



BPJ 420

PARALLEL MACHINE SCHEDULING PROBLEM WITH SEQUENCE
DEPENDENT SETUP TIMES: A CASE STUDY AT A WHEAT MILL

INDUSTRIAL ENGINEERING STUDENT
MARCO CROUCAMP
13035101

SEPTEMBER 27, 2016

[Final report]




 DEPARTEMENT BEDRYFS- EN SISTEEMINGENIEURSWESE
 DEPARTMENT OF INDUSTRIAL AND SYSTEMS ENGINEERING

VOORBLAD VIR INDIVIDUELE WERKOPDRAGTE - 2016	
FRONT PAGE FOR INDIVIDUAL ASSIGNMENTS - 2016	
Persoonlike besonderhede / Personal details	
Studentenommer Student number	13035101
Voorletters en van Initials and surname	M Croucamp
Titel Title	Mnr.
Selnommer Cell number	071 364 9383
Werkopdrag / Assignment	
Modulekode Module Code	BPJ 410
Werkopdragnommer Assignment number	Final project report
Onderwerp Subject	Multiple machine scheduling problem with independent product setup and flushing times
Dosent Lecturer	Dr. J Grobler.
Datum Date	23/09/2016
Verklaring / Declaration	
1. Ek begryp wat plagiaat is en is bewus van Universiteitsbeleid in hierdie verband 2. Ek verklaar dat hierdie my eie oorspronklike werk is 3. Waar iemand anders se werk gebruik is (hetsy uit 'n gedrukte bron, die internet of enige ander bron), is dit behoorlik erken en die verwysings ooreenkomstig departementele vereistes gedoen 4. Ek het nie 'n ander student se vorige werk gebruik en as my eie ingedien nie 5. Ek het niemand toegelaat en sal niemand toelaat om my werk te kopieer met die doel om dit as sy of haar eie werk voor te hou nie	
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	
Handtekening Signature	<i>Mroucamp</i>
Datum van inhandiging Date of submission	19/03/2016
Kantoorgebruik / For office use:	
Dosent Lecturer	Kommentaar / Comments:
Uitslag Result	
Datum Date	

Executive Summary

Scheduling is a key factor for delivering a quality and reliable product. The sudden interest in scheduling problems over the past forty years emphasizes the new opportunity created by using scheduling tools (Kır and Yazgan 2016). Scheduling tools enable companies around the world to minimize non-value added factors like setup times, setup cost and changeovers. On time delivery of products are achieved by optimizing the scheduling of production,(Gupta and Chantaravarapan 2008). This project focuses on a wheat mill in Silverton, Gauteng. This report considers a parallel machine scheduling problem, with sequence dependent setup times for the production of flour products. The total demand of each job must be processed at the same time, not allowing preemption. The primary objective of the schedule is to minimize the total production time.

A Mathematical programming formulation shall form the basis of solving the problem. Five heuristic rules are used. Results were obtained by running all of the heuristic rules over thirty random demand scenarios. The optimal heuristic rule was determined as the process with the most robustness to change in input data. In this project the largest flushing times heuristic rule performed the best.

The heuristic chosen as the best can easily be implemented by the company. No additional resources have to be bought. The solution have been tested against real world data and delivered excellent results. The current run time for the best heuristic rule is 0.0005 seconds.

The current scheduling method will schedule all the demand in approximately 23.9 days. The new heuristic rule scheduling method will be able to produce all the demand in just 18.73 days. The financial impact of implementing the optimal heuristic rule saves the company up to R653.00 on electricity, R430.00 on water and R18 000.00 on overtime per day. That lead to a total savings of R19 083.00 per day. The new heuristic will eliminate four days of production. Equaling the total savings to R76 332.00 for four days.

