UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Jeanne-Marie Hugo
15038832

September 2018
BPJ 420 Final Project Report
Department of Industrial Engineering
University of Pretoria

Batch sizing and scheduling on a volatile production line through the usage of linear optimisation.

1. Declaration of Originality

Full names: Jeanne-Marie Hugo

Student number: u15038832

Declaration

1. I understand what plagiarism is and I am aware of the University's policy in this regard.
2. I declare that this is my own original work
3. Where other people's work has been used (either from a printed source, internet or any other source) this has been carefully acknowledged and referenced in accordance with departmental requirements
4. I have not used another student's past work to hand in as my own
5. I have not allowed and will not allow, anyone to copy my work with the intention of handing it in as his/her own work

Signature: Jeanne-Marie Hugo

(By typing my name, I pledge that I signed this declaration.)

2. Executive Summary

This report examines and determines the optimisation solution to a complex batch sizing problem. A mix of low- and high-volume production causes complexity in batch sizing and scheduling at Mecalc Manufacturing. Prototyping and a volatile demand forecast only adds to the intricacy. After in-depth analyses of the production problem, it is determined that the main restriction of the optimal solution is parameter control.

Various Industrial Engineering techniques that relate to batch sizing and operations research are studied in the report. With the control and determination of the production variables being the most crucial factor, the literature study is summarised into three main models. The first model that is considered, is an economic manufacturing quantity (EMQ) with random breakdowns (Lin & Kroll 2007). Stochastic programming is then studied, from which the next two models originate – The recourse model with a multi-stage problem and the probabilistically constrained model. These two are very similar in results, working with production and variable scenarios.

After comprehensive comparisons between the models researched, it is decided to formulate both stochastic-type models. The reason for this is the unpredictability of Mecalc's production parameters and data. Both models are programmed on Python giving results that are not satisfactory for Mecalc's unique production process. The multi-stage model is chosen as the base for the solution model and adjustments are made accordingly. The final solution formulation is clearly laid out with all its elements, explanations, and interpretations.

Variables are analysed, and complexities are determined. The data frames that function as the input, are explained with some of the variables' simulations. Validation of the solution and the sensitivity analysis thereof is evaluated nearing the end of the report. Some analyses of which resulted in determining the parameters in which the model works and doesn't work. Variables such as inventory quantities, maximum batch sizes, and reliability factors are assessed. Finally the report is concluded with the proposed implementation with some recommendations for Mecalc.

3. Table of Contents

4. List of Figures

5. List of Tables

6. Abbreviations

SMT	Surface Mounting Technology
PCB	Printed Circuit Board
CNC	Computer Numerical Control
FIFO	First In First Out
JIT	Just In Time
EMQ	Economic Manufacturing Quantity
EOQ	Economic Order Quantity
ELS	Economic Lot-Sizing
SP	Stochastic Programming
SBSP	Stochastic Batch-Sizing Problem
SQL	Structured Query Language

7. Background

Batch sizing and scheduling combined with setup times and costs have been investigated by multiple researchers. Extensive literature is found when it comes to decisions integrating batching and scheduling. Batching can be done on jobs with identical setups on a machine or when a single machine can simultaneously process numerous jobs (Potts & Kovalyov 2000). This paper is however centred around the latter, focussing on dynamic programming algorithms for solving problems referred to later in this study.

8. Mecalc (Pty) LTD and Mecalc Manufacturing (Pty) LTD

The company designs, develops and manufactures advanced acquisition and control systems. These systems are utilized to optimize noise, vibration and structural integrity in prototype or quality control testing. The mechanical parts of the systems are machined by Mecalc Manufacturing – where most of the project is based at.

Figure 1: System (source: MECALC)

In the above picture (figure 1) is a PAK MKII system. This system is composed of mechanical and electrical parts, where the mechanical parts are manufactured by Mecalc Manufacturing using computer numerical control (CNC) machines. These parts are primarily made of aluminium. However, the electronical parts are sub assembled with printed circuit boards (PCB) made inhouse by surface mounting technology (SMT) machines.

Being leaders in their field, quality is of utmost importance to the company. Not only do they produce systems of exceptional quality, but the company's after sale customer relationships are also important. As a standard procedure the company offers customer service to their customers for up to 15 years after the initial purchase. Examples that include repairs, calibrations and part replacements are therefore part of the production process.

Mecalc constantly designs and develops new and improved parts and systems, due to strong global competitive forces and innovations. Therefore, the engineers at Mecalc always design new prototypes that are created and tested.

9. Process overview

This paper only focusses on the mechanical production, as well as the ordering thereof. After mechanical production follows electronical assembly that is an entire process of its own; this does not form part of the study. The simplified mechanical production process of Mecalc is formatted in Fig 2 below.

Figure 2: Mecalc's Production Process Overview

The **forecasted demand** is annually compiled by management. Since Mecalc's sole customer is Muller BBM, based Germany, the basic forecast is easily determined. As one can expect, Muller BBM strays from the forecast from time to time. This causes the demand to be quite volatile.

The production order is given to the production manager by management of Mecalc, this is evidenced form a **job card**.

The production manager then sets up his **production schedule** based on the demand, available labour and all the technical details of his CNC machines.

He issues **job cards** to the workers, with all the specifications of the job on the cards.

The relevant workers start their job by **setting up** the machines. The setup is the total time it takes before machining of a batch can begin. Before every unique machining operation, the CNC machine must be programmed according to the assembly drawing design. At Mecalc, there is no fixed setup time since some parts are more complex and require more detailed programming. Included in the setup procedure for machining is exchange/replacement of dies, machine cleaning, cutting testing, transport of materials, and job card creation etc. None of these activities have fixed or calculable times.

Only now the **machining** of parts can commence. When a batch is finished it is sent to its next intermediate process which might include painting, anodizing etc.

The parts are packed after final inspection and **shipped to assembly** where it will be combined with all the necessary electronic parts to form the system(product).

10. Problem statement and scope of objectives

Mechanical scheduling at Mecalc Manufacturing is done periodically and is not shockproof to order variability. If a part prototype (of batch size one or two) needs to be cut, the entire production in the queue of one CNC machine is paused to prioritise the prototype. The engineers first need to test the prototype after its completed manufacturing, and as such the process can take longer than expected. The engineers may after product testing either request further production of the part, as part of the product system; or they will review the part and make design changes resulting in another prototype being cut. This creates uncertainty within the production schedule.

The current scheduling system is according to FIFO (first in first out) of the orders that are sent to production. It is done by the production manager and every schedule that is drawn up is unique. Set priorities are therefore not followed. Normal batch sizing cannot accommodate the complexity of these uncertainties. Additionally, Mecalc's CNC machines' setup times can range anywhere between 30 mins and three hours, this adds to the variability of production turnover rates in a given timeframe.

11. Project Aim

With the company's scheduling system being FIFO without priority of various part productions, the aim is to eliminate confusion and shorten the total production time of mechanical manufacturing. To achieve this, Mecalc requires a model to calculate and determine the optimal batch sizes in advance – preferably on a weekly basis.

The grouping of jobs is desirable in a setting with technological features of process capability, such as Mecalc. One can find the optimal solution by exploiting such a feature, or at least identify the dominant job grouping (batch size). The motivation of grouping at Mecalc, relates to the existence of lengthy setup times on the CNC machines. With time orders change, and prototypes are designed. The model will have the parameter of the demand forecast that can be changed at any point by management. The model should automatically change the batch sizes to find the new optimal quantities if any changes in parameters occur. The output will be easily understandable so that the production manager is able to set up his schedule and job cards for the workers. Taking all the above into consideration, a model in the form of an optimisations program would be the best option to find a solution. The reason being that it can incorporate all the mentioned constraints and variabilities.

Lastly, Mecalc would like the option to use the algorithm in their system even if they change some production processes. They might prioritise some variables over others and eliminate some unnecessary ones later. They might also like to use the program when they decide to build another manufacturing plant elsewhere. With these possibilities in mind, the project should include extension options for the company to be able to use in the future.

12. Literature Review

A. Single machine batching

In a production problem where a single machine is considered with a deterministic demand of one customer order, all orders can be satisfied by changing batch sizing.

Below in Figure 3, is a diagram of two different production schedules for the problem of processing Q units of one order with due date d (Chrétienne et al. 2011). Ψ_1 indicates one extra batch in excess of Ψ_2, but q_1 is the only production unit that is not lagging, out of all production units in the figure. Chrétienne assumes related products with similar production times, where this paper is centred around various parts with different production (machining) times. The problem is just formulated differently in the end, but the batch model described in the figure remains the same.

