Grouping of customer orders into manufacturing batches

Nico Vandaele and Joachim Joye

KBI 0908

Grouping of Customer Orders into Manufacturing Batches

Nico Vandaele
Katholieke Universiteit Leuven (Faculty of Business and Economics, Naamsestraat 69, 3000 Leuven) and Katholieke Universiteit Leuven Campus Kortrijk (E. Sabbelaan 53 B-8500 Kortrijk), Nico.Vandaele@econ.kuleuven.be

Joachim Joye
Katholieke Universiteit Leuven Campus Kortrijk (E. Sabbelaan 53 B-8500 Kortrijk), Joachim.Joye@kuleuven-kortrijk.be

The formation of manufacturing batches, given arriving orders, is a general problem in practice. In essence, it constitutes a optimization problem. The goal for this problem however, may differ from perspective. Most commonly this problem is viewed from a cost-based perspective, where minimizing costs is the goal. Other views can include minimizing overall batch lead times or the deviation from an optimal batch size derived from an operational performance perspective. This paper presents and compares these different approaches to this problem. Starting from a basic model, extensions will be proposed that incorporate operational performance measures as well. Both an integer programming model, and an dynamic programming model are discussed.

Key words: lot sizing, manufacturing batches, dynamic programming, grouping problem

1. Introduction

Any real-life production environment faces stochastic processes on any level. The typical stochastic sources are demand and service processes. This variability tends to be positively correlated with product variety and process flexibility. A typical job-shop environment faces both a large variety of products, with complex routing possibilities, and a high level of flexibility. However, the impact on efficiency can be devastating.

Operational mathematical models exist which determine an optimal value for a controllable variable, which is the result of minimizing or maximizing certain operational performance measures. Typically, this controllable variable is the lot size. Commonly used operational performance measures are lead times, WIP, waiting times, throughput. In literature, a large variety of lot sizing models and algorithms can be found, see Karimi et al. (2003) or Buschkühl et al. (2008). Incorporating operational based models can greatly improve performance, and lower related costs, which can be observed in Vandaele et al. (2000).

The starting point for this work will be the mathematical model used in Lambrecht et al. (1998), where a manufacturing system is modeled on the aggregate level using a queueing network. The resulting parameters of this aggregate model will be used as input for the different short term batching approaches. Figure 1 illustrates this concept, where aggregate planning (stochastic lot sizing) and detailed scheduling (order grouping into manufacturing batches) interact.

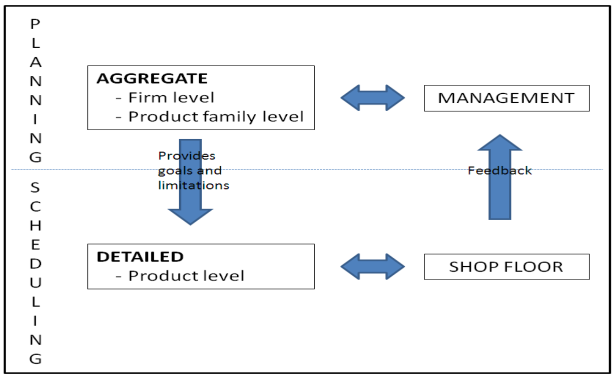


Figure 1: General Approach

First, an IP based approach will be formulated. Secondly, a dynamic programming approach is discussed. Extensions can be made, and intelligent design is mandatory to simplify and enhance the computational aspects of the algorithm.

In a last section, these approaches are compared and results are presented.

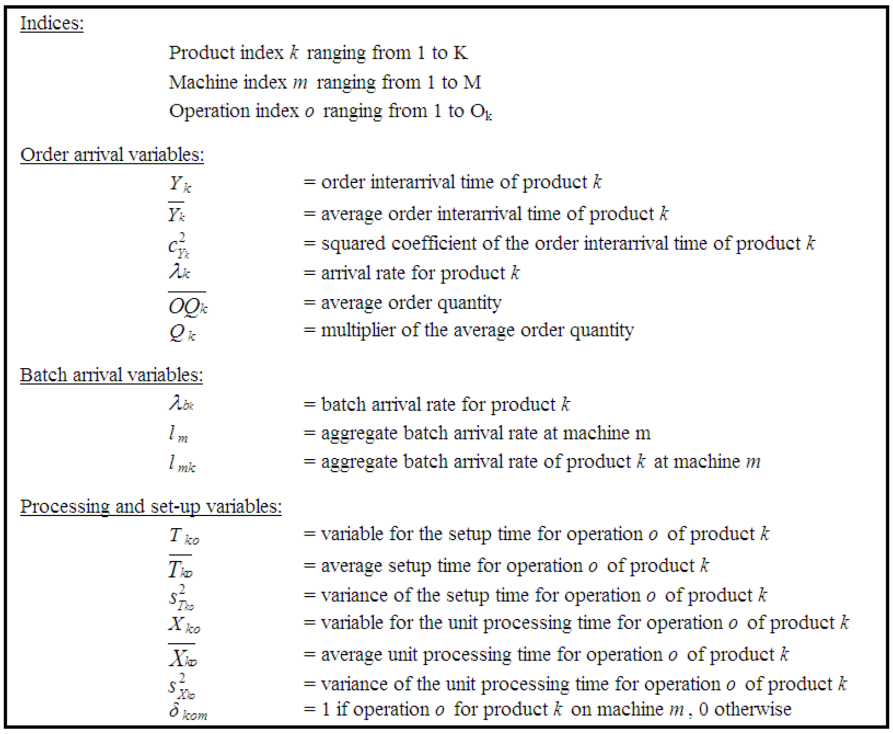
2. Modeling the manufacturing environment

The setting is a multi-operation job shop, where a make-to-order policy dictates production. The model described hereafter is this proposed in Lambrecht et al. (1998).

This highly dynamic environment faces variability in different areas. Being an order driven environment, where individual customers place orders for a large variety of products independently from each other, the resulting interarrival process is stochastic. These individual orders are aggregated on a product level. Furthermore, each individual order for a given product requires processing through different machines, following the routing as described in the bill of processes. Multiple routings are possible for all products. On a machine level, both processing and setup times are assumed stochastic and sometimes subject to failures, quality problems and other disruptions.