Contents

Executive Summary	ii
List of Figures	v
List of Tables	v
1 Introduction	1
1.1 Introduction	1
1.2 Background	1
1.3 Project Aim and Rationale	3
1.3.1 Potential benefits	4
1.4 Approach	4
1.4.1 IE techniques that can be applied	5
1.5 Scope	5
1.6 Deliverables	6
2 Literature review	7
2.1 The problem environment (α)	7
2.2 Constraints linked to jobs (β)	9
2.2.1 Sequence dependent setup times	10
2.2.2 Availability of resources.	11
2.3 Criteria that must be optimized (γ)	11
2.4 Problem statement	12
2.4.1 Premier foods optimization problem	12
2.5 Assumptions	14
2.6 Possible solution strategies	14
2.6.1 Introduction to solution strategies	14
2.6.2 Minimum seeking algorithms	17
2.7 Metaheuristic	18
2.8 The genetic algorithm	18
2.8.1 Simple genetic algorithm	19
3 Data analysis	20
3.1 From to chart	20
3.2 Production rates and demands	20
4 Formulating and solving the Premier foods sequence dependent scheduling problem	22
4.1 Model formulation	22
4.2 Heuristic pseudocode and description	25
4.2.1 New heuristic pseudocode	26
4.2.2 Random Qualitech production order	27
4.2.3 Order Qualitech range so that the total flushing time is a minimum	27
4.2.4 Order Qualitech range that the total flushing time is a maximum	28
4.2.5 Order Qualitech range from smallest demand to largest demand	28
4.2.6 Order Qualitech range from largest demand to smallest demand	29

5	Results	30
5.1	Current heuristic	30
5.1.1	Random ordering of the Qualitech products	30
5.1.2	Scheduling Qualitech to obtain the minimum flushing times	30
5.1.3	Scheduling Qualitech to obtain the maximum flushing times	31
5.1.4	Scheduling the smallest demands for Qualitech first	31
5.1.5	Scheduling the largest demands for Qualitech first	31
5.2	Finding the best heuristic rule based on total number of production days . .	33
6	Verification and validation	36
6.1	Verification	36
6.2	Sensitivity analysis	37
6.2.1	Sensitivity analysis on the product demands	38
6.2.2	Sensitivity analysis on production rates.	39
6.2.3	Sensitivity analysis on mixer mixing times.	39
6.2.4	Sensitivity analysis on Qualitech flushing times.	40
6.2.5	Further analysis on the baseline heuristic	41
6.3	Project achievements	42
7	Conclusion	43
7.1	Summary and impact of project	43
7.2	Future work	43

List of Figures

1	Range of Easymix product produced at Premier foods.	2
2	Process mapping of the processes in the project scope.	3
3	Breakdown of different job shop models (Zandieh, Fatemi Ghomi, and Moattar Hussein 2006).	8
4	Breakdown of the possible solution strategies (Grobler 2008).	15
5	Actual schedule for largest flushing times.	35
6	Heuristic output for producing Easymix range.	37
7	Sensitivity analysis on product demands.	38
8	Sensitivity analysis on production rates.	39
9	Sensitivity analysis on mixer mixing times.	40
10	Sensitivity analysis on Qualitech flushing times.	41
11	Results from running the random rule over 1000 iterations.	42

List of Tables

1	Example of popular constraints in a scheduling problem.	10
2	Formulation of the general optimization criteria.	12
3	Current MS Excel model used for scheduling.	13
4	Current MS Excel model used for scheduling.	13
5	Categorization of optimization.	17
6	Example of flushing times from Product (i) to (j).	20
7	Production processing units per production line.	21
8	Demand per product.	21
9	Notation used for Mixed Integer Programming model.	23
10	Symbols used in heuristic pseudocode.	26
11	New model using the minimum flushing times rule.	30
12	New model using the maximum flushing times rule.	31
13	New model scheduling the smallest demands first.	31
14	New model scheduling the largest demands first.	31
15	New model summary of results.	32
16	New model summary of results with days saved.	33
17	New model summary of results.	34
18	Current model formulas for computing total time needed per month.	36
19	New model using the Random rule for 30 iterations.	54

List of abbreviations

JIT	Just in time.
JSSP	Job shop scheduling problem.
SMSP	Single machine scheduling problem.
FSSP	Flow shop scheduling problem.
PMSP	Parallel machine scheduling problem.
HFSSP	Hybrid flow shop scheduling problem.
SDST	Sequence dependent setup times.
GA	Genetic algorithm.

Chapter 1

1 Introduction

1.1 Introduction

Scheduling problems have been studied for decades by numerous mathematicians and scientists. The focus of this report is on sequencing products with sequence dependent setup times on parallel machines. The effectiveness of a just in time (JIT) production strategy requires ordering, sequencing and scheduling of man, machine and resources.

All of the possible solutions obtained from the model, that is near optimal, have been analysed against specified metrics. The solution was evaluated by the impact it has on the primary objective; minimize the time required for production. The best solution was then further evaluated against its robustness to changes in input data, a sensitivity analysis was conducted on the best heuristic rule and results are shown in latter sections.