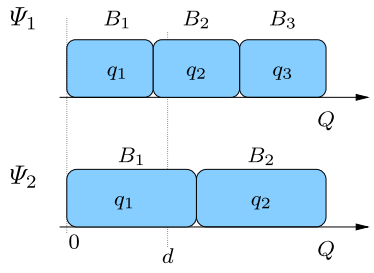

Figure 3: Tardiness vs. setup costs (source: Chrétienne et al. 2011)

As illustrated in Figure 4, Batches B_1, B_3 and B_4 are premature and uniform, where B_2 is on time but also uniform. The rest of the batches are tardy, where B_6 is mixed. A new schedule can be compiled by moving start times of batches earlier. This can be done because no early production penalties are taken into account. (Chrétienne et al. 2011)

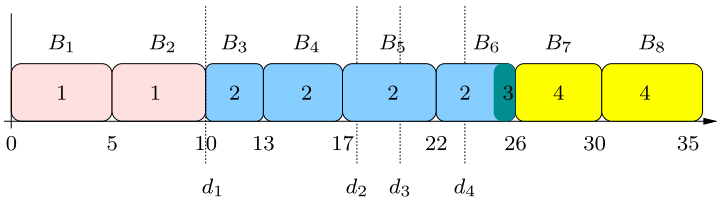

Figure 4: Illustrative example (source: Chrétienne et al. 2011)

Should the number of batches increase; the setups increase. More setups equate to more setup costs. This cost should be compared to the opportunity cost incurred when a production batch is tardy. It is therefore ideal to have an optimal solution where the batches are on time, with a minimal total production cost. Setup costs are just one constraint a production process can have. The next constraint that is examined is machine breakdown.

B. EMQ with random breakdowns

When most production-inventory models are built, the reliability factors are ignored. There is an assumption that the production is not subject to deterioration and/or breakdowns. Defects or scraps are produced when a process state deteriorates from in-control to out-of-control (Lin & Kroll 2007). The optimal batch sizes increase as there is an increase in equipment failure possibility. (Groenevelt et al. n.d.) With high production flow, a machine breakdown stops the flow and causes extreme disruption.

Models such as the economic manufacturing quantity (EMQ) model (Erlenkotter 1990) are widely used. This model typically accounts for a deteriorating process or breakdown possibilities in the process, but not both. Lin and Kroll address this issue. Their article uses the policy that a manufacturing lot is immediately terminated when a breakdown occurs. The new lot will only commence when the available inventories are depleted. They also use the time intervals between events as independent continuous random variables having known exponential distributions. (Lin & Kroll 2007)

Process deterioration speeds up as the production speed increases, maintenance is likely to improve this. Maintenance is however an added cost, where production earns profit. The company thus needs to choose between the two. A choice between products is also important as a high end product provides a higher profit than a low-end one, but leads to a longer manufacturing time – elevating the future maintenance needs and costs. (Kazaz & Sloan 2013).

Lin and Kroll assume the process to be in an in-control state due to the setup having some maintenance. They derived a near-optimal solution stemming from the basic model with a constant percentage of defects. Extended models are also derived with the consideration of the dynamic deterioration (linear and exponential) processes. All these models assume that the defective units can be repaired or reworked. The model can be expanded if the production system is said to be multi-stage. (Lin & Kroll 2007)

C. Stochastic Programming

Mathematical programs can be used to solve numerous decision problems. A function called the objective is aimed to be minimised or maximised during the process. These decisions are constrained by parameters in resources, system requirements, etc. These parameters are represented by variables, where the objectives and constraints are functions of the variables and the data. Examples of data include machining times, unit costs, demand, sales or usage rates. A rational way to set out the problem, is to add the requirement of making immediate decisions and minimising expected costs,

time or utilities that the decision will bring about. This paradigm is called the recourse model. (Holmes 2017)

1. Recourse model with multi-stage problem

Although recourse models can be extended in more than one way, it is most common to include additional stages. With this type of problem, one should effectively make a decision now (at $t = 0$) and wait for some uncertainty to be determined. Only after the realisation of the uncertainty another decision can be made based on what has happened. As discussed previously, die objective function would still be a minimisation of a cost, time or utilisation function. (Holmes 2017)

A distinction needs to be made between a multi- and a two-stage model. The comparison can be seen in Fig 5. The stochastic production problem has four different scenarios (S) with three time periods (t). * indicates the time points for determining the normal ("base") model, where Δ indicates when a decision needs to be taken on an update or recourse.

In Fig 5 the diagram on the left is the two-stage model where the starting point decisions of production must be determined commencing $t = 1$ with no form of uncertainty in the information. The production updates can only be made after awareness of an uncertainty in period one. With the multi-stage on the right, the first point of departure decisions is identical to the previous model. The difference comes in with it having a larger decision space in its baseline production decisions and production recourses. These recourses are allowed based on its preceding decisions and recognitions. (Hu & Hu 2018)

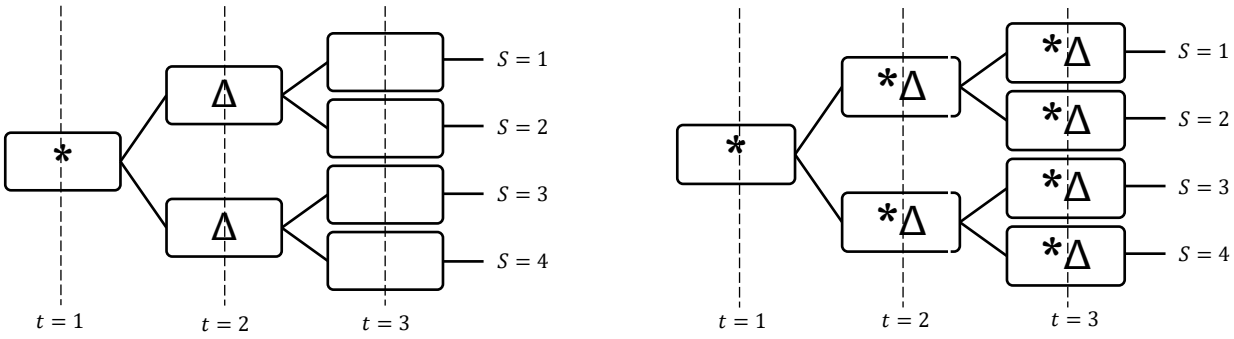

Figure 5: Two-stage model (left) vs Multi-stage model (right)

Hu and Hu's stochastic model for the multi-stage problem is referred to in Appendix C. The uncertain factor that is examined is the demand. Continuous functions are computationally difficult in the model, therefor the uncertainty is represented with discrete scenarios that are probabilistic. The scenarios are represented as the set $S = \{\mu_1, \dots, \mu_S\}$ corresponding to the probability (ν_S) of the original distribution.

Example scenarios at Mecalc would include the review (design change) of a part, a machine breakdown and a prototype being added into production. Each of these would form part of the scenario set S. A recourse of the production schedule and batch sizing will then be considered accordingly.

2. Probabilistically constrained models

In contrast to the above recourse method – it might be more fitting to use a probabilistically constrained model. A set of constraints will rather be used holding a certain probability. Sen and Lulli (2004) formulates a model for the multi-stage stochastic batch-sizing problem (SBSP).

One of the first developed unconstrained models has fixed setup costs with the inventory and production costs being of a linear nature (Wagner & Whitin 1958). Sen and Lulli consider the stochastic version of the problem, but add uncertainty cost parameters of demand, production, setup and inventory. These parameters are discrete random variables. The related decision variables are also discrete. Wagner and Whitin's conditions for the optimal solution do therefore not hold for the SBSP. Optimal solution for stochastic models provides the distinction that they can have non-zero levels for production even if there are no inventory levels of that material (Sen & Lulli 2004). Sen and Lulli have similar scenario generations to the recourse model, but on a much bigger scale. Their model is found in Appendix D.

D. Model comparisons

When defects or breakdowns are considered, Lin and Kroll (EMQ with random breakdowns) use breakdown rates where the probability of a breakdown increases with time. The recourse model (Recourse model with multi-stage problem) uses the option of a breakdown as a scenario. Sen and Lulli's model (Probabilistically constrained models) sets the probability of the breakdown scenario.

All have an optimal solution, but the EMQ model does not have enough capability for Mecalc's numerous constraints. Only the remaining two options are formulated and programmed for Mecalc's batching problem. These remaining options are not a perfect fit for the company and adjustments were made as the models were built. This can be seen in the next few sections of the report.

E. Programming language

The programming of the model is the way in which the resulting batch sizes will be calculated. Python was determined as the programming language of choice. Python has an extension software called PuLP, used for optimisation. In PuLP, the mathematical program is formed before the solving of the program. Key decision variables must be clearly defined, as well as the objective function and constraints. The software is free and open source, written in Python. (Stuart Mitchell et al. 2009)

The software allows for importing input data and exporting output data in preferred layouts. Making the batch sizing results easy to manipulate in the way Mecalc would like.

