powered nestedness of *nb* at operation *i*

For nb_{hi}=3 we have already demonstrated that we face a non-powered nested batching policy. This non-powered nested batching policy introduces the possibility of an extra machine waiting time at the next operation *i*+1. The second subbatch that will arrive at *i*+1 consists of items from both the first and the second subbatch that arrives from *i*-1. Hence, at operation *i*+1 we will face some extra machine waiting time between the earliest start moment of the first subbatch at this operation and the earliest finish moment of the last subbatch at this operation, as is illustrated in Figure 5.5, where we compare the non-powered nested batching policy nb_{hi}=3 with the powered nested batching policy nb_{hi}=4.

For nb_{hi}=4, 6, and 12, values for *a*, *c*, and *y* can be found that make them a powered nested batching strategy. However, for nb_{hi}=5, 7, 8, 9, 10, and 11, we are not able to split the total number of items to be made at a single machine (12) in this number of equal subbatches, i.e., we cannot find an appropriate value for parameter *a*. In Table 5.1 we have summarized the values that we found for the various subbatch strategies that are possible at operation *i*.

We conclude that for this example it is possible to find values that result in powered nestedness of nb_{hi}.

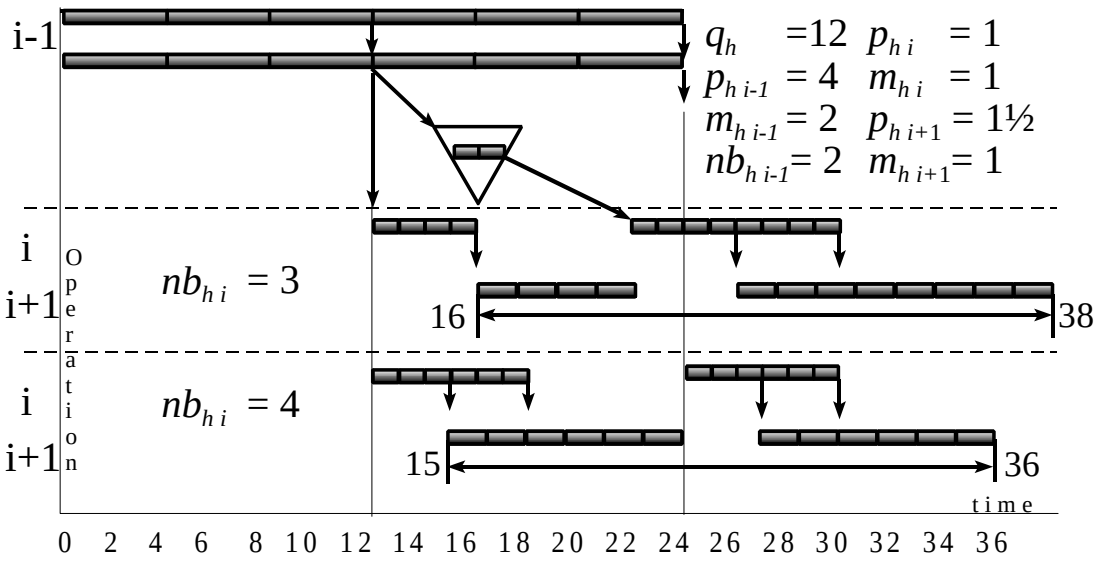


Figure 5.5 Powered nested batching at *i* generate less machine waiting time at *i*+1

In Figure 5.5 we have the same initial situation as in Figure 5.4, but we have added an extra operation *i*+1. The processing time at operation *i*+1 is 1½ time unit. If the non-powered nested batching strategy nb_{hi}=3 is used, the next operation can start earliest at time 16. It finishes at time 38, hence a throughput time of at least 38-16=22 units at operation *i*+1. Note that the processing time at this operation is 12 times 1½=18 time units. The powered nested batching strategy nb_{hi}=4 results in a lower throughput time of 36-15=21 units. Hence, this strategy results in a lower machine waiting time. Other powered nested batching policies result also in a throughput time of 21, which shows that an idle time of at least 3 time units is unavoidable. However, the non-powered nested batching policy nb_{hi}=3 requires one time unit extra idle time at this operation *i*+1, and so increases the machine waiting time.

The reason for this extra delay at *i*+1 for a non-powered nested batching policy is the delay in the transfer of an intermediate batch from operation *i*. This batch contained items from both the first and second subbatch of operation *i*-1 and hence had to wait until the second subbatch from operation *i*-1 arrived at operation *i* before processing could be continued. A powered nested batching policy will never allow such a composition of items of several subbatches into one new subbatch for the following operation.

The earliest start of a subbatch at *i* is for powered nested batching policies only a function of:

- the arrival time of the subbatch of *i*-1 that contains all its items
- the number of preceding subbatches that also contained items from this subbatch of *i*-1
- the set-up time at *i* (assumed to be negligible in example)
- the processing time at *i*

In case of powered nested batching policies, we do not have to correct for the effect of incomplete batches, as was done in the function re_{hi} for non-nested batching policies. To conclude, the discussion on different number of subbatches in subsequent operations has shown that for non-nested batching policies it is more difficult to determine the earliest effective starting time of an operation such that a machine can start producing without facing intermediate idle time. For nested batching strategies, this problem no longer exists. Within the set of nested policies, we have distinguished the subset of powered nested batching policies, which result in less logistical organization and co-ordination problems. Nested batching policies that are non-powered might result in extra machine waiting times and hence longer throughput times at this operation.

From this, we may not conclude that a powered nested batching policy should always be preferred. We have shown two cases where its application generates good results. Firstly, the situation where preceding operations have more machines available to produce the batch than available at operation i. Secondly, the situation where idle time at bottleneck operations has to be minimized. However, in deciding about a batching policy, we do not always aim at minimizing the total time needed for the sequence of operations, nor the idle time at each machine. The more general objective of minimizing total cost with respect to available capacity allows for using various batching policies. Next, we present a mathematical model for determining both P and a suitable subbatch strategy that tries to minimize total cost. Total costs are considered a function of the total time needed, the cost of set-ups, and the cost of subbatch transfers.

§ 5.4.3 Period length determination with (non)-nested batching policies

The model for determining a suitable period length that we described in Expression (8) has to be modified to incorporate the presence of various batching policies. The number of decision variables increases, as we have to determine for each operation *i* of a product *h* a possibly different number of subbatches. The number of constraints remains equal, but computation of Constraint {1} has to be modified, as the total time a product is in the system depends on the batching strategy at each operation. We apply a recursive expression for determining the earliest effective starting time for any subbatch at an operation $ra_{hi}(j)$ in (14). Our formulation includes both the non-nested and the nested subbatching strategies. This makes the expression quite complicated.

Define:

$ra_{hi}(j)$ = earliest effective starting time of the j^{th} subbatch at the i^{th} operation of product *h*

$\lfloor x \rfloor$ = nearest integer smaller than or equal to *x*

Constraint {2} does not change in the modified model, although we have shown that the amount of idle time at an operation *i* depends on the batching policy at the preceding operation *i*-1. The reason for this is that the idle time between the operations can be avoided by postponing the start of the operation. Therefore, the capacity-based lowerbound on the period length does not change.