1.2 Background

Premier foods is a national supplier of flour products to enterprises. The Premier group operates sixteen bakeries, five wheat mills, two maize mills, a sugar confectionery plant and twenty four distribution depots in South Africa, Swaziland and Lesotho (*Premier foods* n.d.). The project focuses on one of the wheat mills.

The three production lines focused on consist of: Two Easymix lines which produces 500g and 1kg products, respectively. The 500g product range consist of eleven different products while the 1 kg product range only consist of seven products. The third line that is focused on in this project is the Qualitech line consisting of sixteen respective products. Although the Qualitech line produces sixteen products the demand for those products is extremely low and does not take precedence over the Easymix line.



Figure 1: Range of Easymix product produced at Premier foods.

The mixer in line can mix one ton of raw material at a time. The mixing time is nine to ten minutes. A production line worker opens one or two lines to feed the respective lines with raw material. Both Easymix lines (500g and 1kg) can be fed together as some of the products are produced in both 500g and 1kg packets. The Qualitech line must be fed separately as this is a completely different product. The worker then decides which of the three lines to feed. He opens the sluices for those lines and the raw materials flow through.

Following successful mixing the materials are then collected in a bin, which is simply a container made from Aluminium with a slope to guide materials to the feeding hole at the bottom. The capacity of the bins are as follows: The Qualitech bin has a capacity of one ton and the Easymix lines both have a capacity of two tons. The raw material is then fed to their respective production machines. These three machines are then fed by their three respective lines and bins. The production capacities for these machines were obtained by interviewing the production manager. These capacities are 37 units per minute for the Easymix 500g line, 27 units per minute for the Easymix 1kg line and 38 units per minute for the Qualitech line. The production machine is responsible for opening the empty packet, filling it with the correct number of ingredients, glueing it and folding the packet. Then, via a conveyer belt, the packets are sent for manual inspection. Following the completion of manual inspection twelve packets are sorted and sealed together in one container. The scope of the processes covered by the project are shown in Figure 2.