13. Data analysis and compilation

Before the results could be found from the models, the right data needed to be compiled. Since the only definite data available is the demand (of a sample in time), most variables had to be randomised for the testing. These randomisations were done by using historical data from Mecalc and formulating parameter distributions accordingly. Samples of the data frames can be seen in Appendix A. The model is formulated in a way where the parameters can be changed when needed. Especially those parameters not yet used by the company. A sample of Mecalc's top 200 most produced parts were extracted and a timeframe in which they do demand planning is best with a maximum of 8 weeks ahead. Below is the list of parameters with their respective specifications.

The per-unit **holding cost** was chosen to be a preliminary uniform rate of R1 as Mecalc does not have restricted inventory space at this point in time.

The per-unit **machining time** is found to be a normal distribution with a mean of 90 minutes and a standard deviation of 20 minutes.

The per-unit **production cost** is measured as R100 plus a variable cost of R1.2 per machining minute.

The per-unit **setup time** is found to be of a Poisson distribution with a rate of 90. An exception to this rate is a prototype as it takes even longer to set up a machine for an unfamiliar part. The setup time for a prototype is taken to be 120 minutes.

The per-unit **setup cost** is measured as R50 plus a variable cost of R0.50 per setup minute.

Concerning the part **review probability**, the following probabilities are chosen**:** 1 if part is a prototype, 0.75 if the setup time is high, and 0.5 otherwise.

The **priority** of a part is chosen to be of a uniform distribution as Mecalc does not use the metric yet. The options are 1,2, and 3. 1 being the highest and 3 the lowest.

A **prototype** is shown as a binary variable. 1 being true and 0 being false.

The **fragility** of a part has also not been measured by Mecalc yet. A uniform distribution with fragility percentages with options 0.3, 0.2, and 0.1 are chosen.

14. Model comparisons

After setting up all the data frames, the recourse model with a multi-stage problem ("Model 1") and the probabilistically constrained model ("Model 2") found in Appendix B and C respectively, have been coded. Screenshots of the Python codes can be seen in Appendix D. Both were firstly formulated with the exact same objective function and constraints as written in the mentioned appendixes. Only after a working model came to surface, adjustments were made to accommodate the actual data.

Table 2 below is a sample of Model 1's batching results. As seen by the repetition of production quantities, the model equally distributes production according to total machine and production capacity, not prioritising the demand. The demand quantity will essentially only be reached weeks after its due date. Even though cost is minimised in the objective function, capacity is prioritised by default. Because this recourse model has the option of overproduction and backorders, production keeps shifting on to the next week.

Table 1: Results of Model 1 for part 19

Week	Demand	Starting Inventory	Regular Production	Overtime Production
1	0	0	7	11
2	155	18	7	11
3	0	0	7	11
4	40	0	7	11
5	0	0	7	11
6	0	0	7	11
7	0	0	7	11
8	0	0	7	11

Taking the entirety of Model 1's complications into consideration, all the different outputs do not meet demand on time. This tardiness is not acceptable and thus means the model needs to be adapted even more to be considered as the best solution to Mecalc's problem. These alterations need to be added over and above the parts having extra probabilistic constraints.

In comparison to the above, Model 2's batching results are seen in Table 3 below. It was run over seven time-intervals for only one part. Because this model uses only a single part, one with a near weekly demand was chosen from Mecalc's demand data. The company has more than enough storage

capacity and does not find inventory to be an issue. They do not use holding and storage costs in their current calculations. Here, a small holding cost of R1.00 was used to have a working model.

Table 2: Results of Model 2 for seven time-intervals

Day	Demand	Inventory level	Production batch level
1	0	80	0
2	80	0	0
3	50	0	50
4	50	0	50
5	20	25	45
6	0	25	0
7	100	0	75

Although the program gives the correct optimal solution, further analysis was done to determine if variables could be added to create probabilistically constrained parameters. For example, adding the reliability of a part in the form of a percentage made the solution infeasible as its main goal is minimising costs by keeping batch sizes to a maximum (and number of setups to a minimum). Adding more parts did however not prove to be a problem, it just made the algorithm more complex. Nevertheless, it was ultimately decided that Model 2 would not be the best option for the project, even if it is altered to fit Mecalc's needs. The constraints that still needed to be added are just too much for the programmed model to handle.

It is clear that the biggest barrier was the capability of the two model when constraints were added. In both, random adding of new parts created infeasible solutions and the addition of a constraint where no more than two setups can happen simultaneously added unnecessary tardiness in satisfying demand. An alternative solution was formulated to satisfy as many constraints as possible, adding the complexity of probabilistic parameters.

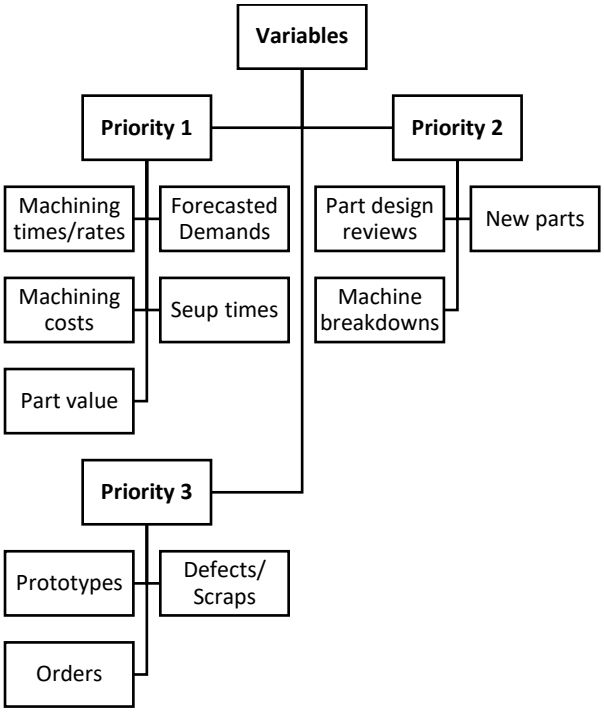

Figure 6: Variables and their Priorities

As in the above Figure 6, the variables are grouped according to priority, 1 being the highest and most important for the optimisation model. This is done for the process of elimination of variables when it is realised that not all variables can be random or indeterminable from the start – it is possible to change the variable type. With this illustration, it was determined that orders out of demand, and defects are not important enough to add to the solution model. They are in the priority 3 category and can therefore be put aside as they will not make a big enough difference in the optimal batch sizes. They add too much to the complexity. Setup costs are also added, proving to make a substantial impact on results as mentioned before. A new grouping is created and is illustrated in Figure 7 below.

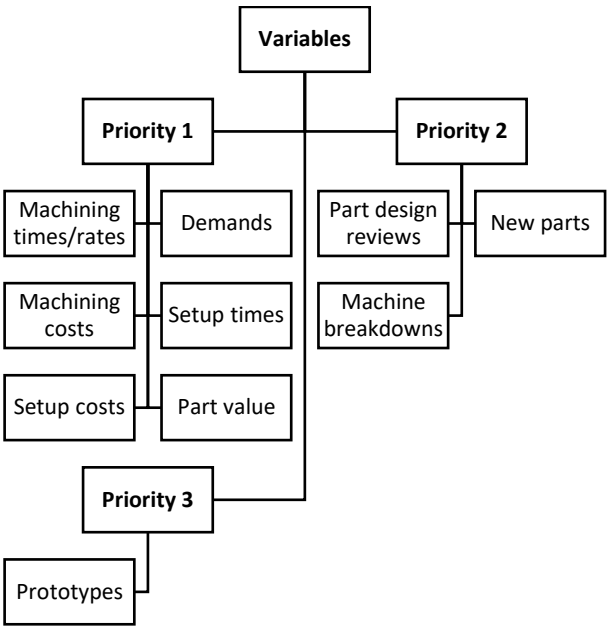

Figure 7: Variables and their Priorities (Updated Version)

15. Solution model

Taking all the complexities encountered in the programming of Model 1 and 2 into account, an alternative is formulated as the final lot-sizing and scheduling solution on parallel machines with sequence dependent setups.

We let $I = \{1,2, \dots, N\}$ be the set of parts with $i, j \in I$, $M = \{1,2, \dots, M\}$ be the set of machines with $m \in M$, and $T = \{1,2, \dots, T\}$ be the set of time periods with $t \in T$. The solution is based off Model 1. With the time parameters, a uniform set of time periods (T) was created. Mecalc's demand is on an end of week basis, where the other parameters such as machining time and setup time are in minutes. The overall scheduling time was decided to be on a daily basis. This made setups easy to carry over to the next day. The previous weekly demand became the demand for the last day of each week, causing the model to add the parts to storage until then. Below is a list of assumptions that are taken into consideration with the building of the model.