In the queueing model proposed in Lambrecht et al. (1998) all relevant parameters are written as a function of the lot size. A resulting optimal lot size can be found, which minimizes the average product lead time. The main advantage of this approach lies in the incorporation of the convex relationship between lot size and average lead time as described by Karmarkar (1987).

The required input parameters for the model are demand, service times and shop parameters. For each product, an aggregate arrival process is characterized by expected demand and the distribution parameters of the order interarrival time. This can either be an estimation based on demand forecast or confirmed orders. See table 1 for the definition of the operational variables.



Indices:	
Product index k ranging from 1 to K	
Machine index m ranging from 1 to M	
Operation index o ranging from 1 to O_k	
Order arrival variables:	
Y_k	= order interarrival time of product k
$\overline{Y_k}$	= average order interarrival time of product k
$c^2_{Y_k}$	= squared coefficient of the order interarrival time of product k
λ_k	= arrival rate for product k
$\overline{OQ_k}$	= average order quantity
Q_k	= multiplier of the average order quantity
Batch arrival variables:	
λ_{bk}	= batch arrival rate for product k
l_m	= aggregate batch arrival rate at machine m
l_{mk}	= aggregate batch arrival rate of product k at machine m
Processing and set-up variables:	
T_{ko}	= variable for the setup time for operation o of product k
$\overline{T_{ko}}$	= average setup time for operation o of product k
$s^2_{T_{ko}}$	= variance of the setup time for operation o of product k
X_{ko}	= variable for the unit processing time for operation o of product k
$\overline{X_{ko}}$	= average unit processing time for operation o of product k
$s^2_{X_{ko}}$	= variance of the unit processing time for operation o of product k
δ_{kom}	= 1 if operation o for product k on machine m, 0 otherwise

Table 1: Operational Variables

For every product *k*, individual orders are aggregated in the arrival rate λ_k. These are transformed into batches of size $\overline{OQ_k}$ *Q_k, which have an aggregate batch arrival rate of λ_{bk}. For the first machine on the routing of product k, λ_{bk} equals λ_k / Q_k,.

Each machine *m* has to process batches from various products, resulting in an aggregate batch arrival rate l_m, equaling $\sum_{k=1}^{K}\sum_{o=1}^{O_k}\lambda_{bk}\delta_{kom}$. This can be seen as the sum of the individual batch arrival rates for all products *k* that are being processed on machine *m*. For the resulting variance, we refer to Lambrecht et al. (1998).

When batches are ready for production, they face both setup and processing times. For each machine m, an aggregate batch processing time $1/\mu_m$ can be defined as

$$\frac{1}{\mu_m}=\sum_{k=1}^{K}\frac{l_{mk}}{l_m}\sum_{o=1}^{O_k}\frac{\lambda_{bk}\delta_{kom}}{l_{mk}}(\overline{T_{ko}}+Q_k\overline{OQ_k}\,\overline{X_{ko}}),$$

where $\frac{l_{mk}}{l_m}$ is the probability that product in front of machine *m* is of product type *k*, and $\frac{\lambda_{bk}\delta_{kom}}{l_{mk}}$ is the weighted average of the operations *o* on machine *m* for the same product *k*. For the resulting variance, we refer to Lambrecht et al. (1998).

An adapted traffic intensity ρ' is proposed, which includes utilization due to both processing and setups. For each machine *m*, this equals to

$$\rho'_m=\frac{l_m}{\mu_m}=\sum_{k=1}^{K}\sum_{o=1}^{O_k}\lambda_{bk}\delta_{kom}(\overline{T_{ko}}+Q_k\overline{OQ_k}\,\overline{X_{ko}}).$$

As batches of different products *k* arrive at a given machine *m*, a queue is formed, resulting in waiting times. This adapted traffic intensity is used in determining $E(Wq_m)$, the expected value of the aggregate waiting time in front of machine *m*. The aggregate waiting time $E(Wq_m)$ is approximated using the Kraemer and Lagenbach-Belz approximation (1976), although any good approximation can easily be accommodated.

Note that on an aggregate level a critical lot size can be found that produces infinite expected waiting times. This Q^k_{crit} is the lowest possible value for the lot size of a given product *k*, and equals the asymptotic left side of the expected lead time function. As batch sizes become smaller, and l_m increases, the adapted traffic intensity ρ' raises, but cannot exceed 1. Q^k_{crit} provides a lower bound for this problem.

The resulting objective function of the aggregate expected lead time in function of the lot size becomes

$$E(W) = \sum_{m=1}^{M} E(Wq_m) + \sum_{k=1}^{K} \frac{\lambda_k \overline{OQ_k}}{\sum\limits_{k=1}^{K} \lambda_k \overline{OQ_k}} \frac{(Q_k \overline{OQ_k} - 1)\overline{Y_k}}{2\overline{OQ_k}} + \sum_{m=1}^{M}\sum_{k=1}^{K} \frac{\sum\limits_{o=1}^{O_k} \lambda_k \overline{OQ_k} \delta_{kom}}{\sum\limits_{k=1}^{K}\sum\limits_{o=1}^{O_k} \lambda_k \overline{OQ_k} \delta_{kom}} \left(\sum_{o=1}^{O_k} \frac{\lambda_k \overline{OQ_k} \delta_{kom}}{\sum\limits_{o=1}^{O_k} \lambda_k \overline{OQ_k} \delta_{kom}} (\overline{T_{ko}} + Q_k \overline{OQ_k}\,\overline{X_{ko}}) \right) \tag{1}$$

being the sum of aggregate waiting time, waiting to batch time and setup and processing time.

Note that the weight $\dfrac{\lambda_k \overline{OQ_k}}{\sum\limits_{k=1}^{K} \lambda_k \overline{OQ_k}}$ demonstrates the relative importance of product *k* over all products.