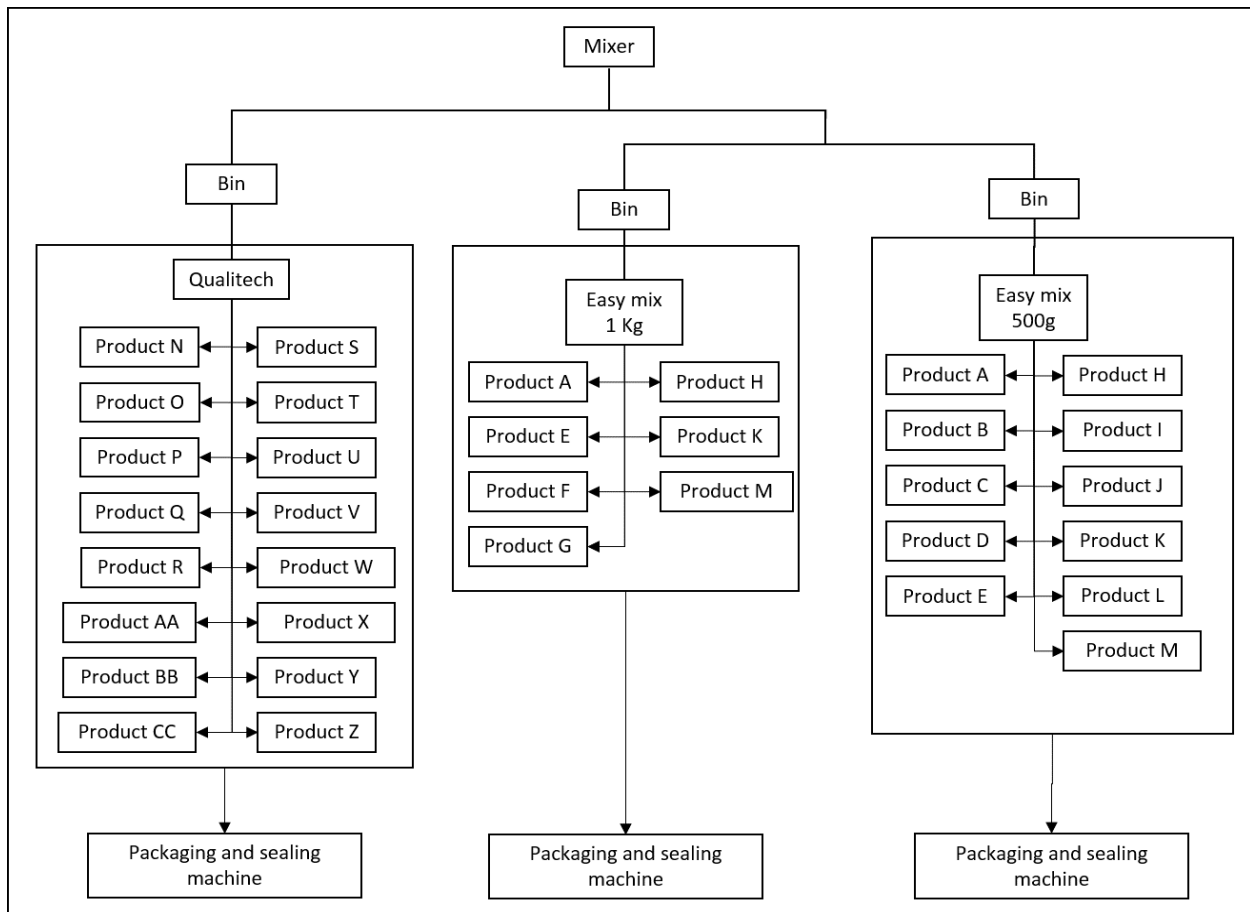


Figure 2: **Process mapping of the processes in the project scope.**

The production of a new product can only happen after the whole system is flushed. The flushing process includes running clean flour through the whole system from the mixer to the processing machine. The clean flour is then recycled. The amount of clean flour needed to flush the system depends on the previously produced product. An average of 200kg of clean flour is required. This flushing process is a time consuming exercise as the clean flour that is running through the system must leave the system as if it is a product to be produced. In other words, the clean flour follows the same processes that a real product would follow. This cleaning process also requires the same production time. Therefore, to run 200kg of clean flour through the system would take the same time as to produce 200kg of finished product. The goal is to minimize the flushing time, but while minimizing the flushing time the production schedule must also provide a (JIT) production schedule (Laguna and Velarde 1991).

1.3 Project Aim and Rationale

The aim of the project was developing a model that will schedule the required demand for a month and optimize the time the mixer, as it is the bottleneck, is used. By optimizing the use of the mixer the total production time can be reduced by 4 days, given the best heuristic rule is used for scheduling.

The current scheduling method will need 23.9 days to produce the demand required per month, the new heuristic rule will be able to produce the same demand in only 18.73 days. This will have a total savings of R76 332.00 per month over the current scenario. The project is important for Premier foods as this project helps improve the production scheduling for the time period specified, along with the options of exploiting new capacities on the production line. Annually Premier foods demand is increasing, increasing the production efficiency will increase production capacity and subsequently meet the market demands.

Premier foods will not be able to meet future demands unless it increases efficiency or invest in more equipment, this project investigates how much an increase in efficiency can benefit Premier foods.

1.3.1 Potential benefits

Production time, idling time, changeover time and flushing time will be shown to provide financial benefits. These non-value adding activities will be minimized by the heuristic to be able to free up time during the month in order to cancel the Saturday and Sunday shifts. The cancellation of the weekend shifts will have a positive financial impact as the salaries for weekend versus weekdays are 1.5 to 2:1. The schedule will help eliminate the overtime schedules on weekend by simply setting reasonable targets. If those targets are not met it is an indication that workers might be idling. The reason that they would be idling is simply the increase in overtime salaries.

The technical benefits related to this project will be the two aspects to be tested and analyzed. These two aspects involves firstly the addition of a secondary mixer which will allow for a quicker changeover between two products and reduce the time lost on flushing as the one mixer can function independently from the other, but on the same or a different production line. The other key aspect to be considered is the unlocking of capacity on the Easymix bins. The current capacity of the bins are two tons. There is still room within the building to enlarge these bins to three or four tons. Enlarging the bin capacities can reduce the constraints on the bottleneck.

The manufacturing benefits will focus on the number of hours lost during flushing time, maintenance time and changeover time. The bottleneck in optimizing the production time has been found to be the time taken to flush the whole system with clean materials, subsequently, implementing schedule optimization will limit the time wasting changeovers and reduce production times. The other benefit that optimization will add to the production is the fact that the schedule will ensure that demand orders are met at the specified date required, leaving the company with a better perfect order filling rate. Better scheduling will also result in better utilization of resources.

1.4 Approach

The project includes an in depth literature study on the different techniques used during the project. The literature study consists of different methods and techniques used for solving the Premier foods problem.