We obtain the following model:

$$\underset{\mathrm{P},\, nb_{hi}}{min} \quad \mathrm{N}\cdot\mathrm{P}\cdot\sum_{h=1}^{H}\left\{D_h\cdot HC_h\right\}+\frac{\sum_{h=1}^{H}\sum_{i=1}^{n_h}\left\{s_{hi}\cdot SC_{hi}+TC'_{hi}+\left(nb_{hi}-1\right)\cdot TC''_{hi}\right\}}{\mathrm{P}} \tag{11}$$

$(i=1..n_h, h=1..H)$

$$s.t. \quad \{1\} \quad \mathrm{N}\ \geq\ \max_{h=1}^{H}\left\lceil\frac{TT_h}{MI\cdot P}\right\rceil^{+} \tag{10}$$

$$\{2\} \quad \mathrm{P}\ \geq\ \max_{k=1}^{K}\left\lceil\frac{\sum_{h=1}^{H}\sum_{i^h\in k}s_{hi}}{1-\sum_{h=1}^{H}\sum_{i^h\in k}\frac{p_{hi}\cdot D_h}{m_{hi}}}\right\rceil \tag{4}$$

$$TT_h = \max_{i=1}^{n_h}\left\lceil ra_{hi}\left(nb_{hi}\right)+p_{hi}\cdot\left\lceil\frac{P\cdot D_h}{m_{hi}\cdot nb_{hi}}\right\rceil^{+}+\sum_{t=i+1}^{n_h}\left\{p_{ht}\cdot\left\lceil\frac{P\cdot D_h}{m_{ht}\cdot nb_{ht-1}}\right\rceil^{+}\right\}\right\rceil \tag{12}$$

$$ra_{hi}(1) = \max\left[\begin{array}{l} s_{hi},\quad ra_{hi-1}(1)+p_{hi-1}\cdot\left\lceil\frac{P\cdot D_h}{m_{hi-1}\cdot nb_{hi-1}}\right\rceil^{+},\\ ra_{hi-1}(1)+\left(p_{hi}+\left\{p_{hi-1}-p_{hi}\right\}\cdot\left\lceil\frac{nb_{hi}-1}{nb_{hi}}\right\rceil^{+}\right)\cdot\left\lceil\frac{P\cdot D_h}{m_{hi-1}\cdot nb_{hi-1}}\right\rceil^{+}\end{array}\right] \tag{13}$$

$$ra_{hi}(j) = \max\left[\begin{array}{l} ra_{hi}(j-1)+p_{hi}\cdot\left\lceil\frac{P\cdot D_h}{m_{hi}\cdot nb_{hi}}\right\rceil^{+},\\ ra_{hi-1}\left(\left\lceil\frac{1}{P\cdot D_h}+\{j-1\}\cdot\frac{nb_{hi}-1}{nb_{hi}}\right\rceil^{+}\right)+p_{hi-1}\cdot\left\lceil\frac{P\cdot D_h}{m_{hi-1}\cdot nb_{hi-1}}\right\rceil^{+}+\\ \quad\left\{(j-1)\cdot\frac{nb_{hi}-1}{nb_{hi}}-\left\lceil(j-1)\cdot\frac{nb_{hi}-1}{nb_{hi}}\right\rceil^{-}\right\}\cdot p_{hi}\cdot\left\lceil\frac{P\cdot D_h}{m_{hi}\cdot nb_{hi-1}}\right\rceil^{+},\\ ra_{hi-1}\left(\left\lceil j\cdot\frac{nb_{hi}-1}{nb_{hi}}\right\rceil^{+}\right)+p_{hi-1}\cdot\left\lceil\frac{P\cdot D_h}{m_{hi-1}\cdot nb_{hi-1}}\right\rceil^{+}-\\ \quad\left\{\left\lceil j\cdot\frac{nb_{hi}-1}{nb_{hi}}-\frac{1}{P\cdot D_h}\right\rceil^{-}-(j-1)\cdot\frac{nb_{hi}-1}{nb_{hi}}\right\}\cdot p_{hi}\cdot\left\lceil\frac{P\cdot D_h}{m_{hi}\cdot nb_{hi-1}}\right\rceil^{+}\end{array}\right] \tag{14}$$

Expression (13) determines the earliest effective starting time of the first subbatch at operation *i*. The expression directly relates to the one presented in Section § 5.4.1 on non-nested batching. It consists of the maximum of three variables: the set-up time at the current operation *i*, the earliest arrival time of the first subbatch from the preceding operation, and the earliest starting time for which the first subbatch at operation *i* can be completed without intermediate waiting time. The latter factor is only necessary in case of non-nested batching strategies.

Expression (14) determines the earliest starting times of the following subbatches at this operation. It also consists of the maximum of three variables. The first variable is simply the sum of the earliest effective starting time of the preceding subbatch at this operation and the processing time of one subbatch at this operation. The second variable determines the arrival time of the subbatch from the preceding operation that contains the *first* element of the j^{th} subbatch at operation *i*. In order to identify this originating subbatch at operation *i*-1, we developed a transformation of *j* to o_{i-1}:

Originating subbatch o_{i-1} at $i-1$ containing first element of j^{th} subbatch at operation i =

$$o_{i-1} = \left\lceil (j-1) \cdot \frac{nb_{hi-1}}{nb_{hi}} + \frac{1}{q_h} \right\rceil^{+}$$

The factor $1/q_h = 1/(P \cdot D_h)$ indicates the first element of the j^{th} subbatch. This arrival time consists of the earliest effective starting time at the preceding operation and the processing time of the $o_{i-1}{}^{th}$ subbatch at that operation. We have to add a factor consisting of the processing time of elements in this $o_{i-1}{}^{th}$ subbatch that have to be processed in the $j\text{-}1^{th}$ subbatch at operation *i*. The processing time of these elements at operation *i* is:

$$\left\{ (j-1) \cdot \frac{nb_{hi-1}}{nb_{hi}} - \left\lceil (j-1) \cdot \frac{nb_{hi-1}}{nb_{hi}} \right\rceil^{-} \right\} \cdot p_{hi} \cdot \left\lceil \frac{P \cdot D_h}{m_{hi} \cdot nb_{hi-1}} \right\rceil^{+}$$

The third variable in Expression (14) determines the earliest arrival time of the last item in the j^{th} subbatch at *i*. Again, we compute the earliest starting time of the originating subbatch at *i*-1, and add the processing time of this batch at *i*-1. However, we do not need to await the completion of this whole batch at *i*-1 before we start at *i*, and hence we subtract the amount of processing time at *i* that already can be scheduled because of items originating from earlier subbatches.

This completes our discussion on a variable batching strategy within a detailed economic period determination approach. We have worked it into a mathematical model for determining the period length P in a PBC system. In the next sections, we will explore and develop solution approaches for this model and show the effect of period length on the various cost factors in a PBC system.

§ 5.5 Solution methods for detailed economic period determination model

We have developed a mathematical model for determining a period length P and a variable subbatch strategy. The model calculates the minimal number of stages N that will be necessary in order to finish a product within the available throughput time N·P. In finding appropriate values for these variables, it tries to minimize the sum of inventory holding costs, set-up costs, and costs for the transfer of subbatches between operations.

In general, there will be no solution methods available that are able to solve this model to optimality. This is due to the integer restrictions, the number of decision variables, and the non-linearity of the constraints. These aspects make it quite difficult both to obtain and to proof optimality of a solution. However, we are not primarily interested in finding such an optimal solution. We want to gain insight in the effect of period length and other PBC design factors on system performance. Therefore, we will already be satisfied when we are able to improve our understanding of the problem through the identification of feasible solutions and good approximations of the optimal solution. A heuristic approach may be used to provide these approximations.

In order to develop such a heuristic solution approach, we will use insights from literature on related problems, presented in Appendix D, and insights on the cost structure of the model with equal subbatches. This will help us to identify general principles for solutions of this model. Literature provides no direct solution approach for our model. We therefore develop a search procedure that tries to find a solution on a restricted domain. The value of such a solution approach is stressed by Williams, Tüfeki and Akansel (1997). It seems useful to solve our model subsequently for various values of N (the number of stages). This means, we have to minimize the sum of a linear holding cost (N is fixed) and the set-up + transfer costs subject to a constraint with respect to the minimum and maximum value of the make span for which the value of N holds. This heuristic search approach is described in Section § 5.5.2, but first we have to improve our understanding of the cost structure of our model. This will help us to develop a more efficient search procedure, for the number of decision variables is high.

First, we discuss the cost structure of the equal subbatch model in Section § 5.5.1. We use data from an example problem that is introduced in this section. In the next Section § 5.5.2 we introduce the enumerative search heuristic, an initial solution approach for general model, and discuss the results for the illustrative example. The enumerative search heuristic finds a suitable variable batching strategy *given* a value of the period length P. Section § 5.5.3 presents a progressive search heuristic that finds both a period length P and a variable batching strategy. Finally, we present an exhaustive search heuristic that further tries to improve the solutions found by the other two heuristics, and compare the results.