An optimal lot size Q_k^* can be found, which minimizes *E(W)*. The minimization problem involves a non-linear objective function and constraints, and can be solved using a dedicated optimization routine as described by Vandaele (1996).

A lognormal distribution is assumed for the lead time distribution, which is used to derive lead time percentiles for the corresponding customer service levels.

The main focus here lies with the optimal batch size Q_k^* produced by the model. This parameter is the longer term aggregate benchmark for grouping orders into manufacturing batches. Within any real production environment however, orders do not demonstrate this average behavior, but rather vary in size and due dates, resulting in deviations from this optimal batch size. These deviations can be optimized, using different methods, and for different standards. From a financial perspective, one can look to minimize related costs. From an operational perspective, one can look to minimize total expected lead times. In addition, the formation of manufacturing batches is a key input parameter for any short term scheduling system.

3. An IP approach

In this section, an integer programming approach will be used to produce an optimal solution for the grouping problem. Starting from a description of the problem, a basic IP formulation will be given. This general formulation focuses mainly on minimizing the total number of inventory-days. The result is a basic formulation, which can be extended to incorporate more information. Related costs will be briefly discussed. Extensions, which incorporate expected lead times, will be presented.

Let N denote the number of orders that have to be grouped into manufacturing batches. Each of these orders has a given due date DD_{kn}, and a given customer order quantity OQ_{kn}. Another important input parameter is the required number of setups S_k. This parameter can be derived from the results obtained from the queueing model previously described.

$$S_k = \left\lfloor \frac{1}{Q_k^*} \sum_{n=1}^{N} OQ_{kn} \right\rfloor \qquad (2)$$

Rounding down to the closest lower integer value eliminates infeasible solutions. Table 2 summarizes the calculations using the metal shop example. Detailed customer order information underlying the parameters in table 2 can be found in table 3, section 5.

	Product P	Product S
Sum of Customer Order Quantities	15	30
Optimal Batch Size Q*	4	6
Real Value of Fraction	3,75	5
Rounding down = S_k	**3**	**5**

Table 2: Required number of setups for the metal shop.

An inventory-day matrix of size (N x N) can be calculated. Note that when speaking of days, this can be generalized towards any usable time unit. Let c_{ij} denote the number of inventory-days that results from grouping order j with order i (i ≤ j), so that

$$c_{ij} = (DD_j - DD_i) \,.\, OQ_j \qquad (3)$$

Grouping orders takes place on a product level, so that the index k can be omitted from (3). Clearly, when i equals j, c_{ij} will be 0, as the batch will be completed on its due date, causing no inventory.

This inventory-day matrix is symmetrical, all calculated values can be transposed, so that c_{ij} equals c_{ji} for all sets of (i,j). The objective is to choose these values of i and j which minimize c_{ij}, given a fixed number of batches, equalling S_k. A decision variable matrix x_{ij} of size (n x n) is formulated.

The objective function becomes:

$$Min \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij}\, x_{ij} \qquad (4)$$

with all $x_{ij} \in \{0,1\}$.

A number of constraints have to be added. A value of 1 for x_{ij} translates that order i will be ready at due date j. The starting point for every manufacturing batch will be a diagonal element of the decision matrix x_{ij}. Hence, a first constraint will be a restriction to equal the number of diagonal elements to S_k.

$$\sum_{i=1}^{n} x_{ii} = S_k \qquad (5)$$

Every order can only be batched once, resulting in a second constraint.

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall\ i = 1,....,n \qquad (6)$$

The remaining constraints ensure that orders have been grouped into manufacturing batches correctly, this is, chronology is assured. Because the inventory-day matrix c_{ij} is symmetric by construction, only one half is needed. For these constraints, we choose the lower half as basis for the formulation, while the elements above the diagonal elements are being equaled to 0, resulting in a lower triangular matrix. An equivalent construction can be formulated when solving the problem, resulting in an upper triangular matrix.
Furthermore, an order can only be assigned to a certain batch if the diagonal element x_{ii} equals 1. This results in

$$\begin{aligned} &\sum_{j=1}^{n} x_{ij} = 0 \qquad \forall j > i \qquad \forall i, j = 1,...,n \\ &x_{ij} \leq x_{jj} \qquad \forall i, j = 1,...,n \end{aligned} \qquad (7)$$

The result of this basic formulation is a minimization problem, delivering the lowest possible object value, given the order parameters and the number of batches required.

Related cost levels will not influence the grouping problem. The number of setups determines the level of fixed costs, which are equal for all feasible solutions for each product *k*. Any feasible solution will have a total number of S_k batches, resulting in S_k*C_{sk} for the fixed setup costs, however, C_{sk} will differ between products *k*. Variable costs (C_{hk}) such as inventory holding costs also do not influence the problem at hand, when assuming a linear function. The IP problem which minimizes $C_{hk}*(c_{ij}x_{ij}) + S_k*C_{sk}$ will forward an identical optimal solution for the IP problem which minimizes $(c_{ij}x_{ij})$, when C_{hk}, C_{sk} and S_k are considered fixed or linear for every product *k*. This conclusion does not hold when assuming a non-linear function for the inventory holding costs. Other functions, such as step-wise increasing variable cost function cannot readily be solved using IP.

Because of this cost-independency, the resulting value of the object function can be viewed as the amount of days needed to deplete all manufacturing batches. This optimal solution yields the lowest cost level, and minimizes stock cycle time.

Although the optimal batch size produced by the queueing network is incorporated into this approach by means of S_k, any deviation from this is not taken into account. No guarantees can be made to ensure the manufacturing batch sizes produced by this approach have a minimal deviation from S_k. The resulting average value of the manufacturing batch size will be close or equal to Q^*, however, deviations can be substantial. Small batches will be equaled out by large batches, but deviation is not punished, the resulting variance can be large.
At an aggregate level, this approach can even produce infeasible batch sizes, which violates Q^k_{crit}, being the lowest possible batch size at an aggregate level.