1. Model assumptions

- The is an unlimited amount of aluminium as material.
- The initial inventory is assumed to be zero.
- Since the demand won't be fulfilled each time t, inventory is are allowed.
- Demand is not time dependent (Hu & Hu 2018), thus demand in time t is not dependent on time $t - 1$.
- At the end of each period (weekly) uncertain demand is realised.

- At the end of each period, inventory is measured.
- A setup can be carried over to the next period making it the first default setup in the next period.
- Only two setters are available at every time *t*, meaning there can only be two setups happening simultaneously.
- A production day is 10 hours.
- A production week is 5 days.

2. Parameters

$d_{i,t}$ Part *i* demand for time *t*
h_i Part *i* holding cost
cap_m Machine *m* capacity time
$mach_i$ Part *i* machining time
p_i Part *i* production cost
sc_i Setup cost for part *i*
st_i Setup time for part *i*

3. Decision variables

$I_{i,t}$ Part *i* inventory quantity at end of time *t*
$X_{i,m,t}$ Part *i* production quantity on machine *m* during time *t*
$V_{i,m,t}$ Auxiliary variable assigning sequence of production on machine *m* for time *t*.
$Y_{i,j,m,t}$ Binary variable which is 1 if there is a changeover with a setup from part *i* to *j* on machine *m* at time *t* and 0 otherwise
$Z_{i,m,t}$ Binary variable which is 1 if there is a changeover from part *i* to *j* on machine , at time *t* and 0 otherwise

4. Objective function

$$\text{minZ} = \sum_{i=1}^{M}\sum_{j=1}^{I}\sum_{}^{I}\sum_{}^{T}(sc_j * Z_{i,j,m,t}) + \sum\sum^{I}\sum^{T}(h_i * I_{i,t}) + \sum^{M}\sum^{I}\sum^{T}(p_i * X_{i,m,t}) \quad (1)$$

5. Subject to constraints

$$I_{i,0} = 0, \quad \forall i \quad (2)$$

$$I_{i,t} = I_{i,t-1} - d_{i,t} + \sum^{M}(X_{i,m,t}), \quad \forall i, t \quad (3)$$

$$\sum^{I}(mach_i * X_{i,m,t}) + \sum_{i=1}^{I}\sum_{j=1}^{I}(st_j * Z_{i,j,m,t}) \leq cap_m, \quad \forall m, t \quad (4)$$

$$X_{i,m,t} \leq B\left(\sum_{j=1}^{I} Z_{i,j,m,t} + Y_{i,m,t}\right), \quad \forall i, m, t \quad (5)$$

$$Y_{i,m,t-1} + \sum_{j=1}^{I} Z_{i,j,m,t} = Y_{i,m,t} + \sum_{j=1}^{I} Z_{j,i,m,t}, \quad \forall i, m, t \quad (6)$$

$$\sum_{i=1}^{I} Y_{i,m,t} = 1, \quad \forall m, t \quad (7)$$

$$X_{i,m,t} = 0, \quad \forall i, m, t = T \quad (8)$$

$$V_{i,m,t} + N * Z_{i,j,m,t} - N * Y_{i,m,t} \leq V_{i,m,t} + N - 1, \quad \forall i, m, t \quad (9)$$

$$\sum^{T} X_{i,m,t} \leq B * A_{i,m}, \qquad \forall i,m \tag{10}$$

$$\sum_{m=1}^{M}\sum_{i=1}^{I} Y_{i,m,t} \leq 2\,, \qquad \forall t \tag{11}$$

$$X_{i,m,t} \geq 0 \tag{12}$$

$$I_{i,t} \geq 0 \tag{13}$$

$$Z_{i,j,m} \in (0,1) \tag{14}$$

$$Y_{i,m,t} \in (0,1) \tag{15}$$

$$Z_{i,m,t} \in Z \tag{16}$$

$$V_{i,m,t} \in \mathrm{R} \tag{17}$$

6. Model description:

Equation (1) sets out the objective function, which is the sum of setup, holding and production costs – minimizing the total production cost. Constraints (2) sets the initial inventory. (3) is for the inflow and outflow of the parts. The inventory can be extra, allowing it to be fulfilled later. Constraint (4) makes sure the daily capacity per machine is not exceeded by production. Enforcing the constraint (5) ensures the setup of a machine when production takes place. Constraint (6) keeps track of the carryover of a setup while (7) ensures every machine is set up for one production run in each period. Constraint (8) states that there is no activity allowed in the last time period, as it is only the setup that is carried over to the next. While two consecutive periods are linked with (6), the subtours and detouring of nodes are eliminated by constraint (9). Constraint (10) ensures that a machine only produces a part that it can produce. With only two setters, constraint (11) is set up.

F. The problem of probabilities

Adding probabilities to the model, makes it unsolvable using an average computer. For Mecalc, this is a big issue since both the management and production staff should be able to do calculations of the algorithm. It was decided to rather use the probabilistically constrained parameters before model calculations. More specifically, to add them before the final calculations. Adding them in such a way, decreases the model complexity too.

When anything changes with prototypes being added to the demand or simple demand changes occur for example, recourse is possible. The model would just recalculate the schedule after alterations have taken place.

G. Adding reliability factor:

The model aims to have the lowest cost with it being a minimisation function. Adding the reliability factor α, the model would favour the smaller factor over the higher. The variables that were used needed to be manipulated to get an optimal factor for every part used in production. Part fragility μ, probability of a review β and production priority ϕ are used in the calculation. The final formula is stated below.

$$\alpha = \phi + \frac{2\beta + (1-\mu)}{3} \qquad (equation\ 1)$$

The priority of a part is the most important when a schedule is drawn up. For this reason, it is left as-is. The next part is a weighted average between the review probability β and fragility μ, with β carrying double the weight. This weight was chosen after consultation with Mecalc. The volatility of their manufacturing process is more important than the physical properties of a part that form its respective fragility. However, if they wish to change it they can. Just like any other variable that can be changed.

For example, part one and two are compared. Part one with $\phi = 2, \beta = 0.5, \mu = 0.1$ and part two with $\phi = 3, \beta = 0.5, \mu = 0.1$. They amount to 2.63 and 3.63 respectively, prioritising part one (the lower) over part two.

H. Adjusted objective function:

$$minZ = \sum_{i=1}^{M}\sum_{j=1}^{I}\sum_{}^{I}\sum_{}^{T}(\alpha_i * sc_j * Z_{i,j,m,t}) + \sum\sum^{I}\sum^{T}(h_i * I_{i,t}) + \sum\sum^{M}\sum^{I}\sum^{T}(p_i * X_{i,m,t}) \qquad (20)$$

I. Additions to consider

Although the final model works for Mecalc and their needs, their needs might change soon. To accommodate possible changes, a few extra constraints and model properties are proposed. The first one would be overtime and backorders, allowing the unfulfillment of demand. $O_{i,m,t}$ would denote the number of overtime units produced of part i on machine m during period t and $B_{i,t}$ would denote the number of parts backordered during period t. These can be added by changing constraint (3) in the following way.

$$I_{i,t} - B_{i,t} = I_{i,t-1} - B_{i,t-1} - d_{i,t} + \sum^{M}(X_{i,m,t}) + \sum^{M}(O_{i,m,t}), \qquad \forall i,t \qquad (21)$$

Since overtime production time is not equal to regular time production, a factor between the two should be used, as showed in constraint (22). θ is the ratio between the two and its value is decided by the industry.

$$\sum^{I} O_{i,m,t} \leq \theta * \sum^{I} X_{i,m,t}, \qquad \forall m,t \qquad (22)$$

16. Results

The model was run without constraint (10), (11), (21) and (22). Constraint (11) was added as part of the sensitivity analyses in §17. The reliability factor α is also only added at model finalisation as it does not directly affect the validity of the model, only the resulting schedule and batch sizes. The sets were restricted to 30 parts, 8 weeks, and 10 machines. This was done for the model to run quickly and form definite results. Below is a snippet of part 19's results which was randomly chosen. The days range from day 2 to 15. The manner the demand was satisfied can be seen clearly. Day 7 to 13 is omitted because of repetition.