§ 5.5.1 Exploring the cost structure of the model with equal subbatches

We have to improve our understanding of the sensitivity of our model for changes in the decision variables period length and subbatch strategy, in order to develop a suitable and efficient heuristic solution approach. We will first present the data of a simple example problem for which we are able to show the consequences of changes in the decision variables.

Example problem

Table 5.2 shows the data of the example problem. We consider only two products with a known yearly demand. The first product requires nine operations, the other product eight. All operations are performed at different machines or cells, and for each operation we have just one machine available. In total, we have seventeen machines in our system and there is no interaction between the products on any resource. Machines that perform operations for the same product all have both identical set-up times and identical item processing times.

h	Product index	1	2	
D_h	Demand	1040	800	products per year
n_h	number of operations for product h	9	8	operations per product
s_{hi}	Set-up time operation i (years)	0.00721	0.00577	years $\forall$ i=1..n_h
	Set-up time operation i (hours)	15	12	hours $\forall$ i=1..n_h
p_{hi}	Processing time operation i (years)	0.00048	0.00072	years $\forall$ i=1..n_h
	Processing time operation i (hours)	1	1.5	hours $\forall$ i=1..n_h
m_{hi}	number of machines for operation i	1	1	machine $\forall$ i=1..n_h
P_{min}	Minimum period length (capacity)	0.01442		years = 3.75 days
HC_h	Holding costs	4		per unit per year
SC_{hi}	Set-up costs	50		per set-up $\forall$ i=1..n_h
TC'_{hi}	Transfer costs per proces batch	0.4		per batch $\forall$ i=1..n_h
TC''_{hi}	Transfer costs per extra subbatch	0.4		per transfer $\forall$ i=1..n_h

Table 5.2 Data example problem

For this example problem, we have to determine a period length P, and a subbatch strategy. In order to explore the cost structure of the model, we restrict our attention first to the equal subbatch situation: a product uses the same number of subbatches at each operation. As we have only one machine per operation in our example problem, one of the reasons for preferring a variable subbatch strategy is not present. Usage of a variable subbatch strategy in the exploration of the cost structure results in a huge increase of the number of decision variables. Therefore, we restrict our attention in this section to the equal subbatch strategy.

The mathematical model calculates from the values of the decision variables P and nb_h the total throughput time of each product in the system TT_h (Formula (9)). Subsequently, the required number of stages N can be calculated according to Formula (7) and the costs of the solution according to (8).

Figure 5.6 shows the various partial cost curves as well as the total cost curve as a function of the period length P for the example problem[4] with two subbatches (nb_h =2) for all products.

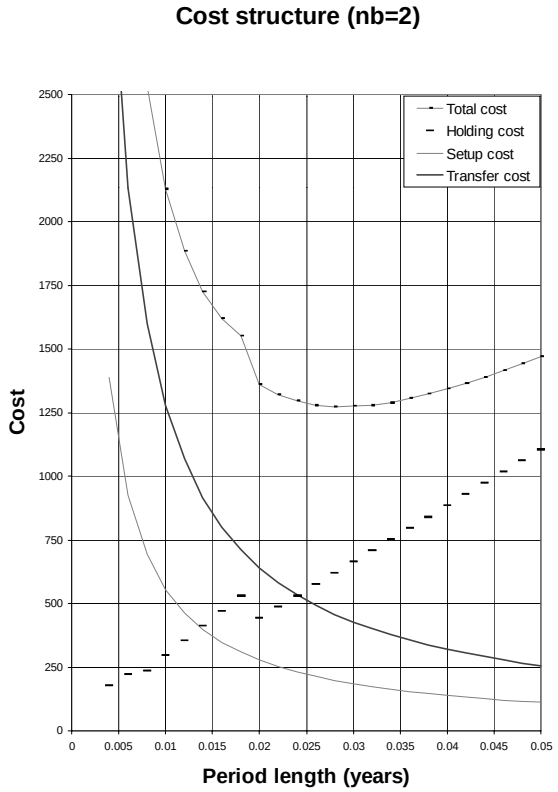


Figure 5.6 Partial and total cost curves

Both the set-up cost curve *Scost* and the transfer cost curve *Tcost* show a rather familiar pattern: the costs are high if the period length is small, because the number of set-ups and transfers are a linear function of the number of periods, and this number is inversely proportional to the period length.

The holding cost function *Hcost* does show a less familiar pattern. The curve is discontinuous and has two breakpoints (at P=±0.008 and P=±0.019). These breakpoints are caused by the decrease in the number of stages required to produce a batch of products if P increases. Batch sizes increase linearly with the period length, as q_h=P·D_h and the holding costs depend on the amount of units in the system. However, holding costs also depend on the total time TT_h these units stay in the system. If the period length increases, the throughput time of the batch can be divided amongst a decreasing number of stages. If a reduction in the number of stages occurs, this reduces the amount of items in the system and hence the holding costs.

[4] In Figure 5.6 we have included small period lengths that are in fact infeasible if no parallel machines are applied. However, we think this gains more insight in the cost structure. In developing a solution approach, we will consider the capacity limits of the system.

Discontinuity of one of the partial cost functions results in discontinuity of the total cost function *Total*. That means that we cannot simply differentiate this function to obtain a minimum. We have to apply a more refined search procedure that checks on the possible value of N (the number of stages) at a specific period length P. Furthermore, we may also have to search for a suitable and cost effective subbatch strategy. In Figure 5.6 we fixed the number of subbatches at two for all products. However, it may be possible to find lower cost solutions with other numbers of subbatches. We therefore first explore the change in the cost structure if we apply other numbers of subbatches for all products before we develop a search approach for a period length P.

Figure 5.7 illustrates the effect of an increasing number of subbatches on the total cost curves. It consists of four figures, each representing a specific number of subbatches, respectively nb=1 (i.e. no extra transfer batches), nb=2, nb=3 and nb=4. Note that in all situations we have an equal number of subbatches policy: $nb_{hi} = nb_h \; \forall \; i=1..n_h$, and we use the same number of subbatches for all products $nb_h = nb \; \forall \; h$=1..H.

Each figure in Figure 5.7 shows seven total cost curves, for N=1,2,..,7. These cost curves represent the total cost as a function of the period length P if exactly N=n stages are used to produce the batches. We call these curves *conditional cost curves*, as they assume that N=n, independent of the period length. The lowest conditional cost curve is of course N=1. For a specific value of P, the total cost becomes higher as N increases. The actual total cost curve for the specified batching policy may be discontinuous, but will proceed along the conditional cost curves in these figures. This actual cost curve is identified with the underscore symbol.

Figure 5.7a shows the conditional cost curves if no batch splitting is applied and Figure 5.7b if nb=2. We will first concentrate on the changes that occur in the conditional cost curves due to the increasing number of subbatches. We see that all conditional cost curves show both a *rightward shift* and an *upward shift*: the total cost increases due to the higher number of transfers, both per period and over the year. This cost increase is strongest if the period length is small, as this leads to a high number of periods a year. The curve shifting leads to an important change with respect to the period length for which the curves have their minimum conditional total cost. We conclude that:

the optimal period length increases if the number of subbatches increases.

If we further take a look at Figure 5.7c and Figure 5.7d we find that the distance between the optimal period length for conditional cost curve N=n+1 and the optimal period length for curve N=n increases as the number of subbatches increases. We conclude that:

the optimal period length becomes increasingly sensitive
for changes in the number of stages N
if the number of subbatches increases.