This problem can be addressed by incorporating expected lead times for every product k in the formulation. Choosing a batch size which deviates greatly from Q^*_k will result in longer expected lead times. To ensure identical dimensions, the extension must be formulated in inventory-days. This can be observed in figure 2. Here the progress of a manufacturing batch, which combines four orders, while facing three operations can be seen. Note that the black surfaces represent expected waiting times prior to any operation.

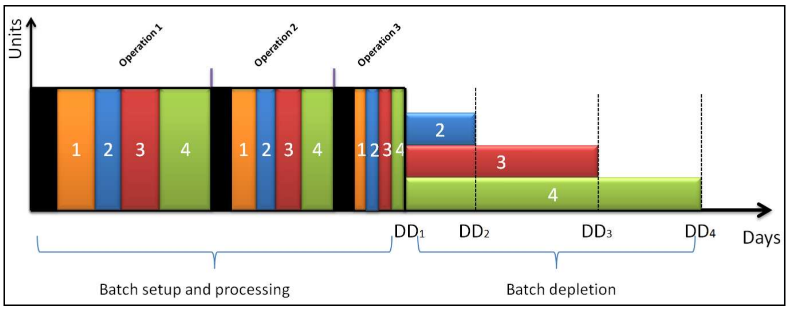


Figure 2: Processing and depletion of a manufacturing batch

As the dimension is inventory-days, the observed value equals the surface of the above figure, which multiplies days by units. This can be divided into two separate sections. The batch setup and processing section constitutes of the rectangle in figure 1. This surface is the size of the manufacturing batch times the lead time of the manufacturing batch. The batch depletion equals $c_{i,j}$, the total number of inventory-days needed to deplete a manufacturing batch of size $\sum_{n=i}^{j} OQ_{kn}$, given the corresponding due dates of the orders.

Incorporating the lead time poses different problems. The main problem can be found when observing the aggregate expected lead time, where a convex relationship can be seen, as described by Lambrecht and Vandaele (1996). This non-linear function cannot be dealt when using an IP approach. The problem, however, is quite different when observing the function on a product level, given expected waiting time restrictions. Let the expected lead time for a manufacturing batch of size $\sum_{n=i}^{j} OQ_{kn}$, which groups orders i through j, be written as

$$E(W_k)_{ij} = \sum_{o=1}^{O_k} E(Wq_m(Q^*))\delta_{kom} + \overline{T_{ko}} + (\sum_{n=i}^{j} OQ_{kn})\overline{X_{ko}} \qquad (8)$$

For every operation *o* in the routing of product *k*, the sum is made of the waiting time, setup time and processing time. A first extension will use the aggregate optimal lot size Q^* in regard with the aggregate expected waiting time, rather than the actual batch size $\sum_{i=1}^{j} OQ_{kn}$. A constant term will be added for all manufacturing batch sizes.

Using this constant expected waiting time for all batch sizes, a linearly evolving function can be approximated. Incorporating this additional parameter primarily discourages the use of small manufacturing batch sizes, or encourages the use of manufacturing batch sizes equal to or higher than the average value Q^*. As the number of batches S_k is a hard constraint, one-sided elimination of extreme manufacturing batch sizes (either low or high, as compared to the aggregate optimal lot size Q^*) will result in a decreased manufacturing batch size variance. The formulation is expressed in the main decision parameters of the IP formulation, i and j. Using this formulation, a nearly linear evolution can be observed for the batch setup and processing surface, equaling $(\sum_{n=i}^{j} OQ_{kn}) * E(W_k)_{ij}$, expressed in inventory-days.

The new objective function becomes

$$Min \sum_{i=1}^{N} \sum_{j=1}^{N} [(DD_j - DD_i)\ .\ OQ_j + (\sum_{n=i}^{j} OQ_{kn}) * E(W_k)_{ij}]\ x_{ij} \qquad (9)$$

Although $E(W_k)_{ij}$ is an additional input parameter, it varies with values of i and j, and can be found by using the model described in section 1. The constraints remain identical as in the basic formulation. It has to be noted that (9) again equals a matrix of size (N x N). It is the sum of two different matrices of sizes (N x N), first the batch depletion matrix c_{ij} from the basic IP formulation and secondly the average lead time matrix.

This extension produces the total amount of inventory-days that are needed to setup, process and deplete all customer orders, giving a fixed number of manufacturing. Using this extended formulation results in an improved solution for the grouping problem, as the total manufacturing batch size variance is reduced. This lower variance translates as a reduction in total aggregate lead time for all orders, given that $Q*$ minimizes expected lead time. Results will be discussed in section 5.

A second extension on the basic model uses varying expected waiting times rather than a constant term, as in the first extension. As this would result in a non-linear function, IP cannot be used. The main problem when using IP formulations remains, the impracticability of using parameters that display a non linear function, a problem which can be addressed using dynamic programming.

4. A dynamic programming approach

Dynamic programming algorithms can deliver an optimal solution if properly formulated, while taking much less computational time. Therefore, the problem more often lies in formulating the different stages, state and decision spaces, rather than the complexity of the problem itself. Perhaps the biggest advantage for this setting is the incorporation of non-linearly evolving parameters. In a first section, a basic formulation is given, similar to the basis IP formulation. Extensions are possible, and follow a similar structure as the IP formulation. Both the constant and varying expected waiting time extensions will be discussed.

Dynamic programming breaks up an optimization problem of a sequential decision process into smaller and better manageable sub-problems, which are linked through the formulation and recursion, and have to obey Bellman's Principle of Optimality (1957). The batch grouping problem has a total number of $\binom{N-1}{S_k-1}$ feasible solutions.

Recursion links the different decisions to be made, and is realized by implementing forward or backward procedures. Due to the nature of the problem, either of these can have advantages over the other. When considering the grouping problem at hand, forward and backward recursion result in equivalent formulations and identical computational time needed. For this work, forward recursion is chosen.