In solving the problem a model was developed based on the constraints of the problem. A basic mathematical model was developed which would be able to obtain the exact solution to the problem. However using mathematical models will not be efficient as they require significant amount of time to solve. Heuristics were thus used to obtain an approximate solution in a given period of time. Additional heuristic rules were then developed to improve on the previous heuristic. The parameters of the most optimal heuristic were also optimized in relation to the input parameters. These parameters were optimized using a sensitivity analysis.

The final solutions obtained from the program will then be evaluated against different metrics. If some of the processes needs to be adjusted management can evaluate the cost and impact to implement the suggested solution. Management can then make an informed decision on whether to proceed with the schedule or find a new one.

1.4.1 IE techniques that can be applied

The engineering techniques used in this project included the implementation of techniques such as just in time inventory (JIT). The JIT principle can now be better implemented in the factory as the workers now exactly which product is going to be produced next. Raw materials can now be order to arrive JIT. Cost benefit analysis shall be used in the project. The program was developed using simpler heuristic models working together and newly developed heuristics based on the specified rules of the company. Operation research techniques such as the development of algorithms, modeling of a mathematical program and sensitivity analysis will be used along with operations management techniques.

The next section shall cover the scope of the project, including project activities, tasks and deliverables.

1.5 Scope

The scope of the project shall be restricted to three different productions lines and their output. The processes preceding the production process such as the grinding, sifting and filtering the wheat will not be included.

Additionally no transferring methods or warehouse optimization shall be considered. The scope of the project consist strictly of the different processes described in Chapter 1. Information obtained from existing literature shall be used to analyse the problem and identify solution strategies.

The sample space for solutions are very large thus the development of a heuristic based algorithm shall be necessary. This algorithm will be developed and fully functional. The algorithm shall be developed using freeware that is always available to the company. A list of processes and jobs that are in scope are listed below:

1. The mixing of the raw materials.
2. Transferring of the raw materials to bins.
3. Processing of raw materials fed by the bin.

The following process will not be included in the project and falls outside the scope:

1. The preceding processes before mixing.
2. Forecasting demand.

1.6 Deliverables

The deliverables for this project include the following:

- An in depth literature review on similar problems.
- Heuristic based solution for scheduling the production cycles. The heuristic approach will be able to provide a near optimal solution providing the constraints, yet feasible in real life.
- The project will evaluate the different solutions for improvement on throughput.
- Presentation on the solution, methods and recommendations.
- The project will also provide recommendation for further solutions that can be reviewed as further developing ideas.

Report outline: The structure for the rest of the document will be organized as follows: Chapter 2 will contain a detailed literature review on problem classification and possible approaches to solving the problem. The problem statement will be described in the section thereafter as well as the literature based on possible solution strategies. Chapter 3 shall be the data analysis and Chapter 4 includes the formulation of the Premier foods scheduling problem. Chapter 5 will be used to reflect on the results obtained from the five heuristic rules, Chapter 6 will include the verification and validation of the heuristic rules and Chapter 7 will discuss the impact of the project and recommendations for further research.

Chapter 2

2 Literature review

Optimization of parallel machines production schedule has already been studied with respect to many different variables. These variables include; due dates, earliness tardiness, penalty costs, production time, flushing times and maintenance times. Solving a sequencing problem have also been studied by multiple researchers. A review on solving scheduling problems identified the following variables that may have an impact: total production time, tardiness and due dates (Kır and Yazgan 2016) .

Problem classification is of vital importance. The more accurate the classification of the problem, the easier it is to find methods to solve the problem. In order to classify a problem we distinguish between typology and notation. The typology refers to the classification of the problem in terms of its nature, which includes the machine's environment and jobs related to the machines (T'kindt 2006). Notation helps to differentiate a specific problem. Typologies of scheduling problems already exist in the literature. The notation most commonly used for classifying scheduling problems were introduced by Graham et al.(1979). The notation consists of three fields: $\alpha/\beta/\gamma$. Field α describes the structure of the problem. The structure consists of two parts: α_1 the machines environment and α_2 the number of available machines. Field β consists of the constraints of the problem. Field γ specifies the specific criteria that must be optimized.

Introduction to the notation and additional symbols are defined below. Denote n jobs by J_i ($j = 1, \dots, n$) where jobs J_i have to be processed on m machines M_i ($i = 1, \dots, m$). Lets assume that one machine can only process one job at any time and that each job can be processed on at most one machine at a given time. The following data is specified for each J_i :

- A number of operations M_i : one or more processing times p_j or p_{ij} .
- A release date r_i , on which the job becomes available for processing.
- A due date d_i , by which the job must be completed.
- A non-decreasing real cost function f_i , measuring the cost $f_i(t)$ incurred if J_i is completed at time t .