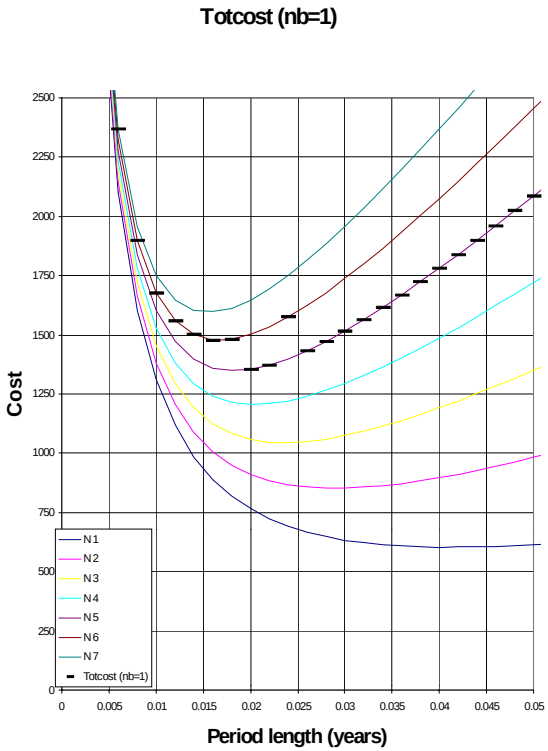


Figure a Cost curves for *nb*=1

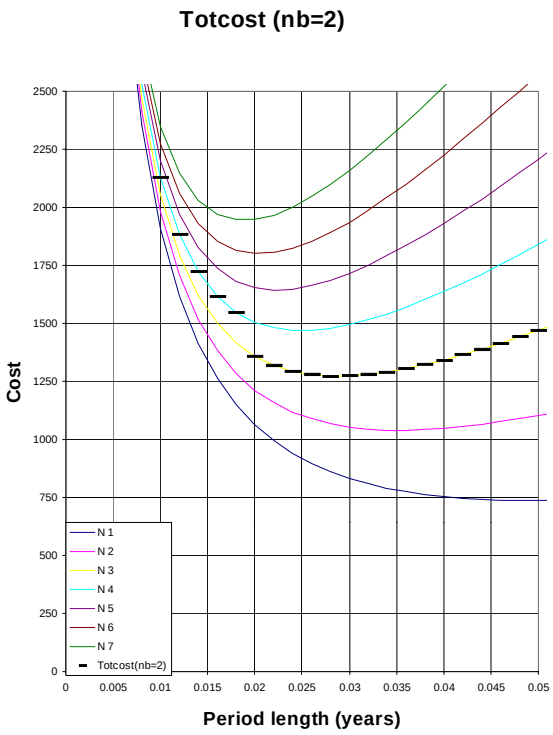


Figure b Cost curves for *nb*=2

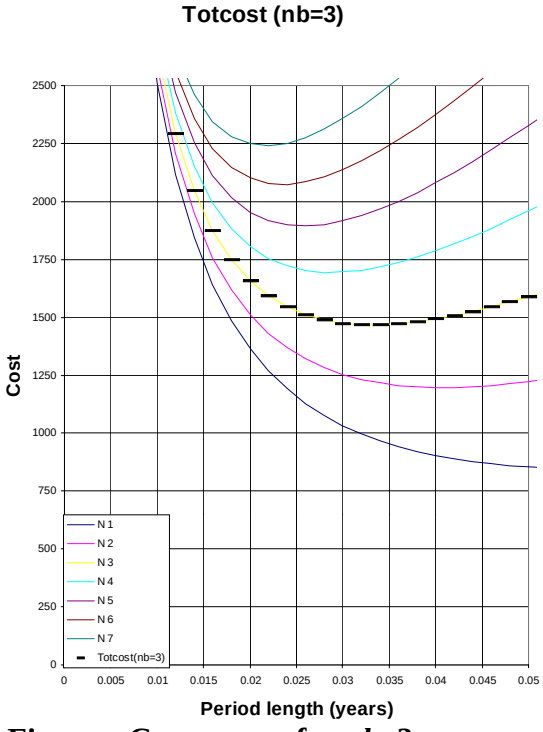


Figure c Cost curves for *nb*=3

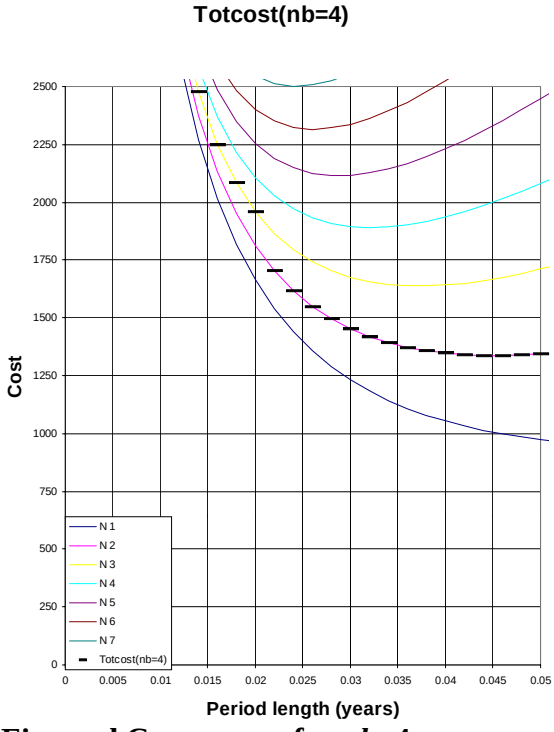


Figure d Cost curves for *nb*=4

Figure 5.7 Conditional and actual total cost curves for increasing numbers of subbatches

Both effects can be used in designing an adequate solution procedure for the general problem of finding both suitable values for the number of subbatches per operation and for the period length in order to minimize total cost.

We now pay attention to the change in the *actual* cost curves if we increase the period length P (while still applying an equal number of subbatches strategy). Remind that these curves are identified with the underscore symbol in the Figure 5.7a-d . The actual cost curve for *nb*=1, *nb*=2 and *nb*=4 are all discontinuous in the selected domain of period length values, so there occurs a change in the required number of stages due to a change in the period length.

If *nb*=1, the valid N-values are 6 (or higher) if P is smaller than 0.02, and 5 for period lengths higher than this breakpoint, as shown in Figure 5.7a. The minimum cost (1354) is found at the breakpoint. That means that at this period length for at least one product *h* the throughput time TT_h almost equals the available time of P·N, where N=5. A slightly smaller period length P leads to a higher required number of stages. A further increase of the period length normally leads to somewhat more slack between throughput time and available time. Eventually this may lead to a further decrease of the required number of stages.[5]

The actual cost curve at *nb*=2 is found at N=4 and N=3. The actual cost at this batching policy is much lower compared with *nb*=1, as the reduction in holding costs exaggerates the increase in transfer costs[6]. The minimum cost for *nb*=2 is not found at a breakpoint, but at the minimum of the conditional cost curve N=3: P=0.028 with a total cost of 1274.

If we further increase the number of batches to *nb*=3, the actual cost curve is found at N=3. The optimum is found at a higher P (0.034) with a higher total cost 1467. Note that we see here the effect of both the upward and rightward shift of the conditional cost curve of N=3. The minimum is taken at a higher P and the total cost is higher if the number of subbatches increases while the required number of stages remains the same. The actual minimal total cost as a function of the number of subbatches decreased when *nb* increased from 1 to 2, but increased when *nb* increased to 3. If we still further increase the number of subbatches to 4 (Figure 5.7d), a new breakpoint occurs at P= 0.022, decreasing the required number of stages to 2. The minimum cost occurs at P=0.046 with a cost of 1337. So the further increase of the number of subbatches reduces the minimal total cost compared with *nb*=3. However, if we

---

[5] Remark the momentary increase of the actual cost curve for *nb*=1 at P=±0.024. At this period length the batch size $[P \cdot D_2]^+$ is between 19 and 20. For the computation of the batch throughput time we use an integer valued number of items per batch. The increase in the batch size leads at this P value to a permanent increase in the required throughput time and a temporary increase in the required number of stages. Therefore, the discontinuity of TT_h as a function of P causes $N \geq [TT_h/P]^+$ to be no strictly non-increasing function of P. If the number of subbatches increases, this effect may again show up, due to an increase of rounding up errors.