Definition of the stages and states is needed, before presenting a formulation. Stages represent the batches that are formed, and will equal the total number of batches needed, hence equaling S_k. When stages are represented this way, the different states a stage can assume represents the choice what orders have to be grouped into a manufacturing batch. Chronology must at all times be maintained, this is, the batch that groups orders n and $n+2$ must also include order $n+1$. When entering a new stage, this requires knowledge of previous decisions, this is, what orders have been grouped so far. A stage will be represented by a subset S_s, where S_s is the set of the orders that have been grouped after having reached a decision for stage s, so that $S_s \subset N$. Because of chronology, the sequence of orders must be respected, hence a subset S_s can be identified by one single parameter, being the last order u. As more batches are formed, and s increases, S_s steadily grows to eventually coincides N when the last stage has been decided upon, so that $S_{S_k} = N$.

The basis for calculations is identical to the IP formulation, but will be formulated as $c_{i,j}$ rather then c_{ij}. It remains the difference in due dates, multiplied by the order quantity of order j. As stated, the formulation uses forward recursion, and a distinction has to be made between the first stage, and stages *2* to S_k.

$$
\begin{aligned}
&For\ s=1:\\
&f_s(t)=\left\{c_{1,t}\right\}\\
&\textit{Where } t = 1, \ldots, N - S_k + 1\\
&with\ subset\ S_1=\left\{1,t\right\}\\
&&(10)\\
&For\ \ 2\le s\le S_k:\\
&f_s(t)=\min\left\{c_{u+1,t}+f_{s-1}(t_{s-1})\right\}\\
&\textit{Where } u+1 \le t \le N - S_k + s\textit{, u being the last order in subset } S_{s-1}\\
&with\ subset\ S_s=\left\{1,t\right\}
\end{aligned}
$$

This approach of course renders identical results as the IP formulation. The computational complexity equals $O(N^2S_k)$. The curse of dimensionality remains. The number of possible states for every stage grows rapidly, reducing the usability of this formulation. This problem can be tackled using different methods which will be discussed in section 5.

This basic formulation can be extended to incorporate any parameter. Restrictions on the function of included variables do not impose for dynamic programming formulations. Introducing step-wise variable inventory holding costs can be realized. Fixed setup cost remain the same for all feasible solutions and can be left out.

Identical extensions can be realized to incorporate average lead times. The formulation becomes

$$\begin{gathered}
\textit{For } s = 1: \\
f_s(t) = \left\{ c_{1,t} + (\sum_{n=i}^{j} OQ_{kn}) * E(W_k)_{ij} \right\} \\
\textit{Where } t = 1, \ldots, N - S_k + 1 \\
\textit{with subset } S_1 = \{1, t\} \\
\textit{For } 2 \le s \le S_k: \\
f_s(t) = \min \left\{ \left[c_{u+1,t} + (\sum_{n=i}^{j} OQ_{kn}) * E(W_k)_{ij}) \right] + f_{s-1}(t_{s-1}) \right\} \\
\textit{Where } u+1 \le t \le N - S_k + s, \textit{ u being the last order in subset } S_{s-1} \\
\textit{with subset } S_s = \{1, t\}
\end{gathered} \tag{11}$$

Using the IP formulation, the expected waiting time was reduced to an aggregate constant term, using the waiting time at optimal batch size Q^*, to ensure linear behavior. This requirement does not hold when using dynamic programming. A more profound extension allows the expected waiting time to evolve with the manufacturing batch size, corresponding to the original convex function with a minimum at Q^*. As the average must be maintained, and the convex function discourages use of extreme batch sizes, the effect will be stronger then the linear average lead time function. We refer to section 6, where results will be discussed. The formulation of (1) becomes:

$$E(W_k)_{ij} = \sum_{o=1}^{O_k} E(Wq_m(\sum_{n=i}^{j} OQ_{kn}))\delta_{kom} + \overline{T_{ko}} + (\sum_{n=i}^{j} OQ_{kn})\overline{X_{ko}} \tag{12}$$

5. Algorithm improvements

Using dynamic programming can lead to reduced computational efforts. Much however depends on incorporating the problem specific characteristics using the solution procedure. In this section a short description is given of the algorithm implementation. Furthermore, some add-ons will be discussed which will not only enhance the algorithm, but also reduce the computational time needed to solve the problem at hand, by domination/eliminating excessive feasible sets. Finally, the impact of these add-ons on the efficiency of the algorithm will be presented and discussed.

Although this problem can be coded in different ways, a short description may prove useful in helping to understand the discussed add-ons. Following the problem formulation, two main features can be identified. A first feature starts the algorithm, and produces all possible sets for the first manufacturing batch, being the first stage. Here, grouping of orders *1* through $N - S_k + 1$ can take place. Within this feature the actual recursion part is called. This recursion feature is basically a selection routine. Depending on the values of the incoming parameters, a distinction can be made between a normal stage ($s \in \{2; S_k - 1\}$) recursion, which performs all necessary calculations and recalls the recursion using increased parameters, the last stage ($s = S_k$) recursion, which performs all necessary calculations and compares the calculated objective function value to the current minimum, and an abortion routine, which ends the recursion when an infeasible set of parameters has been detected. The used parameters are the two orders *i* and *j* that are being grouped, the stage number and the value of the objective function at this point in the recursion. A normal stage recursion typically consists of a loop, which recalls the recursion for all possible sets that group orders $u + 1$ through $N - S_k + s$.

Infeasible sets are produced by a combination of infeasible parameters. Two extremes can be identified here. Either the last stage has been reached, but does not group all orders, or all orders have been grouped prior to the last stage decision. The second extreme is countered by restrictions in the loop of a normal stage recursion, which allows all possible sets up to $N - S_k + s$. The first extreme parameter set is dealt with by the abortion routine. However, by including a selection within the normal stage recursion routine, it can be avoided to reach the last stage prior to all orders being grouped, basically rendering the abortion routine obsolete. This selection makes a distinction between stages *2* through $S_k - 2$, which functions as described above, and stage $S_k - 1$, which will forward the manufacturing batch $(u+1, N)$, *u* being the last order batched in stage $S_k - 1$, as the last stage decision. This will exclude all recursions that are caused by infeasible sets.