In general $M_i, p_i, p_{ij}, r_i, d_j$ and w_i are integer variables.

2.1 The problem environment (α)

The first field of the notation for classifying scheduling problems identifies the machine's environment. Given the first field $\alpha = \alpha_1\alpha_2$ specifying the machine environment. Let 0 denote the empty symbol. If $\alpha_1 \in \{0, P, Q, R\}$, each job J_i consist of a single operation that can be processed on any machine M_i , then the processing time of J_i on M_i is p_{ij} .

The first field of (Graham et al. 1979) focuses on the environment of the machines and the flow pattern of jobs. The α component can be subdivided down as follows:

1. A single machine environment: M_i , where $i = 1$.
2. Identical parallel machines: $p_{ij} = p_j, (i = 1, \dots, m)$.
3. Uniform parallel machines; $p_{ij} = q_i p_j$ for a given speed factor q_i of $M_i, (i = 1, \dots, m)$.
4. Unrelated parallel machines.

These different models are classified into primary groups;

- Characteristics of the routes followed by each job,
- Number of operations required by each job,
- Number of resources available to perform the required operations.

The different models can be further extracted using the primary groups listed above. The different models can be observed using the figure below.

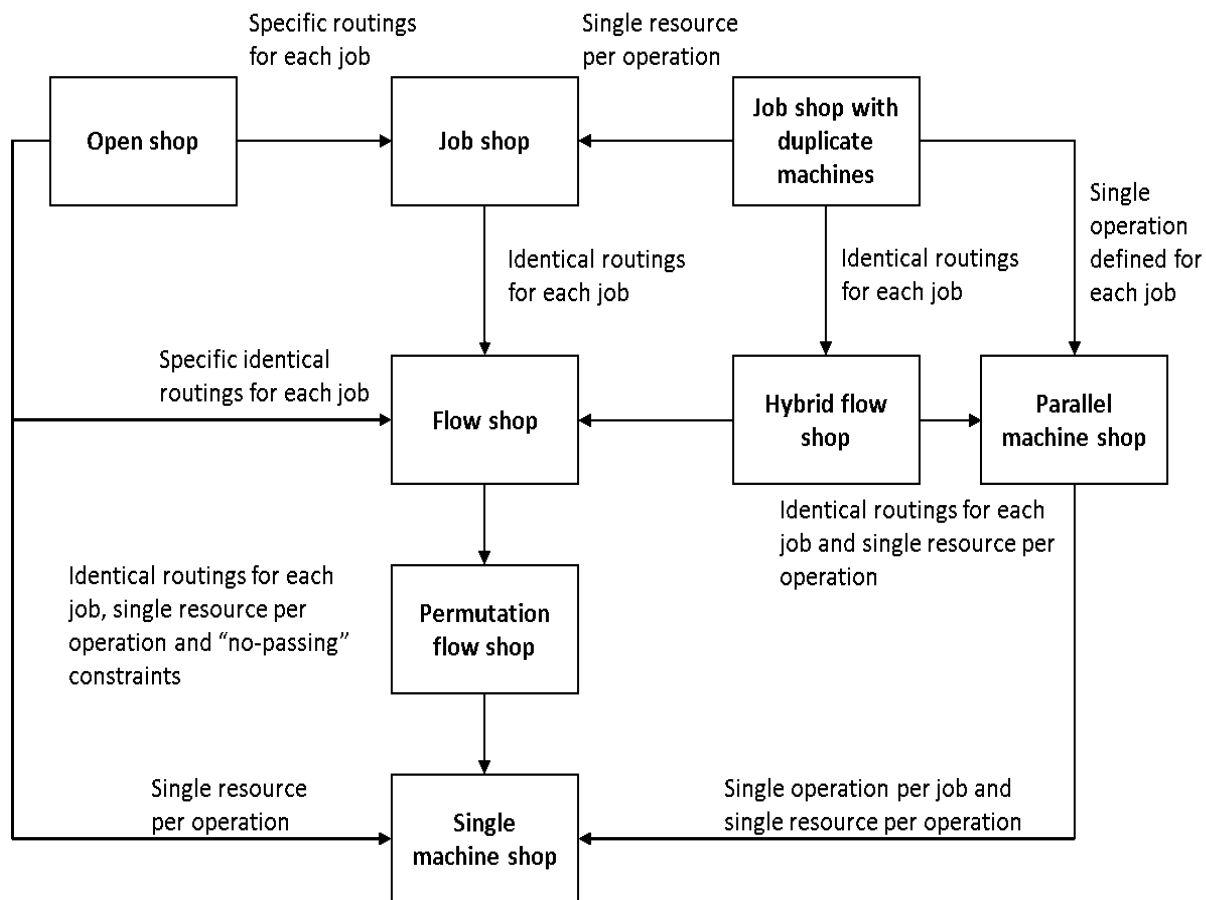


Figure 3: Breakdown of different job shop models (Zandieh, Fatemi Ghomi, and Moattar Hussein 2006).

The most common scheduling problem in this classification is the job shop scheduling problem (JSSP). All classifications of models can be described by specific instances of the JSSP. General rules and constraints are adapted from the solutions of a JSSP (Kolisch and Hartmann 1999).

A single machine scheduling problem (SMSP) and flow shop scheduling problem (FSSP) can be generalised to a specialised JSSP (Grobler 2008). The SMSP is a branch of the JSSP where the number of operations are limited to one per job. The FSSP is exactly the same as a JSSP with the added constraints of specified applications where identical routes are defined for each job. The permutation flow shop is linked to the FSSP with the addition of the no passing constraint. The no passing constraint ensures that only one job sequence is followed for an individual resource. When solving either of these problems, the algorithm would have to find the order in which the jobs should be scheduled.

In contrast to the JSSP, the parallel machine shop problem (PMSP) is restricted to jobs only requiring one operation which may be executed by any of the resources. The last type of problem related to the job shop is the hybrid flow shop scheduling problem (HFSSP). The HFSSP uses multiple resources per operation over identical routes. Other generalization of the classic JSSP include the duplicate machine JSSP, where the system consist of two identical resources.

2.2 Constraints linked to jobs (β)