Table 3: Snippet of part 19's results

Time (days)	Inventory	Production	Demand
2	63	63	0
3	133	70	0
4	214	81	0
5	74	0	140
6	74	0	0
…			
14	108	34	0
15	0	252	360

17. Analysing model sensitivity

The ultimate test is to determine how flexible the model is and how big the difference in batch size change would be if something in the model changes. This leads to a sensitivity analysis that will be

done on the variables, with the focus being the topics showed in Figure 8. Computational experiments will be done with the interim solution algorithm.

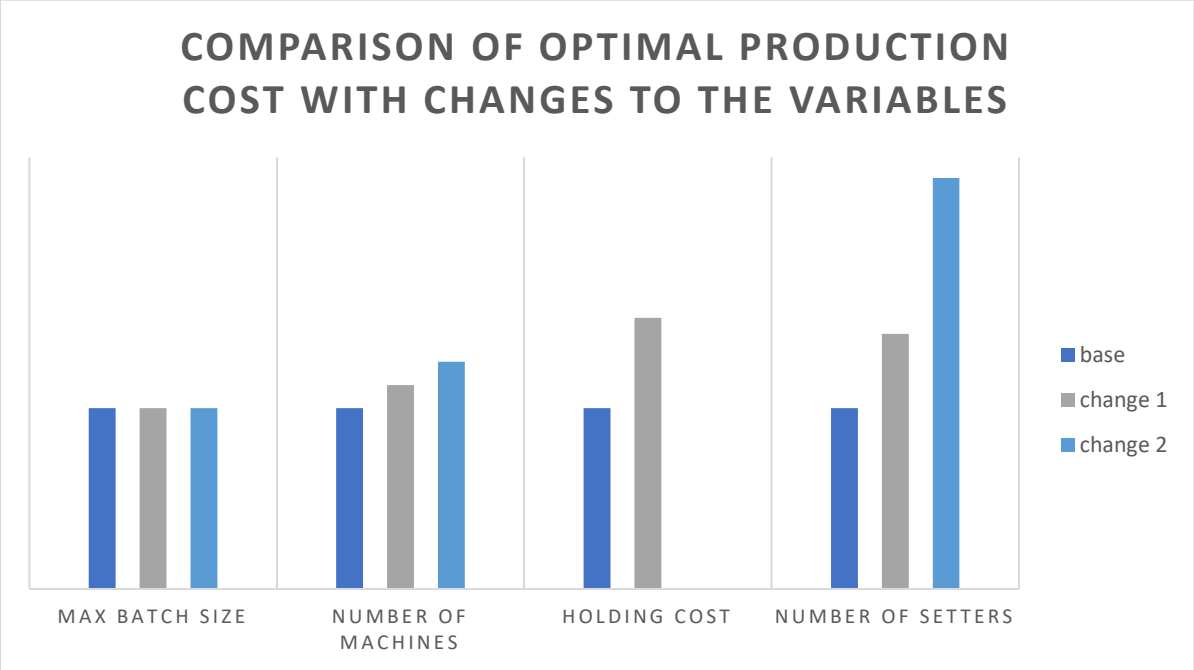

Figure 8: Graph with sensitivity analyses comparisons

A uniform **maximum batch size** B is unknown and its impact thus needed to be tested. No difference in outcome could be found in changes of this variable. A value of 350 (base) was used in the rest of the calculations.

At the base, the **number of machines** were changed from 9 down to 8 and 7 respectively. It is seen that there is a constant difference of overall cost increase when the total number of machines are decreased. More machines running cuts the total production cost enough for it to make a significant difference.

With the **holding cost** not being of importance now, the difference of R1 per part per day was compared to R1.5 from the base. The only effect this change had on the optimal solution was the total cost, more specifically on the inventory cost.

Adding an overtime option to the model made no difference to the output and is therefore left out to decrease complexity and the time it takes to solve the calculations. With this, the **number of setters** is set to unlimited in the base and adjusted twice to see the effect of limiting the number. The model is first constrained to three setters and then to two setters at any given time. It is evident that these changes have a negative effect on the total cost as the model gets more constrained.

J. Planning horizon

Expanding the production schedule to 8 weeks instead of 4 weeks, had a major impact on the optimal solution. The longer timeline of the two is roughly 40 times more expensive. This is expected since the demand is much higher in week 7 and 8. The validity of the impact of the planning horizon change is not calculable because of demand having the ability to change drastically from one week to the next.

Expanding the schedule influences the production of some parts. As seen below in Table 5 part 4 is scheduled early on the timeline and 217 units are kept in inventory for 18 days before the demand is

satisfied. This can be done because of the holding cost not having an impact. Although the batch sizes of production might seem irregular, it is still the optimal cost solution.

Table 4: Snippet of part 4 results

Time	Inventory	Production	Demand
13	63	10	0
14	112	49	0
15	112	0	0
16	212	100	0
17	217	5	0
18	217	0	0
…			
34	217	0	0
35	0	0	217

K. Number of parts in the model

When more than 45 parts are added to the model, the time to calculate the solution is too long. Solving the model more quickly unfortunately restricts the number of parts to 45 or less. One way of simplifying this is to only add parts with a definite demand on the planning horizon in the data frame. This way the program does not loop through unnecessary parts.

L. Analysing inventory

Even though holding costs of parts are not of importance for the model, one cannot discard the possibility of it becoming a restriction in the company in the future. An analysis is done on the inventory per part during the eight-week period. As seen below in figure 9 There is a big spike in inventory from day 25 to day 34. As mentioned in §17.J, this is due to the big demand near the end of the eight weeks. The model "stocks up" on inventory before the demand is satisfied, leaving an empty storeroom on day 40.

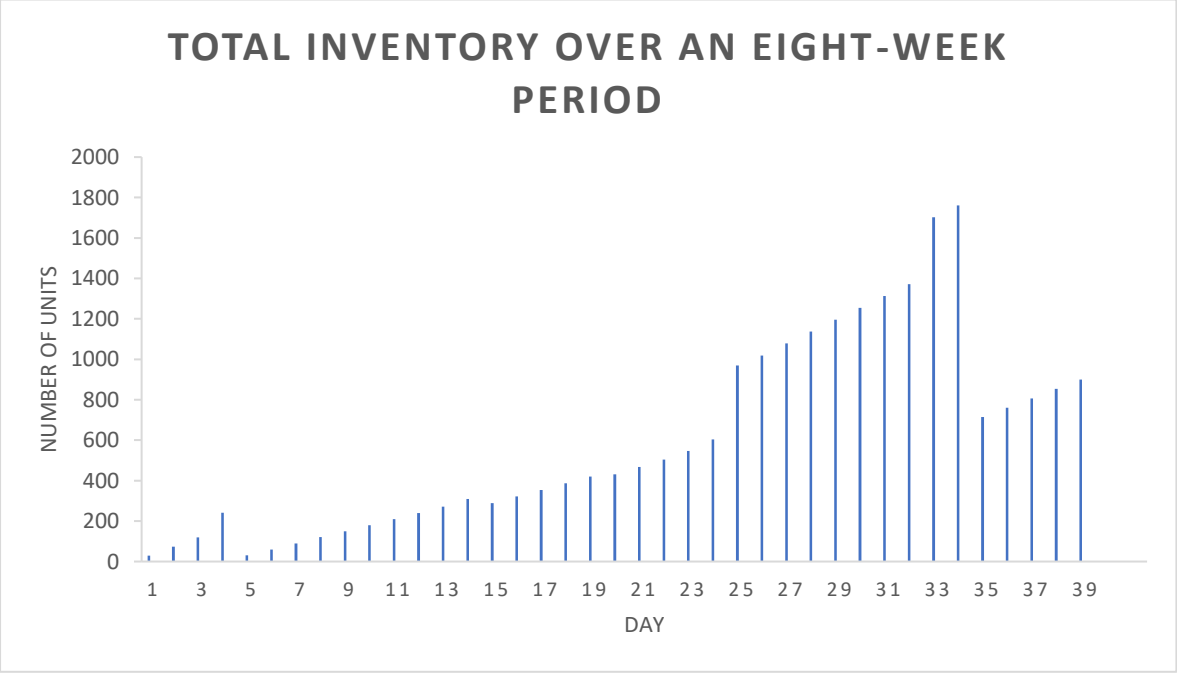

Figure 9: Graph of the inventory over an eight-week period

This gives an indication of the behaviour of the model if holding costs are as close to zero. It would not be the same case when the cost of storage becomes much more. Another observation is the slight drop every five days. This is due to the demand only being satisfied on the last working day of the week. The inventory therefore piles up through the week.

18. Proposed Implementation and Recommendations

The model's batch results are everchanging as uncertain parameters are realised after each week. Engineers, management, and the production planner all have their parameter changes that they attend to. The creation of simple interface is recommended for them to make the necessary adjustments.