Many feasible solutions will produce an object value far greater than the optimal solution. A simple example would be batches of one order, and a final batch that groups orders $N - S_k$ through N. Although feasible, this set will produce a large variance, given its extreme manufacturing batch sizes. Typically, any dynamic programming routine only verifies the calculated object value after all stages have been decided. A selection routine will determine if the object value of the last feasible set is lower than this of the current lowest object value, given a minimization problem. Incorporating this current lowest object value in the recursion will eliminate a great deal of these extreme feasible sets. As the recursion routine generates a loop consisting of the possible states within a stage, it can be stated to allow further recursion only if the object value of the subset at this point is lower than the object value of the current optimal set. When this condition is false, it is stated that all feasible sets that incorporate the current subset are dominated by the current optimal set. No feasible set that consists of this subset will produce an object value lower than this of the current optimal set, and therefore is excessive.

Another aspect of intelligent design lies in the memoization. Calculations made on all batches *i* through *j* are written in a map. Prior to any calculation, this map is investigated, and results in the elimination of recalculations. All subsequent calls do not need to be recalculated, but are taken from this memory. Introducing this aspect in the routines will greatly reduce computational efforts, as some batches demonstrate a high call rate.

The combination of these above steps result in a faster solution procedure, that arrives at the optimal value without having to calculate all feasible sets. Table 3 summarizes the results. The three major add-ons that were discussed will be referred to as (a) making the abortion routine excessive, (b) including the object value of the optimal solution in the normal stage recursion routine and (c) the memoization. For reasons of comparison, a basic model was formulated, which allows all possible values for the parameters. Possible values here refers to all values between *1* and (*N – last order batched + s*) within the normal stage recursion routine, as prescribed by the formulation. This will be the benchmark. Teller variables were added to the different routines in the recursion, and will be used to discuss the impact of the add-ons. The extended formulation including the varying aggregate expected waiting time was used.

		Basic Model	*(a)*	*(a) + (b)*	*(a) + (b)+ (c)*
Number of called recursions	**P**	19	15	13	13
	S	4367	2365	995	995
Number of recursions aborted	**P**	4	0	0	0
	S	2002	0	0	0
Number of last stage calculations	**P**	6	6	5	3
	S	1001	1001	387	11
Total times memory accessed	**P**	--	--	--	2
	S	--	--	--	886

Table 3: Impact of add-ons

It is clear that the impact of minor changes of the algorithm on a coding level can enhance the operation of the solution procedure greatly. Both (a) and (b) allow for a reduction in computational effort by reducing excessive feasible sets. The impact of (c) lies in the reduction of the amount of calculations. The impact of these enhancements raises quickly with values of N and S_k, being low for small sized problems, and high for moderate sized problems. For product P, small sized problem (N = 5, S_k = 3), a reduction in computational effort was found of 30%. For product S, a moderate sized problem (N = 15, S_k = 5), a reduction of close to 80% was obtained by implementing these add-ons. Note that the final row in table 5 shows the number of times the memory is accessed by both the last stage and normal stage recursion steps. The number of calculations made in the last stage recursion, when introducing all three changes is an absolute minimum. For any problem where N orders have to be grouped in S_k manufacturing batches, there exist $N – S_k + 1$ feasible solutions for the last manufacturing batch.

6. Results

After having presented the used data, an overview will be given of the result for the different approaches, both IP and DP. Finally, results of the coding add-ons will be presented.

The data used in this paper originates from Lambrecht et al. (1998). Here, the mathematical queueing model is implemented on an example, a small metal shop, for numerical illustration of the different steps within their proposed procedure. As stated in the above, input parameters consist of demand, service times and shop parameters. For this paper, only the demand in terms of customer orders is relevant. The example is based on two products, P and S, and the order information can be found in table 4. All other parameters are implied by the optimal lot size Q^*, which minimizes the aggregate lead time, as calculated by the queueing model in Lambrecht et al.

(1998). Indices for the metal shop use P and S as product indices. The machine index ranges from 1 to 3, O_P ranges from 1 to 3, O_S ranges from 1 to 2.

Product	Q^*	Booked orders															
P	4	Order number	1	2	3	4	5										
		Quantities	1	5	3	2	4										
		Due dates (days)	22	28	37	41	44										
S	6	Order number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		Quantities	1	3	2	3	1	1	3	2	3	1	2	3	3	1	1
		Due dates (days)	17	18	19	22	24	26	27	30	33	34	35	36	39	42	44

Table 4: Demand characteristics of products P and S

Order numbers will be used to indicate what orders have been grouped. E.g. manufacturing batch (1,4) equals the batching of orders 1 through 4. All calculations of $c_{i,j}$, used in both the IP and DP approaches, can be made based on information found in table 4.

For both extensions using the expected lead time, more information is needed. Routing information is needed, along with setup and processing times, see table 5.

Product	Operations	Machine	Setup Average	Processing Average
P	3	Cutter	20	30
		Grinder	20	10
		Lathe	24	12
		Sum	64	
S	2	Lathe	16	8
		Grinder	20	10
		Sum	36	

Table 5: Production characteristics of products P and S (in hours)

The information found in the table above is used to calculate average setup and processing times for every manufacturing batch size, as stated in (8). On average, a batch of product P needs 64 hours for setup, and a batch of product S needs 36 hours for setup, as setup times remain constant through all manufacturing batch sizes. Processing times increases linearly with batch size.