[6] Figure 5.8 shows the differences between the actual cost curves in one graph. Note that it amplifies the differences through the selection of a different scale of the Y-axis.

compare this solution with the solution of *nb*=2, we see that the holding cost reduction (one stage less) is not enough to overcome the increase in transfer costs (two transfer batches per operation per period extra). Figure 5.8 summarizes the differences between the actual cost curves for the four batching strategies discussed. Note the different scale of the Y-axis compared with Figure 5.7.

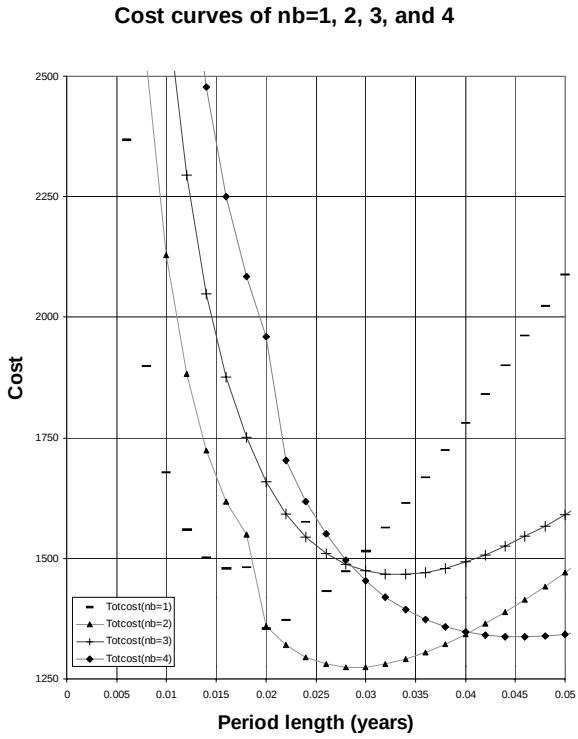


Figure 5.8 Change in actual cost curves for various equal number of subbatch policies

The effect of an increase in the number of subbatches on the required throughput time obeys the law of diminishing marginal returns. This effect can be seen from Figure 5.8 and is also known from research on related problems that we discuss in Appendix D, such as lot streaming and repetitive lot problems. The cost of an increase in the number of subbatches is linear with both the number of subbatches and the number of periods. Due to the law of diminishing returns, there will either be a number of subbatches for which the extra transfer costs exceeds the benefits of a further reduction of the throughput time (lower holding costs), or the minimal number of stages has been reached earlier. In the example, N=1 is reached for P=0.11 and *nb*=11, resulting in a minimal cost of 1467, which illustrates that a huge increase in the number of transfer batches is needed to further reduce the number of stages.

We conclude that an increase in P has an important effect on total costs, due to the reduced number of stages after a breakpoint. The number of subbatches has a strong impact on both throughput time and transfer costs. A solution approach should consider this complex relationship.

§ 5.5.2 Variable batching strategies: an enumerative search heuristic

The insight that we gained on the relationship between costs, period length, number of stages and subbatch strategy are based on the use of an equal number of subbatches for all operations and all products. Such a batching policy is very transparent and easy to use, but may result in reducing throughput times of products without any effect on the required number of stages, but with effect on costs. In fact, in a PBC system we should use overlapping production only for products with throughput times longer than the available time in a stage. As we not yet have defined the stages, we can relax this criterion to the use of a higher number of subbatches only if it helps to reduce the total manufacturing throughput time N·P and consequently helps to minimize total cost, according to our formulation of the equal subbatch model in Formula (8) or the variable subbatch model in Formula (11). The latter distinguishes subbatch strategies between the products as well as between the operations.

It may be difficult to find suitable values for the variables nb_{hi}, even for a restricted number of products *h*. There is no standard solution for the problem of determining values for nb_{hi} that minimizes total cost for a given period length P. Appendix D discusses several solution procedures from literature. For the general problem, we can only derive approximations of the optimal solutions. Such heuristic approaches may therefore even find solutions that have larger total costs than an equal number of subbatches strategy. We have experimented with the solution described in Graves and Kostreva (1986), but with their method for determining the number of batches (described in Appendix D), we obtained disappointing results for total costs. Hence, we decided to design a new heuristic.

Enumerative search heuristic

The *enumerative search heuristic* finds feasible values for nb_{hi} given a period length P. It starts with an initial (large) number of stages N, and an equal subbatch strategy where all products have only one subbatch (nb_{hi}=1). The enumerative search heuristic directs its attention to finding breakpoints in cost curves that depend on N and nb_{hi}, compares the solutions for these breakpoints, and selects the batching strategy that causes the lowest total cost. The heuristic determines an increase in the number of subbatches per operation only for operations for which it is allowed to find multiple subbatches, i.e., the user may provide the heuristic with a maximum number of subbatches at a specific operation. The increase in the number of subbatches at this operation is based on information with respect to the location of the breakpoints and the expected reduction in the total throughput time[7]. All products whose throughput times have to be shortened are considered. Figure 5.9 describes the enumerative search heuristic in detail.

[7] The estimated reduction in throughput time due to an increase in the number of subbatches at operation *i* depends in the heuristic on the ratio of processing time and number of machines for this and the successive operation *i*+1. If this ratio is smaller for operation *i*, we estimate the decrease of throughput time as one (resized) subbatch times the processing time at *i*. If the ratio is smaller for operation *i*+1, we take into account the effect it has on operation *i*+1.

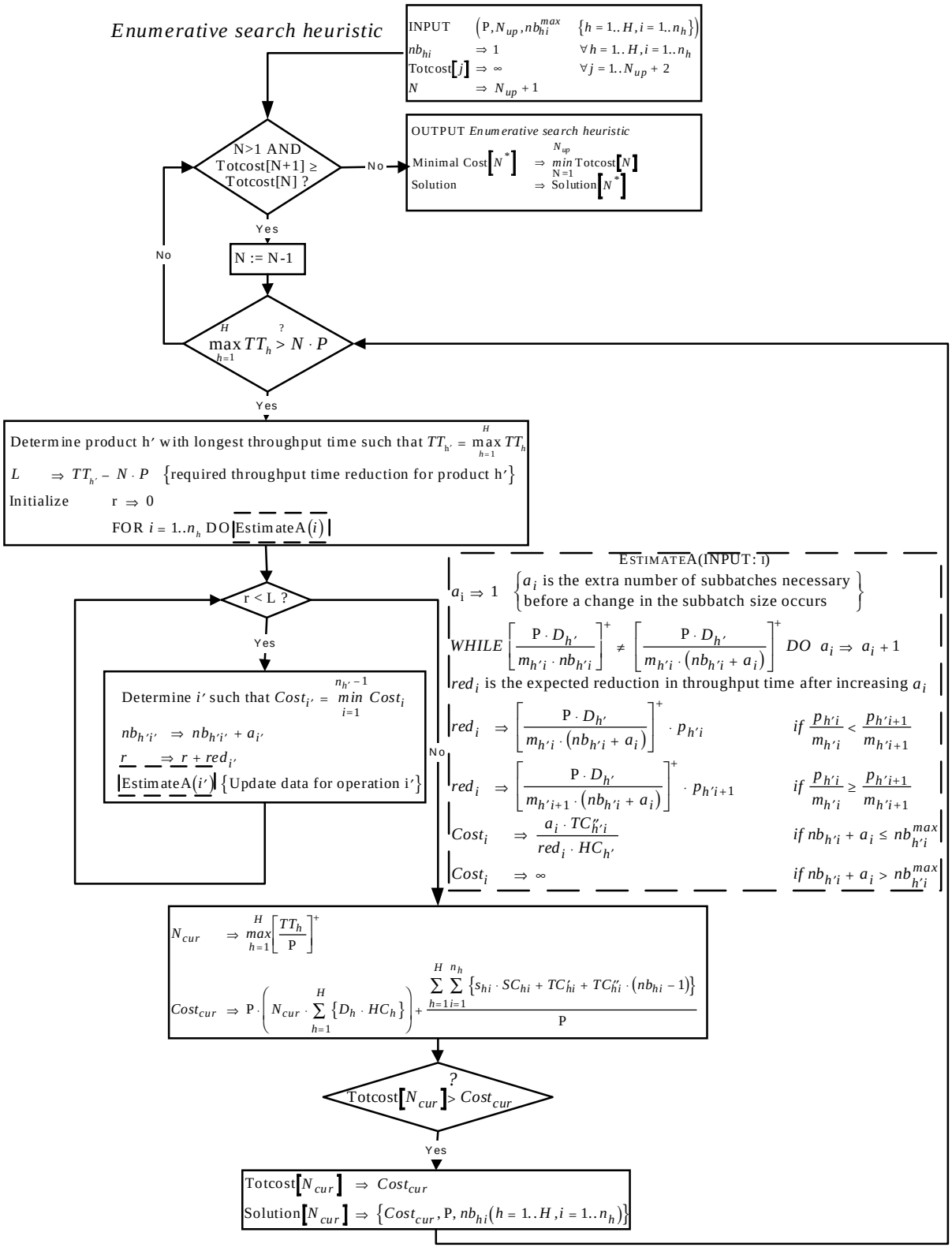


Figure 5.9 Enumerative search heuristic that finds P, N, and nb_{hi} with minimal cost