The algorithm is written on Python and PuLP. Having an interface on Mecalc's company database that can link to the algorithm would be the best option for the business. The model will recalculate batch sizes and sequencing on a weekly basis when uncertain demand is determined. All the other changes in data – for example a prototype being added, a priority of a part changing, or simply when the machining time is found to be wrong – can be made during the week. The new batch sizes should only be sent to the production manager at the end of the week however. This is so he doesn't have added uncertainty and storage build-up.

The final recommendation for Mecalc would be to first test the model on changing data for about four weeks before it is implemented in the place of their current scheduling system. This gives them time to adapt to the new system.

19. Conclusion

The main goal of the current project is to address the volatile lot-sizing and scheduling problem Mecalc is faced with. Parallel machines have sequence dependent setups and the aim lies with minimising total production cost, setup time, and lead times. The industry currently plans and schedules their production without complex computerised tools. This led to inefficient planning and tardiness in demand satisfaction.

After reviewing relevant literature, two stochastic batch sizing models were programmed to find results. The stochastic recourse model with a multi-stage problem, and the probabilistically constrained stochastic batch-sizing problem (SBSP). After distinctively adjusting both for Mecalc's unique production process, the latter was determined as unfeasible and the former is chosen to be the base for the solution model. The solution model is clearly set out. It is set out with all its parameters, sets and constraints. After finding baseline results, sensitivity analyses were done to see how sensitive the model is to changes outside of the baseline. With this the parameters of the program were found, whereby the solution is validated.

Ending the report off is the implementation of the batch sizing system with Python and PuLP software and testing the system over a period of time. A possible area of future research would be to find more efficient algorithms to provide solutions in shorter time.

20. References

Chrétienne, P. et al., 2011. Integrated batch sizing and scheduling on a single machine Due date for customer order k. *J Sched*, 14, pp.541–555. Available at: https://link-springer-com.uplib.idm.oclc.org/content/pdf/10.1007%2Fs10951-011-0229-x.pdf [Accessed May 4, 2018].

Erlenkotter, D., 1990. *FORD WHITMAN HARRIS AND THE ECONOMIC ORDER QUANTITY MODEL*, Los Angeles, California.

Groenevelt et al., Production Batching with Machine Breakdowns and Safety Stocks. *Abraham Operations Research*, 40(5). Available at: https://search-proquest-com.uplib.idm.oclc.org/docview/219173599?OpenUrlRefId=info:xri/sid:wcdiscovery&accountid=14717 [Accessed May 1, 2018].

Holmes, D., 2017. *What is Stochastic Programming?*, Available at: http://users.iems.northwestern.edu/~jrbirge/html/dholmes/StoProIntro.html [Accessed April 26, 2018].

Hu, Z. & Hu, G., 2018. A multi-stage stochastic programming for lot-sizing and scheduling under demand uncertainty. *Computers & Industrial Engineering*, 119(August 2017), pp.157–166. Available at: https://doi.org/10.1016/j.cie.2018.03.033.

Kazaz, B. & Sloan, T.W., 2013. The impact of process deterioration on production and maintenance policies. *European Journal of Operational Research*, 227(1), pp.88–100. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0377221712009046 [Accessed May 1, 2018].

Lin, G.C. & Kroll, D.E., 2007. Economic lot sizing for an imperfect production system subject to random breakdowns. , 0273.

Potts, C.N. & Kovalyov, M.Y., 2000. Scheduling with batching: A review. *European Journal of Operational Research*, 120(2), pp.228–249. Available at: https://www-sciencedirect-com.uplib.idm.oclc.org/science/article/pii/S0377221799001538?_rdoc=1&_fmt=high&_origin=gateway&_docanchor=&md5=b8429449ccfc9c30159a5f9aeaa92ffb&ccp=y [Accessed May 4, 2018].

Sen, S. & Lulli, G., 2004. A Branch-and-Price Algorithm for Multistage Stochastic Integer Programming with Application to Stochastic Batch-Sizing Problems. *Management Science*, 50(6), pp.786–796.

Stuart Mitchell et al., 2009. The Optimisation Process — PuLP 1.6.0 documentation. Available at: https://pythonhosted.org/PuLP/main/the_optimisation_process.html [Accessed September 26, 2018].

Wagner, H.M. & Whitin, T.M., 1958. Dynamic Version of the Economic Lot Size Model. *Management Science*, 5(1), pp.89–96.

21. Appendices

A. Appendix A: Mecalc's production data

Table 5: Part Specifications and Parameters (part 1)

Part	HoldingCost	BackorderCost	MachiningTime	ProductionCostReg	ProductionCostOver
1	1	75.2	42	150.4	225.6
2	1	78.933	114	236.8	355.2
3	1	80.933	119	242.8	364.2
4	1	104	90	208	312
5	1	108.8	98	217.6	326.4
…					
200	1	202	85	202	303
…					
N					

Table 6: Part Specifications and Parameters continued (part 2)

Part	SetupTime	SetupCost	Priority	ReviewProb	Fragility[1]	Prototype
1	83	91.5	2	0.5	0.1	0
2	85	92.5	3	0.5	0.1	0
3	84	92	3	0.5	0.2	0
4	100	100	2	0.75	0.1	0
5	97	98.5	2	0.75	0.2	0
…						
200	68	84	1	0.5	0.1	0
…						
N						

Table 7: Part Demand per week

Part	week 0	week 1	week 2	week 3	week 4	week 5	week 6	week 7	week 8
1	0	50	0	0	0	0	0	0	0
2	0	108	0	0	0	0	0	0	0
…									
9	0	0	0	6	15	15	0	388	0
10	0	0	0	12	30	30	0	776	0
11	0	0	0	0	0	0	0	14	0
…									
200	0	0	0	0	0	0	0	0	50
…									
N									

[1] The fragility of a part indicates how easily a machine breakdown might occur. Partly because of the part's machining difficulty, but mostly caused by its physical fragility aspects.

Table 8: CNC Machine Properties

Machine	Breakdown probability (low, medium, high)	Capacity in minutes
1	medium	300
2	low	400
…		
14		

Table 9: CNC Machine Time Capacity per part per week

Week	MachineCapacity (minutes)[2]			
	1	**2**	**…**	**14**
1	720	720		720
…				
8	720	720		720

Table 10: Maximum Batch Sizes per part per week

Part	week 1	week 2	week 3	week 4	week 5	week 6	week 7	week 8
1	200	200	200	200	200	200	200	200
…								
N								

[2] Note that these 720 minutes of machine capacity were used for Model 2's results. The solution model addresses machine capacity on a weekly basis, changing these values.

B. Appendix B: Multi-stage Stochastic model (Hu & Hu 2018)

1. Parameters

$d_{i,t}$ Part i demand at time t

h_i Part i holding cost

b_i Part i backorder cost

cap_t Machine capacity time at time t

p_i Part i machining time

p_{ir} Part i production cost, regular time

p_{io} Part i production cost, overtime

$q_{i,t}$ Part i max batch size at time t, regular time

$sc_{i,j}$ Setup cost from part i to part j

$st_{i,j}$ Setup time from part i to part j

α Quantity ratio of regular and overtime production

N Amount of part families

2. Decision variables

$I_{i,t}$ Part i inventory quantity at end of time t

$B_{i,t}$ Part i backorder quantity at end of time t

$X_{i,t}$ Part i production quantity during time t, regular time

$O_{i,t}$ Part i production quantity during time t, overtime

$V_{i,t}$ Sequence of production in time period t. It takes value from 1 to N

$Y_{i,j,t}$ $\begin{cases} 1 \text{ if there is a changeover with a setup from part } i \text{ to part } j \text{ during time } t \\ 0 \text{ otherwise} \end{cases}$

$Z_{i,t}$ $\begin{cases} 1 \text{ if part } i \text{ setup is carried overto time } t \text{ from preceding time period} \\ 0 \text{ otherwise} \end{cases}$

3. Objective function

$$\min Z = \sum_{s=1}^{S} \nu_s * \left(\sum_{1=1}^{I} \sum_{t=1}^{T} p_i^r * X_{i,t,s} + \sum_{i=1}^{I} \sum_{i \neq j}^{I} \sum_{j=1}^{J} \sum_{t=1}^{T} Tsc_{i,j} * Y_{i,j,t,s} + \sum_{i=1}^{I} \sum_{t=1}^{T} p_i^o * O_{i,t,s} + \sum_{i=1}^{I} \sum_{t=1}^{T} h_i * I_{i,t,s} + \sum_{i=1}^{I} \sum_{t=1}^{T} b_i * B_{i,t,s} \right) \quad (1)$$