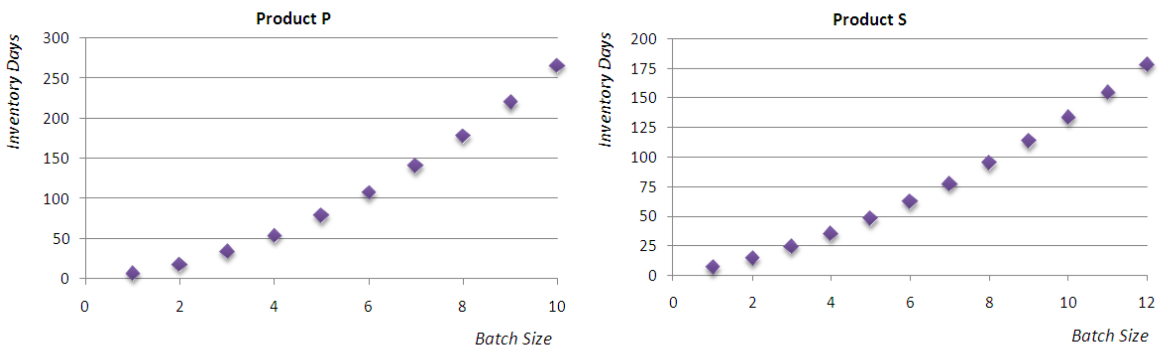
A first extension incorporates a fixed waiting time, by adding a constant term (equal to the expected waiting times at optimal batch size level) for all manufacturing batch sizes. A second extension uses waiting times that vary with the manufacturing batch size.

For this purpose, waiting times were derived from a queuing model that uses approximations as described by Whitt (1993) in order to obtain aggregate expected waiting times for every machine, given batch arrivals. To obtain waiting times per product, these have to be redistributed towards all products that face the machine on their routing. Weights used for this redistribution to the product level are those proposed in Lambrecht et al. (1998), being $\frac{l_{mk}}{l_m}$, the probability that a randomly picked product in front of machine m is of product type k. This weight is derived from the batch sizes, and will be constant, as here the optimal batch size is assumed throughout all possible manufacturing batch sizes. Results for this can be found in table 6. The aggregate batch arrival rate on machine m for product k, l_{mk}, equals $\frac{1}{144*Q_P}$ for product P and $\frac{1}{48*Q_S}$ for product S.

Machine	Expected Waiting Time	l_{mp}	l_{ms}	l_m	Waiting Time Product P	Waiting Time Product S
Cutter	4,24	0,0017361	0,0034722	0,0052083	4,24	---
Grinder	110,93	0,0017361	0,0034722	0,0052083	36,98	73,95
Lathe	46,15	0,0017361	0,0034722	0,0052083	15,38	30,77

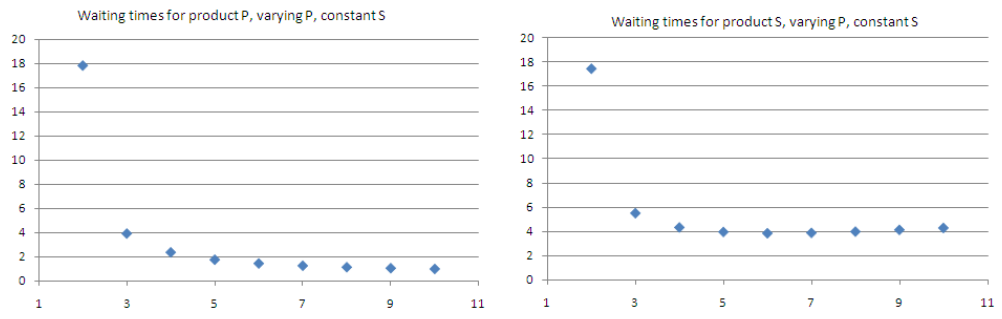
Table 6: Fixed Waiting Time Redistribution (in hours)

Waiting times for product P add up to a total of 56,6 hours, or 2,36 days, and for product S to a total of 104,72 hours, or 4,36 days. This fraction, along with setup times, remains constant throughout all manufacturing batch sizes, as they represent waiting times at optimal conditions, and is independent of the batch sizes of both product P and S. The resulting function of $(\sum_{n=i}^{j} OQ_{kn}) * E(W_k)_{ij}$ can be viewed in graph 1, which is the product of two linear functions. Note that in graph 1 all batch sizes are possible. Although expected lead time is viewed on a product level rather than an aggregate level, this does not always hold. Basic restrictions such as machine occupation still apply, and may result in a lower bound on the product level. In general, the lower bound will shift towards the vertical axis, when comparing expected lead times between aggregate and product level, resulting in a higher number of possible manufacturing batch sizes. However, this does not consolidate the statement that all manufacturing batch sizes are possible.