The increase in the number of subbatches for an operation causes extra transfer costs, but a throughput time reduction will eventually reduce holding costs. The ratio between both cost changes drives the selection of a suitable batching strategy. The enumerative search heuristic in itself does not prefer nested or non-nested batching strategies.

The enumerative search heuristic has been used to obtain solutions for P=0.002·X, X=1..50. The total cost curve for successive values of P in the example problem of Section § 5.5.1 is presented in Figure 5.10. Note that the solutions are approximations of the optimal solution, and the distance to this (unknown) optimal solution may vary for successive values of P. The symbols that we use in the figure to indicate a solution denote the number of stages N for which the minimum total cost was obtained. For small values of P, solutions with a higher number of stages are dominant, but if P increases, N decreases.

The best solution obtained is for P=0.044 years with N=2 and a total cost of 1237.5. The batching strategy is variable and not specifically nested. The values of nb_{hi} are presented as nb[h=1, i=1-9; h=2, i=1-8]. The best solution has nb[3,3,3,4,4,3,3,3,1; 3,3,3,4,3,3,4,1]. Note that we only have applied a maximum number of subbatches at the last operation of a product.

The solution found with the enumerative search heuristic is an improvement over the best solution found with an equal number of subbatches strategy. The latter was shown in Figure 5.8 (P=0.028, N=3, nb_{hi}=2 $\forall h,i$, total cost = 1274). The difference in total cost between both solution is only 3%, but the difference in period length is more than 50%, i.e. an increase from less than 8 days to more than 11 days!

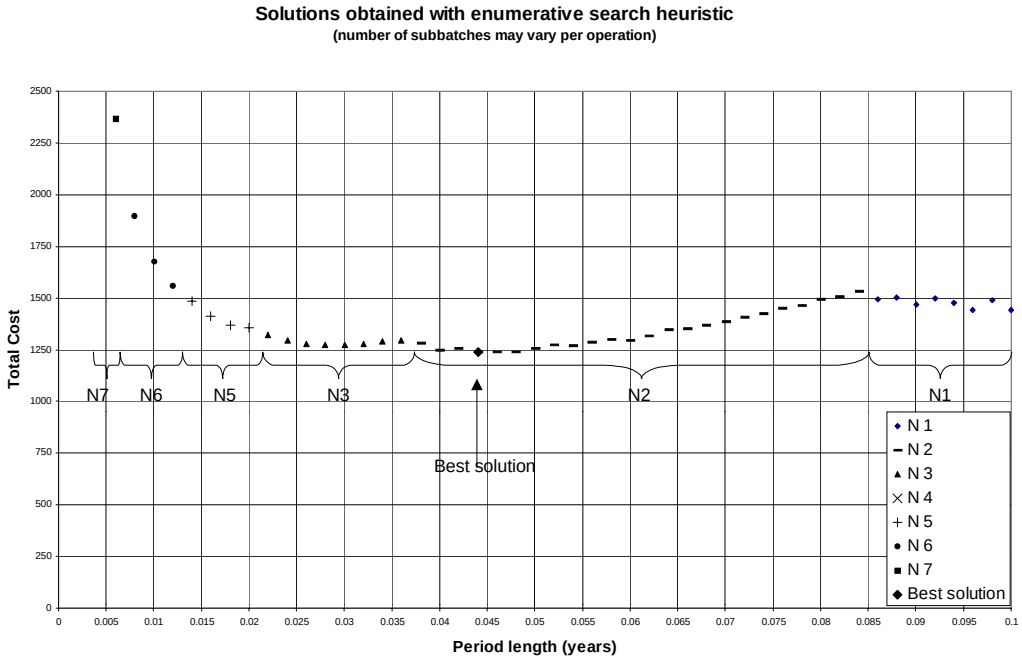


Figure 5.10 Cost of solutions obtained with enumerative search heuristic

The cost difference between an equal number of subbatches strategy and a variable batching strategy is not high for the example problem of Section § 5.5.1. Note however, that the characteristics of the example, such as identical processing times per operation of a product, almost equal sized product throughput times, few products, and equal transfer costs for all operations, lead to a quite positive impression of the performance of an equal number of subbatches strategy. The results of this strategy in terms of total costs would deteriorate quickly if one or more of these characteristics would change.

We can conclude from Figure 5.10 that (in the example problem) the total cost function is not very sensitive to an increase in the period length, as long as a variable batching strategy is allowed and P is not too small. For small period lengths, the total cost function is very sensitive to a change of P. Therefore, the determination of the number of subbatches per operation should get careful attention in a solution approach for configuring a PBC system.

§ 5.5.3 Solution approach for the complete model: progressive search heuristic

The enumerative search heuristic finds an approximate solution for the model (11) only for a given period length P, which is input to the heuristic. The solution approach does therefore not treat the complete model, which includes the determination of a period length P. From the discussion on related problems in Appendix D and on the cost structure of the model in Section § 5.5.1, we know that trying to find an optimal solution for the complete model is not realistic. Restricting the search to an equal number of subbatches strategy may result in a good approximation, but this depends strongly on the characteristics of the input to the model. Applying an enumerative search for all possible values of P is computationally unattractive, inefficient, and also not necessary as the total cost curve is quite flat over a broad range of P-values. Hence, we strive to develop a solution approach that finds both a period length P, a number of stages N, and a batching strategy $nb_{\mathrm{hi}}\ \forall\ h,i$. This (heuristic) solution approach should have a performance which is less sensitive to changes in the input parameters, and which is less time consuming than complete enumeration.

Progressive search heuristic

The *progressive search heuristic* finds a solution for the complete model (11) of determining P and nb_{hi}. It calculates the required number of stages N and the resulting costs from the value of these decision variables. The heuristic sets the minimum period length and maximum number of stages at initial values that take the available capacity into account. It applies the enumerative search heuristic for specific values of P in order to find suitable values for nb_{hi}. These values of nb_{hi} are used to determine a better approximation of the best value for period length P. The heuristic iterates between these two main parts until no further improvements are made, as is illustrated in Figure 5.11. We describe the heuristic in detail in Appendix E. There we will also pay attention to the application of the heuristic on the example problem.