4. Constraints

$$X_{i,t,s} + O_{i,t,s} = d_{i,t,s} + I_{i,t,s} - B_{i,t,s}, \quad \forall i, t = 1, s \quad (2)$$

$$I_{i,t-1,s} - B_{i,t-1,s} + X_{i,t,s} + O_{i,t,s} = d_{i,t,s} + I_{i,t,s} - B_{i,t,s}, \quad \forall i, t = 2 \dots T+1, s \quad (3)$$

$$X_{i,t,s} \leq q_{i,t,s} * \left(Z_{i,t,s} + \sum_{j \neq i}^{J} Y_{j,i,t,s} \right), \quad \forall i, t, s \quad (4)$$

$$\sum_{i=1}^{I} p_i * X_{i,t,s} + \sum_{i=1 i \neq j}^{I} \sum_{j=1}^{J} st_{i,j} * Y_{i,j,t,s} \leq cap_t, \quad \forall t, s \quad (5)$$

$$O_{i,t,s} \leq \alpha * X_{i,t,s}, \quad \forall i, t, s \quad (6)$$

$$\sum_{i=1}^{I} Z_{i,t,s} = 1, \quad \forall t, s \quad (7)$$

$$Z_{i,t,s} + \sum_{j \neq i}^{J} Y_{j,i,t,s} = Z_{i,t+1,s} + \sum_{j \neq i}^{J} Y_{i,j,t,s}, \quad \forall i, t = 1 \dots T, s \quad (8)$$

$$X_{i,t,s} = 0, \qquad \forall i, t = T + 1, s \tag{9}$$

$$V_{j,t,s} \geq V_{i,t,s} + 1 - N * (1 - Y_{i,j,t,s}), \qquad \forall i, j \neq i, t, s \tag{10}$$

5. Model description

Equation (1) sets out the objective function – minimizing the total production cost. Constraints (2) and (3) are for the flow of the parts. The inventory can be extra, or it can be backlog demand, allowing it to be fulfilled later. Constraint (4) prohibits the production quantity to go beyond the max regular time quantity. Each product family permits only one setup, leading to constraint (4). Constraint (5) makes sure the capacity of the machine is not exceeded by regular time production. Constraint (6) sets the overtime capacity quantity limit. The beginning of each time period's setup carried over is set out by constraint (7). Constraint (8) shows the flow going into part i is equal to the flow going out of it. Constraint (9) states that there is no activity allowed in the last time period, as it is only the setup that is carried over to the next. Finally, constraint (10) allows no detouring of nodes; that one tour covers all the necessary steps to ensure completion.

C. Appendix C: Stochastic Batch Sizing Problem with probabilistic constrained parameters (Sen & Lulli 2004)

1. Parameters

b batch size

C_t number of batches production capacity at time t

I_t number of batches inventory capacity at time t

$d_{t,s}$ demand of scenario s at time t

$c_{t,s}$ production cost of scenario s at time t

$h_{t,s}$ holding cost of inventory of scenario s at time t

$f_{t,s}$ fixed setup cost of scenario s at time t

p_s scenario s probability

2. Decision variables

$X_{t,s}$ production batch level of scenario s at time t

$I_{t,s}$ inventory level of scenario s at time t

$Z_{H(t,s)}$ production quantity on node $H(t,s)$ of the scenario tree

$Y_{t,s}$ $\begin{cases} 1 \; if \; there \; is \; a \; setup \; for \; producion \; of \; scenario \; s \; at \; time \; t \\ 0 \; otherwise \end{cases}$

3. Objective function

$$minZ = \sum_{s=1}^{S} \sum_{t=1}^{T} p_s * \left(c_{t,s} * X_{t,s} + h_{t,s} * I_{t,s} + f_{t,s} * Y_{t,s}\right) \quad (11)$$

(11)

4. Constraints

$$I_{t-1,s} + b * X_{t,s} = d_{t,s} + I_{t,s}\,, \quad \forall t, s \quad (12)$$

$$X_{t,s} - Z_{H(t,s)} = 0, \quad \forall t, s \quad (13)$$

$$X_{t,s} \leq C_t * Y_{t,s}, \quad \forall t, s \quad (14)$$

$$I_{t,s} \leq b * I_t, \quad \forall t, s \quad (15)$$

$$Y_{t,s} \in (0,1), \quad X_{t,s} \in R, \quad I_{t,s} \in R, \quad \forall t, s \quad (16)$$

5. Model description

Constraint (12) is for balancing in the inventory, taking the demand and production into account as well. Constraint (13) shows that the production levels are not realized before the demand outcomes. The constraints (14) and (15) restrict the capacity. Y_t required to be one if there is production.

6. Added economical view with a change in constraint (12)

Constraint (12) states that any possible demand result should be covered by a large enough preceding production level. This can cause the system to become quite uneconomical, but it can be avoided. The scenario count can be restricted by adding a probabilistic constraint. It is formulated as followed:

q service level

μ_s $\begin{cases} 1 \; if \; scenario \; s \; is \; used \; in \; the \; solution \\ 0 \; otherwise \end{cases}$

The above results in the condition:

$$\sum_{s \in S} p_s * \mu_s \geq q \quad (17)$$

Giving the new optimal solution as:

$$\min Z = \sum_{s \in S, i \in G_s} p_s * c_{i,s} * \mu_s \tag{18}$$

Subject to:

$$\sum_{i \in G_s} X^i_{t,s} * \alpha^i_s - Z_{H(t,s)} \geq -M * (1 - \mu_s), \qquad \forall t, s \tag{19}$$

$$\sum_{i \in G_s} X^i_{t,s} * \alpha^i_s - Z_{H(t,s)} \leq M * (1 - \mu_s), \qquad \forall t, s \tag{20}$$

$$\sum_{s \in S} p_s * \mu_s \geq q, \qquad \forall s \tag{21}$$

$$\sum_{i \in G_s} \alpha^i_s - \mu_s = 0, \qquad \forall s \tag{22}$$

$$\alpha^i_s \geq 0, \qquad \forall i \in G_s, \qquad \forall s \tag{23}$$

$$\mu_s \in (0,1), \qquad \forall s \tag{24}$$

Where M is a very large number ($M \in R$) and $c_{i,s}$ indicates the ith column of scenario s.

D. Appendix D: Screenshots

```
import pandas as pd
import pulp
```

Data

```
parts = list(range(1,21))
weeks = list(range(1,9))
```

```
df1 = pd.read_csv('PartParams.csv')
df1.head()
```

```
demand = pd.read_csv('PartDemandPW.csv')
demand
```

```
time_params = pd.read_csv('TimeParams.csv')
time_params.head()
```

```
max_batch = pd.read_csv('MaxBatchSizes.csv')
max_batch.head()
```

```
# number of part families
N = 20
# quantity ratio of regular and overtime production (a = Overtime/RegularTime)
a = 0.6
```

Parameters

```
#demand
d = demand
#holding cost
h = df1['HoldingCost']
#backorder cost
b = df1['BackorderCost']
#machine capacity
cap = time_params['MachineCapacity']
#machining time
m = df1['MachiningTime']
#production cost, regular time
pr = df1['ProductionCostReg']
#production cost, overtime
po = df1['ProductionCostOver']
#max batch size
q = max_batch
#setup cost
sc = df1['SetupCost']
#setup time
st = df1['SetupTime']
```

Decision variables

```
I = pulp.LpVariable.dicts("I", (parts, weeks), lowBound=0, cat='Integer')
B = pulp.LpVariable.dicts("B", (parts, weeks), lowBound=0, cat='Integer')
X = pulp.LpVariable.dicts("X", (parts, weeks), lowBound=0, cat='Integer')
O = pulp.LpVariable.dicts("O", (parts, weeks), lowBound=0, cat='Integer')
V = pulp.LpVariable.dicts("V", (parts, weeks), lowBound=0, upBound=N, cat='Integer')
Y = pulp.LpVariable.dicts("Y", (parts, parts, weeks), cat='Binary')
Z = pulp.LpVariable.dicts("Z", (parts, weeks), cat='Binary')
```

```
model = pulp.LpProblem("Model 1", pulp.LpMinimize)
```

Objective function

```
# Regular production cost
sum1 = 0
for i in parts:
    for t in weeks:
        sum1 += pr[i-1] * X[i][t]

# Overtime production cost
sum2 = 0
for i in parts:
    for t in weeks:
        sum2 += po[i-1] * O[i][t]

# Setup cost
sum3 = 0
for i in parts:
    for t in weeks:
        for j in parts:
            if i!=j:
                sum3 += sc[i-1] * Y[i][j][t]

# Holding cost
sum4 = 0
for i in parts:
    for t in weeks:
        sum4 += h[i-1] * I[i][t]

# Backorder cost
sum5 = 0
for i in parts:
    for t in weeks:
        sum5 += b[i-1] * B[i][t]

model += sum1 + sum2 + sum3 + sum4 + sum5
```