The constraints that have an influence on the jobs must be identified before solving the problem. Graham et al. (1979) have established a decent platform to work from when defining these constraints. Constraints include the allowance or dis-allowance of preemption (job splitting). If preemption is allowed the processing of any operation can be completed at a later time.

The number of resources must be determined and usage of resources must be constrained to ensure that no more than 100% of the resources are used. Setup times, dependent and independent, will form part of the constraints linked to a job. Permutation constraints, one job may never overtake another job or itself, is applicable for flow shop problems. With regards to line balancing, machines must complete jobs assigned to them in the sequence they were assigned.

Depending on the type of problem at hand it may be necessary to add precedence relationships between jobs. The precedence relationship will ensure that the specified job will be completed before the next job may start. Certain problems lend themselves to add a constraint for the release date. The last constraint to consider is the processing time per unit for each operation, which may differ for each machine.

Abbreviation	Description
pmtn	Authorizing preemption or not.
split	Splitting of jobs are authorised.
prec	Operations are connected by specific preceding constraints.
batch	Indicates that jobs are joined to form batches.
no-wait	No time lapse between different jobs.
prmu	One job may not overtake another job in the production flow.
$d_i = d$	All due dates are identical.
$P_i = P$	All processing times are identical.
Snsd	Setup time for each resource (independent of the sequence)
Rnsd	Removal time per resource (independent of the sequence)
Ssd	Setup time for each resource (dependent on the sequence)
Rsd	Removal time per resource (dependent on the sequence)
blcg	Balancing, machines must complete jobs assigned to them
block	Referring to the stock capacity between machines
recre	Indicates that a single job may be processed more than once
unavailj	Indicates that all the resource machines are unavailable

Table 1: **Example of popular constraints in a scheduling problem.**

The next two sections shall be cover sequence dependent setup times and availability of resources. These two topics are directly related to the Premier foods scheduling problem. The availability of resources play a vital role in scheduling production. Jobs cannot be scheduled on machines that are inactive or under maintenance. The Premier foods problem consist of processes which have sequence dependent setup times, such as the flushing processes between jobs.

2.2.1 Sequence dependent setup times

Chang stated that the setup time, raw materials and equipment necessary to prepare for the following job is dependent on the preceding job, i.e. the setup times are sequence dependent. Sequence dependent scheduling problems (SDSP) tend to be the most difficult optimization problem to solve (Chang, Hsieh, and Wang 2003). The JSSP is another version of the general JSSP.