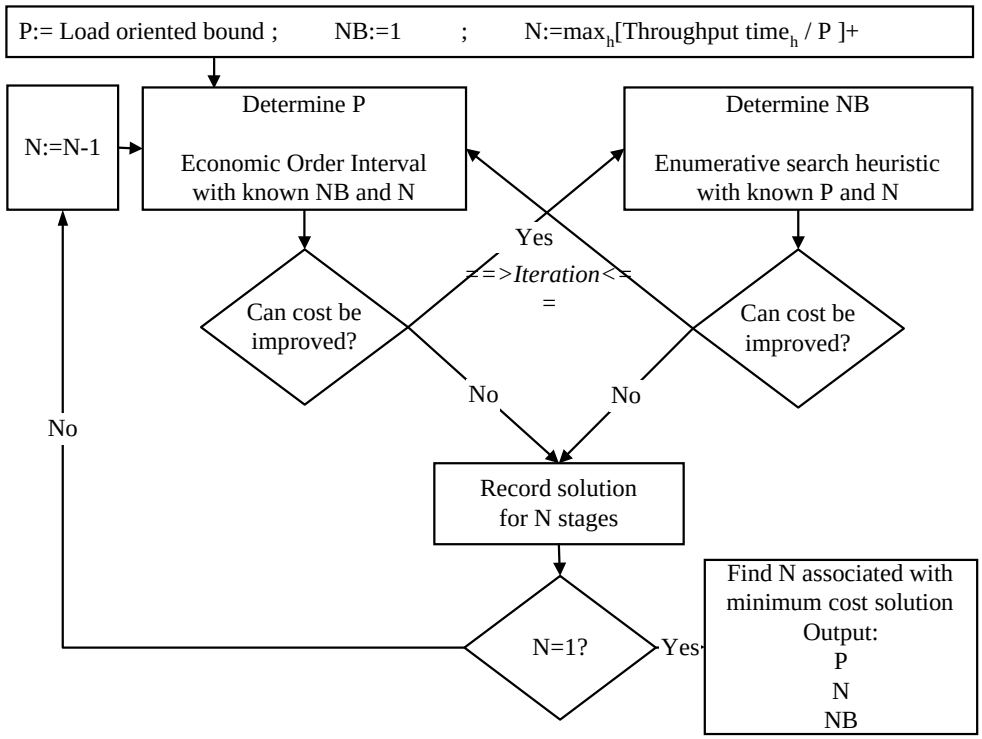


Figure 5.11 Structure of progressive search heuristic (details in Appendix D)

Remark that the progressive search heuristics uses an economic order interval determination approach to determine the optimal period length for given values of N and NB. The proposed period length will probably be higher than the one used in determining the batching policy NB and number of stages N that served as input. Therefore, the new P might increase the throughput time TT_h. If we observe such an increase, we will try to find a larger number of subbatches at a critical operation. The increase in the number of subbatches leads to an increase in the optimal period length, as we know from our analysis in Figure 5.7a. We have to return to our period length determination model in order to accomplish for this tendency. The iteration results in a continuously increasing number of subbatches for increasing period lengths. However, it only decides on a further increase of either of the decision variables if it results in lower expected costs.

§ 5.6 Comparing the heuristics: an exhaustive search heuristic

We have applied both heuristics to find a solution for the example problem of Section § 5.5.1. Table 5.3 presents the results of both heuristics. We see that the progressive search heuristic finds a very good approximation of the best solution found in the enumerative search heuristic. The efficiency of the progressive search heuristic was much better, as it took considerably less time to find this solution. The question rises if this result is specific for the example problem. Therefore, we want to compare the performance of both heuristics more extensively. Generally, we would like to use an optimal solution as a benchmark criterion. However, we have no technique available to find such an optimal solution for the problems we solve, hence we developed an improved heuristic solution procedure, denoted as the *exhaustive search heuristic*.

	progressive search heuristic	*enumerative search heuristic*
Total cost	1255	1237.5
N	2	2
nb_{hi}	(4,4,3,3,3,3,3,3,1; 4,4,4,4,3,3,3,1)	(3,3,3,4,4,3,3,3,1; 3,3,3,4,3,3,4,1)
P	0.04263	0.044

Table 5.3 Solutions of heuristics for example problem

The *exhaustive search heuristic* performs a very comprehensive search for a variable subbatch strategy for the values of P found by respectively the enumerative search and the progressive search heuristic. After each increase in a nb_{hi}, i.e. one of the operations is divided into one more subbatch, the *exhaustive* search heuristic *recalculates* the effect on the expected throughput time. Note that the enumerative and progressive search heuristics both *estimate* the effect on the throughput time, as a recalculation is very time consuming. The resulting solution consists of the same period length as at least one of the other heuristics had proposed, but generally a different batching strategy. The cost of this solution is used as a benchmark for the enumerative and progressive search heuristic.

Performance of heuristics

We constructed 200 problems that were solved by the three heuristics. Each heuristic finds a period length P, a number of stages N, a subbatch strategy NB, and the associated total costs. The data from the illustrative example of § 5.5.1 was used as input for the random procedure that determines values for set-up and processing times of the operations. Processing times were negative exponentially distributed with means as described in § 5.5.1. Set-up times were normally distributed with means as described (resp. 15 and 12 hours per set-up) and standard deviations equal to ½ its mean. The load oriented lowerbound for P had to be less than four weeks.

We first applied the progressive search heuristic, starting with P identical to the load oriented lowerbound and NB equal to 1.

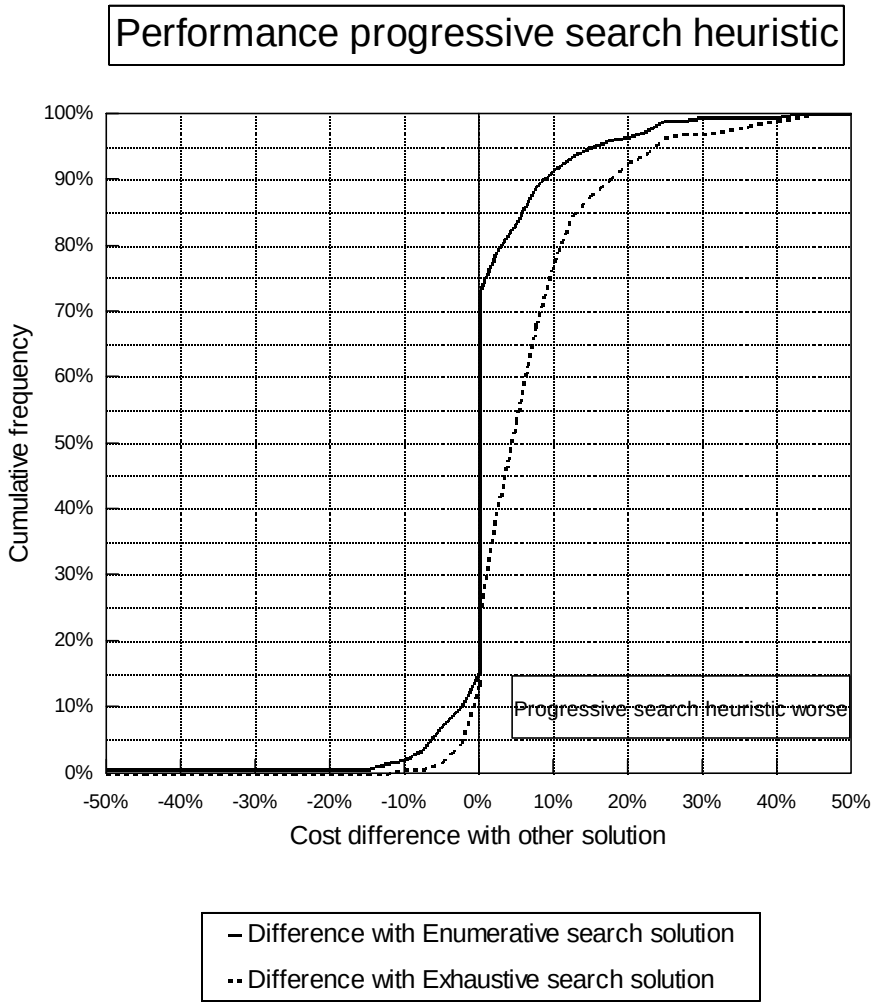


Figure 5.12 Performance progressive search heuristic

Cost difference of progressive search to other solution	Enumerative	Enumerative Cumulative	Exhaustive	Exhaustive Cumulative
x ≤ -7½ %	3.5%	3.5%	0.5%	0.5%
-7½ % < x ≤ -5 %	3.5%	7.0%	1.0%	1.5%
-5 % < x ≤ -2½ %	3.0%	10.0%	3.0%	4.5%
-2½ % < x < 0 %	5.5%	15.5%	9.0%	13.5%
x = 0 %	57.5%	73.0%	10.5%	24.0%
0 % < x ≤ 2½ %	6.5%	79.5%	16.5%	40.5%
2½ % < x ≤ 5 %	4.0%	83.5%	13.0%	53.5%
5 % < x ≤ 7½ %	5.5%	89.0%	14.5%	68.0%
7½ % < x ≤ 10 %	2.5%	91.5%	9.5%	77.5%
x > 10 %	8.5%	100.0%	22.5%	100.0%

Table 5.4 Performance of progressive search heuristic

Next, we applied the enumerative search heuristic. We considered a broad range of possible values for the period length P. The centre of the range was the P value such as determined by the progressive search heuristic. The lower range value was 90% below this value, with a minimum at the load oriented lowerbound. We chose the range to be symmetrical around the progressive search solution, so the upper range level could be calculated. The enumerative search heuristic was applied 80 times at equally distanced P values. For each P, it determined a suitable batching strategy, a number of stages N, and the total costs of this configuration.