Constraints

```
for i in parts:
    model += X[i][1] + O[i][1] == d.loc[i-1][1] + I[i][1] - B[i][1]
```

```
for i in parts:
    for t in range(2,9):
        model += I[i][t-1] - B[i][t-1] + X[i][t] + O[i][t] == d.loc[i-1][t] + I[i][t] - B[i
```

```
for i in parts:
    for t in weeks:
        six=0
        for j in parts:
            if i==j:
                continue
            six += Y[i][j][t]
        model += X[i][t] <= q.loc[i-1][t] * (Z[i][t] + six)
```

```
for t in weeks:
    one=0
    two=0
    for i in parts:
        one += m[i-1] * X[i][t]

        for j in parts:
            if i==j:
                continue
            two += st[i-1] * Y[i][j][t]

    model += one + two <= cap[t-1]
```

```
for i in parts:
    for t in weeks:
        model += O[i][t] <= a * X[i][t]
```

```
for t in weeks:
    three=0
    for i in parts:
        three += Z[i][t] == 1
    model += three
```

```
for i in parts:
    for t in weeks:
        Z[i][9] = 0
        model += Z[i][t] + pulp.lpSum([Y[i][j][t] for j in parts if i != j]) - pulp.lpSum([
```

```
for i in parts:
    model += X[i][8] == 0
```

```
for i in parts:
    for j in parts:
        if i==j:
            continue
        for t in weeks:
            model += V[j][t] >= V[i][t] + 1 - N*(1 - Y[i][j][t])
```

Solution

```
model.solve()
```

```
pulp.solvers.LpSolver(model)
```

```
pulp.LpStatus[model.status]
```

```
output = []
for i in range(10,20):
    for t in weeks:
        var_output = {
            'Part': i,
            'Week': t,
            'Production': round(X[i][t].value()),
            'Overtime production': round(O[i][t].value()),
            'Inventory': round(I[i][t].value()),
            'Demand': d.loc[i-1][t]
        }
        output.append(var_output)
output_df = pd.DataFrame.from_records(output).sort_values(['Part', 'Week'])
output_df.set_index(['Part', 'Week'], inplace=True)
print(output_df)
```

Figure 10: Multi-stage Stochastic model (Hu & Hu 2018), "Model 1"

```
df2 = pd.read_csv('Model 2 data.csv')
df2
```

```
C = df2['C']
I = df2['I']
d = df2['d']
c = df2['c']
f = df2['f']

n = len(df2)
```

```
times = list(range(1,8))
times
```

```
inv = pulp.LpVariable.dicts("inv", (times), lowBound=0, cat='Integer')
X = pulp.LpVariable.dicts("X", (times), lowBound=0, cat='Integer')
Y = pulp.LpVariable.dicts("Y", (times), cat='Binary')
```

```
# EXTRA INPUTS
h = 1
b = 30
```

```
my_problem = pulp.LpProblem("Model 2", pulp.LpMinimize)
```

Objective function

```
sum1 = 0
for t in df2.index+1:
    sum1 += c[t-1] * X[t]

sum2 = 0
for t in df2.index+1:
    sum2 += f[t-1] * Y[t]

my_problem += sum1 + (h * sum(inv)) + sum2
```

Constraints

```
my_problem += inv[1]==80
for t in range(2, 8):
    my_problem += inv[t-1] + ( X[t]) == d[t-1] + inv[t]

for t in df2.index+1:
    inv[0]=0
    my_problem += X[t] <= C[t-1] * Y[t]
    my_problem += inv[t] <= I[t-1]
```

```
my_problem.solve()
```

```
pulp.LpStatus[my_problem.status]
```

```
output = []
for t in df2.index+1:
    var_output = {
        'Time': t,
        'Production batch level': X[t].value(),
        'Inventory level': inv[t].value(),
        'Is production': Y[t].value(),
        'Demand': d[t-1]
    }
    output.append(var_output)
output_df = pd.DataFrame.from_records(output).sort_values(['Time'])
output_df.set_index(['Time'], inplace=True)
output_df
```

Figure 11: SBSP (Sen & Lulli 2004), "Model 2"

In [1]:
```
import pandas as pd
import pulp
```

In [4]:
```
from mss import mss
```

In [6]:
```
with mss() as sct:
    sct.shot()
```

In [2]:
```
params = pd.read_csv('PartParams.csv')
params.head()
```

In [3]:
```
demand = pd.read_csv('PartDemandPD.csv')
demand.head()
```

In [4]:
```
time_params = pd.read_csv('TimeParams.csv')
time_params.head()
```

In [5]:
```
max_batch = pd.read_csv('MaxBatchSizes.csv')
max_batch.head()
```

In [6]:
```
#demand
d = demand
#holding cost
h = params['HoldingCost']
#backorder cost
b = params['BackorderCost']
#machine capacity
cap = time_params['MachineCapacity']
#machining time
mach = params['MachiningTime']/600
#setup cost
sc = params['SetupCost']
#setup time
st = params['SetupTime']/600
```

In [7]:
```
B = 350
```

In [8]:
```
h=1
```

In [9]:
```
parts = list(range(1, 41))
N = parts[-1]
machines = list(range(1, 8))
times = list(range(1,21))
```

In [10]:
```
I = pulp.LpVariable.dicts("I", (parts, list(range(0,21))), lowBound=0, cat='Integer')
X = pulp.LpVariable.dicts("X", (parts, machines, times), lowBound=0, cat='Integer')
V = pulp.LpVariable.dicts("V", (parts, machines, times), lowBound=0, cat='Integer')
Y = pulp.LpVariable.dicts("Y", (parts, machines, list(range(1,22))), cat='Binary')
Z = pulp.LpVariable.dicts("Z", (parts, parts, machines, times), cat='Binary')
```

In [11]:
```
model = pulp.LpProblem("Model 5", pulp.LpMinimize)
```

Objective function

In [12]:
```
sum1 = 0
for m in machines:
    for i in parts:
        for j in parts:
            for t in times:
                sum1 += sc[j-1] * Z[i][j][m][t]

sum2 = 0
for i in parts:
    for t in times:
        sum2 += h * I[i][t]
```

In [13]:
```
model += sum1 + sum2
```

Figure 12: Solution model

Constraints

In [14]:
```
# inventory balance
for i in parts:
    model += I[i][0]==0
    for t in times:
        model += I[i][t] == I[i][t-1] - d.loc[i-1][t] + pulp.lpSum([X[i][m][t] for m in machines])
```

In [15]:
```
# production and setup vs cap
for m in machines:
    for t in times:
        a = 0
        for j in parts:
            a += st.loc[j-1]*pulp.lpSum([Z[i][j][m][t] for i in parts])

        model += pulp.lpSum([mach.loc[i-1]*X[i][m][t] for i in parts]) + a <= 1
```

In [16]:
```
# force binaries
for i in parts:
    for m in machines:
        for t in times:
            model += X[i][m][t] <= B*(pulp.lpSum([Z[i][j][m][t] for j in parts]) + Y[i][m][t])
```

In [17]:
```
# setup carry over
for m in machines:
    for t in times:

        model += pulp.lpSum([Y[i][m][t] for i in parts]) == 1

        for i in parts:
            b = 0
            c = 0
            for j in parts:
                b += Z[i][j][m][t]
                c += Z[j][i][m][t]

            model += Y[i][m][t+1] + b == c + Y[i][m][t]
```

In [18]:
```
# eliminating disconnected subtours
for m in machines:
    for t in times:
        for i in parts:
            for j in parts:
                if i==j:
                    continue
                model += V[i][m][t] + N*Z[i][j][m][t] - (N-1) - N*Y[j][m][t] <= V[j][m][t]
```

Solution

In [19]:
```
model.solve()
pulp.LpStatus[model.status]
```

Out[19]: 'Infeasible'

In [20]:
```
pulp.value(model.objective)
```

Out[20]: 673.7248845400002

In [21]:
```
output = []
for m in machines:
    for i in parts:
        for t in times:
            var_output = {
                'Machine': m,
                'Part': i,
                'Time': t,
                'Production': round(X[i][m][t].value()),
                'Inventory': round(I[i][t].value()),
                'Demand': d.loc[i-1][t],
                'Sequence': V[i][m][t].value(),
                'Y': Y[i][m][t].value(),
            }
            output.append(var_output)
output_df = pd.DataFrame.from_records(output).sort_values(['Time', 'Machine', 'Part'])
output_df.set_index(['Time', 'Machine', 'Part'], inplace=True)
print(output_df)
```

Figure 13: Solution model (continued)