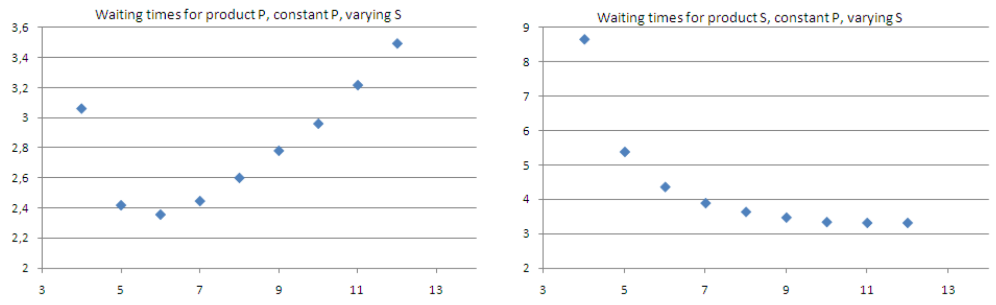


Graph 1: [Expected Lead Time x Batch Size] for products P and S

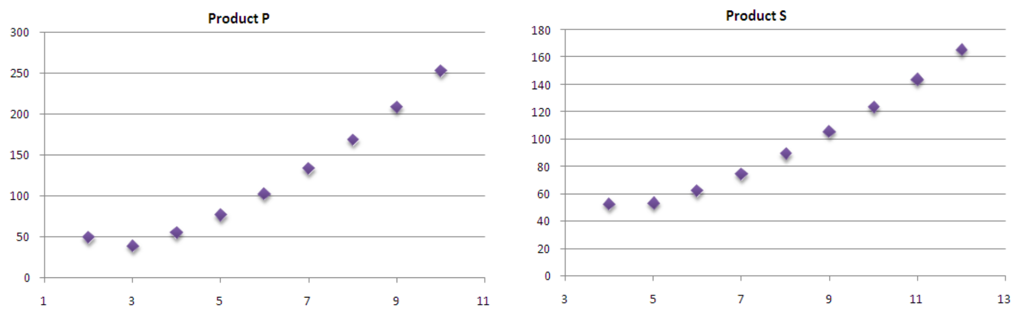
A similar approach is used when facing waiting times that vary with the manufacturing batch size. However, when using this approach, expected waiting times have to be viewed on an aggregate level, resulting in the typical convex relationship, and a minimum at Q^{crit}. On an aggregate level waiting times are the result of the interference of arriving batches of all products k facing machine m on their routing. In order to deduct the influence on the aggregate waiting time of one product, we let the manufacturing batch size (and the corresponding batch arrival rate) of this product vary, while keeping the manufacturing batch size of all other products fixed at their respective Q^*. Weights used to redistribute the aggregate waiting time are identical to the fixed waiting time approach. The results can be viewed in graph 2 for varying values of the batch size for product P, and graph 3 for varying values of the batch size for product S. Note that in these graphs Q^{crit} is the lower bound on the horizontal axis, resulting in an asymptotic behavior on the vertical axis. Graph 4 demonstrates the resulting values of the product of [Expected Lead Time x Batch Size] for both products.



Graph 2: Varying waiting times for products P and S in function of batch size of product P



Graph 3: Varying waiting times for products P and S in function of batch size of product S



Graph 4: [Expected Lead Time x Batch Size] for products P and S

The convex relationship can clearly be seen in both graphs 2 and 3, for the product with the fixed batch size. As waiting times are much higher for small batches, the trend will shift towards choosing batch sizes close to the average value, which as a hard constraint incorporates the optimal batch size. The final result is a decreased manufacturing batch size variance. The impact of including varying waiting times rather than a fixed waiting time for all batches, can be seen when observing graphs 1 and 4, which represent the first and second extensions of the basic model. When observing the difference in inventory-days (Y-axis value) for all batch sizes between these graphs, it is noted that this difference is negligible around the optimal batch size for both products, but increases when observing manufacturing batch sizes lower than Q^*.

The results for all approaches can be found in table 7. Model (1) is the basic formulation, model (2) the fixed waiting time formulation and model (3) the varying waiting time formulation.

Model	*Approach used*	*Grouped Orders*	*Corresponding Batch Sizes*	*Q^**	*Resulting Variance*
(1)	IP, DP	Product P: (1) (2) (3,5)	1 , 5 , 9	4	35
		Product S: (1,3) (4,6) (7,8) (9,12) (13,15)	6 , 5 , 5 , 9 , 5	6	12
(2)	IP, DP	Product P: (1,2) (3,4) (5)	6 , 5 , 4	4	5
		Product S:(1,3) (4,6) (7,8) (9,12) (13,15)	6 , 5 , 5 , 9 , 5	6	12
(3)	DP	Product P: (1,2) (3,4) (5)	6 , 5 , 4	4	5
		Product S: (1,3) (4,6) (7,8) (9,11) (12,15)	6 , 5 , 5 , 6 , 8	6	6

Table 7: Results of the different approaches

As can be observed, the basic model can propose very large or very small manufacturing batch sizes, resulting in a large variance. Improving this model by incorporating the expected lead time greatly improves the quality of the provided solution. Including varying waiting times further reduces variance, and manufacturing batches are obtained that proximate the optimal batch size, resulting in the lowest possible expected lead time.

7. Future research

In this, a model and extensions were proposed that group realistic orders into manufacturing batches, given certain requirements based on operational standards. The main contribution lies in the incorporation of aggregate operational performance measures to ensure optimal grouping on a product level. However, using the DP approach, extensions can also be made to incorporate different costs, using the dynamic programming the financial aspects. Any cost function can be added to produce an optimal solution that renders the lowest cost level. An ideal solution will combine both the operational and financial aspect.

Customer order information was at the basis of all calculations. In any realistic planning environment, timelines are often divided into periods. Here, only one period was assumed. A next step would involve a multi-period grouping problem, where orders at the start and the end of any period can be shifted to a previous or sequential period, if total variance of the manufacturing batch size over all periods is reduced, on a product level.

References

Bellman, R. E. 1957. *Dynamic Programming*. Princeton University Press.

Bellman, R. E., S. E. Dreyfus. 1971. Applied *dynamic programming*. Princeton University Press.

Buschkühl L., F. Sahling, S. Helber, H. Tempelmeier. Dynamic capacitated lot-sizing problems: a classification and review of solution approaches. *OR Spectrum*, 10.1007/s00291-008-0150-7.

Karimi B., S.M.T. Fatemi Ghomi, J.M. Wilson. 2003. The capacitated lot sizing problem: a review of models and algorithms. *Omega*, 31:365-378.

Karmarkar U., Lot sizes, lead times and in-process inventories. *Management Science. 33, 409-423.*

Kraemer W., M. Lagenbach-Belz. 1976. Approximate formulae for the delay in the queueing system. GI/GI/1 *Congressbook*, Eight International Teletraffic Congress, Melbourne 235-1/8.

Lambrecht M., N. Vandaele. 1996. A general approximation for the single product lot sizing model with queueing delays. *European Journal of Operational Research, 95 (Nov.)(1), 73-88.*

Lambrecht M., P. Ivens, N. Vandaele. 1998. ACLIPS: A capacity and lead time integrated procedure for scheduling. *Management Science, 44(11 – PartI), 1548-1561.*

Vandaele N. J. 1996. *The impact of lot sizing on queueing delays: multi product, multi machine models.* Ph.D. thesis, Department of Applied Economics, Katholieke Universiteit Leuven, Belgium.

Vandaele N., M. Lambrecht, N. De Schuyter, R. Cremmery. 2000. Improved lead time performance at Spicer Off-Highway. *Interfaces* 30(1), 83-95.

Whitt W. (1993). Approximations for the GI/G/m queue. *Production and Operations Management, 2, 114-161.*