Finally, we applied the exhaustive search for both values of P found.

The results of the experiments are presented both in Figure 5.12 and in Table 5.4. Figure 5.12 shows the cumulative distribution functions of the performance of the progressive search heuristic in terms of cost difference with either the enumerative search heuristic or the exhaustive search heuristic. The function that describes the difference with the enumerative search heuristic shows that 73% of the solutions of the progressive search heuristic had an identical or even better cost performance than the solutions of the enumerative search heuristic.

Table 5.4 shows that 15.5% of the progressive search solutions had lower cost than the enumerative search solutions. Remark that the progressive search heuristic internally applies the enumerative search heuristic in order to improve the batching strategy that it uses! 57.5% of the solutions were identical and only 8.5% of the solutions had a total cost that was more than 10% above the cost found by the enumerative search heuristic.

The required CPU time for computing a solution was much higher for the enumerative search heuristic. The computations for the experiments showed a factor 60 difference. A solution for the progressive search heuristic was found within the smallest time measure of our computer clock (0.054 seconds). The required CPU time for the enumerative search depends on the number of steps (predetermined at 80 in our experimental design). Both the quality of the solution found and the CPU time increase with the number of steps in the heuristic. The quality of the enumerative search solution may also improve if a more precise search range is applied.

The solution of the exhaustive search heuristic was meant to function as an indication of the optimal solution. We see from Figure 5.12 that the performance of the progressive search heuristic with respect to this solution is considerably smaller. 76% of the solutions found by the progressive search heuristic were improved. 22.5% of the solutions had a total cost that was more than 10% above the cost found by the exhaustive search heuristic.

However, a remarkable fact can be seen. In 13.5% of the experiments, the progressive search heuristic performed better than the exhaustive search heuristic and 10.5% of the cases the result could not be improved. The explanation for this behaviour is that the exhaustive search is a greedy algorithm. It applies a depth first search in the solution tree and computes the

effect on the throughput time after each modification in the subbatch strategy. However, the heuristic will never trace back in the solution tree: an increase in the number of subbatches for a specific operation early in the search process will not be reconsidered later in the search process of the heuristic. A complete enumeration of the solution tree (i.e. of all possible subbatch strategies) would identify the optimal solution, but such a search is unrealistic. We conclude that the exhaustive search heuristic has selected in 13.5% of the cases a branch of the solution tree that seemed profitable but proofed to be less fortunate and successful.

Further study revealed that the cost reduction obtained by the exhaustive search heuristic comes from a different subbatch strategy. In 92% of the experiments, the exhaustive search heuristic chose the same period length as the progressive search heuristic and only 8% resulted in a higher period length. The enumerative search heuristic chose in 17% of the experiments a higher period length and 83% of the cases resulted in an identical advice on the period length. The difference between these 8% and 17% leads to the conclusion that in case of a different advise of the two heuristics the exhaustive search heuristic chooses in more than 50% of the cases for the period length advised by the progressive search heuristic.

We conclude that the progressive search heuristic shows a high performance compared to the time consuming enumerative search heuristic. The distance to the lowest cost solution found by the exhaustive search heuristic can be considerable, but in 92% of the cases the period length that was found by the progressive search heuristic remains unchanged in the best solution found. Reconsidering the subbatch strategy therefore will improve the performance of the progressive search heuristic with respect to cost. The advice on the period length P and the number of stages N can be considered good.

§ 5.7 Conclusions

We have developed a new approach for the determination of a suitable configuration of a PBC system. This configuration consists of a period length P, a number of stages N, and a variable subbatch strategy. The mathematical model that masters this approach allows for overlapping operations in determining P. The formulation of the minimal period length in our model is more exact than existing modelling approaches that include overlapping operations, which are all based on the initial work of Szendrovits (1975) (see Appendix D).

First, we developed a detailed decoupled period determination approach, which assumes that the allocation of operations to the stages and machines is known in advance. This approach yields some general restrictions on the length of P, given a number of stages N.

We have integrated the detailed decoupled period determination approach with an economic approach to period determination. This integrated approach is denoted as a detailed economic period determination approach. We distinguish echelon holding costs, set-up costs and

transfer costs in our integral model and apply an economical cost perspective in determining a period length and a transfer batch strategy. Hence, a trade-off is made between manufacturing throughput time (holding costs) and frequency of replanning (set-up and transfer costs).

The integrated approach relaxes the assumption that the allocation of operations to stages is known in advance. This causes the number of stages N to become a model variable instead of a parameter: N is a direct consequence of the maximum throughput time that results from the chosen period length and subbatch strategy. The approach has been worked out into two mathematical models: a model with an identical number of subbatches per product and a model that allows the number of subbatches per operation to vary.

Using the same number of subbatches for all operations and all products results in a system that is very transparent and easy to use, but may also lead to unnecessary reducing throughput times of products without any effect on the required number of stages. In a PBC system, overlapping production should be used only for products with throughput times longer than the available time in a stage. As we not have defined the contents of the stages in these mathematical models, we apply a variable number of subbatches only if it helps to reduce the total manufacturing throughput time N·P and hence helps to minimize total cost.

Variable subbatch policies can be distinguished in non-nested, nested and powered nested policies. Non-nested policies result in a smaller number of subbatches at a following operation. This may lead to extra product waiting time compared with nested policies. Nested policies may lead to extra intermediate machine waiting time.

We developed two search heuristics that are able to determine a variable batching strategy, the enumerative and progressive search heuristic. The enumerative search heuristic finds a suitable subbatch strategy for a given period length P. Tests with the enumerative search heuristic showed that a varying number of subbatches strategy produces a rather flat total cost function for various values of the period length. This indicates that as long as one uses a high quality search algorithm for the various numbers of subbatches per operation, the total cost is not very sensitive for changes in the period length. We also showed that if the (allowed maximum) number of subbatches increases, the optimal period length tends to increase.

The progressive search heuristic finds a solution both for the period length P and the various numbers of subbatches nb_{hi}. The progressive search heuristic was shown to produce a good approximation of the best solution that we could find for a large set of problems. Especially the quality of the period length determination was high.

Do these results mean that we can apply the progressive search heuristic in PBC system design? In order to answer this question, we should reconsider the various assumptions that we made in the underlying model. First, we assumed that costs can be decomposed into echelon holding costs, set-up costs, and transfer costs. The model enables a trade-off between manufacturing throughput time (holding costs) and frequency of replanning (set-up and

transfer costs). Both period length and subbatch strategy strongly influence these costs. However, the effect of variety and uncertainty in volume and mix on these costs is ignored.

Furthermore, we assumed that the allocation of operations to the stages was not known when we determined the period length, batching strategy, and the number of stages. This approach results in multi-phased PBC systems, as the moment of release of work to the stages is not synchronous. We need a better understanding of the relationship between the PBC design factors in single-stage systems before applying the heuristic to the design of a PBC system. Chapter Six will determine the effect of stochastic fluctuations on PBC performance. It uses simulation to evaluate this performance. Chapter Seven will examine the applicability of the progressive search heuristic for finding a PBC configuration for the simulated situations.