Solving a sequence dependent JSSP leads to solving a NP-hard problem, which means the use of heuristics and approximation algorithms are necessary (Moghaddas and Houshm 2008). In complex cases of a sequence dependent problem matrices are required for each resource.

In contrast to sequence dependent setup times, are sequencing with independent setup times. Independent setup times problems occur when the preceding job does not have an effect on the setup times for the next job. This may be because the setup times between the jobs are the same.

2.2.2 Availability of resources.

The availability of resources can be described as the duration of up-time, which is a measure of how often the system is up and running (Barringer 1997). It can also be expressed mathematically as with many different variations. Up-time refers to the capability to produce jobs while downtime refers to the lack of capability to produce jobs. Three frequently used availability terms are explained next:

Inherent availability refers to the probability that a resource shall be functional at a given time period, when used in an ideal environment. It excludes preventive maintenance, supply delays and admin delays. Inherent availability can be calculated using the formula (Barringer 1997):

$$\text{Inherent - availability} = \frac{(MTTF)}{(MTTF) + (MTTR)} \quad (1)$$

where $MTTF$ is mean time to failure and $MTTR$ is mean time to repair.

Achieved availability includes both types of maintenance, corrective and preventive, but it does not include delays caused by administration or supply. The calculation provided by Barringer (1997) states that:

$$\text{Achieved - availability} = \frac{(MTBM)}{(MTBM) + (MAMT)} \quad (2)$$

where $MTBM$ is mean time between maintenance and $MAMT$ is mean active maintenance time.

Operational availability includes time used for logistics, setup time, waiting time and administrative downtime. The operational availability is equal to:

$$\text{Operational - availability} = \frac{(MTBF)}{(MTBF) + (MDT)} \quad (3)$$

where $MTBF$ is mean time before failure and MDT is mean down time.

2.3 Criteria that must be optimized (γ)

The third field of problem classification $\gamma \in \{fmax, \sum fi\}$, with the optimization criterion chosen, in this case schedule optimization. A General criterion set by Graham et al. (1979) includes the minimization of total completion time, where total completion time refers to the total time it would take to produce the jobs. Given a schedule we can compute for each J_i .

Lateness of demands must be minimized. The lateness of a product is the time needed to finish the job after the due date has passed. Other criteria include the minimization of tardiness and minimization of penalty costs. The last mentioned criteria are very problem specific and may not be applicable to all problems. Specific problems may require the optimization of yet another criteria, which must then be identified.

The most commonly chosen optimal criteria are the minimization of completion time and lateness. Table 2 shows the general criteria used in a simple optimization problem. These include: total production times, lateness of jobs, tardiness and penalties involved. The formulation for these criteria are shown below.

Criteria	Formulation
Total production time of job i	C_i
Lateness of job i with due date d_i	$L_i = C_i - d_i$
Tardiness of job i	$T_i = \max(0, C_i - d_i)$
Penalty cost of job i	$U_i = \text{if } C_i \leq d_i, \text{ then } 0, \text{ else } 1.$

Table 2: **Formulation of the general optimization criteria.**

This concludes the notation of classifying the problem. The next section will introduce the Premier foods scheduling problem. Before attempting to solve the problem the problem has to be fully understood.

2.4 Problem statement

2.4.1 Premier foods optimization problem

The problem that has been identified at Premier foods is: Currently Premier foods is calculating the time needed for production in MS Excel. Table 3 provides an example of this model. They are currently scheduling the Easymix range first and then the Qualitech range afterwards. This leaves the mixer with idle time between the large Easymix demands, as mixing is a lot quicker than the actual production.

The development of a scheduling model will lead to improved productivity, minimum inventory between processes and the on time delivery of quality and reliable products (Gupta and Chantaravarapan 2008). The model was based on the given constraints of the company, which may include preemption, splitting of jobs, setup times, line balancing and capacity blocks of the resources. The mill currently has only one mixer feeding three parallel production lines. The problem is between the mixer and the production lines, at the bins, which have a limited capacity. The schedule must ensure that those bins are frequently supplied with raw material from the mixer. The problem is that the one mixer must feed all three bins and on occasion two of the lines require the same raw material which would be simple but most of the time all three lines require different raw materials. The schedule will assist the mixer to optimally feed all three lines to ensure there is no lack of raw materials on any line.

Production rate	T/H				
500G	1.17		Days Required Easymix and Qualitech		
1KG	1.74		23.906		
Qualitech	Units/ min		Hours required		
1KG	38		1434.33		
Start production for the month with Easymix 500g range					
SKU		Tons	Hours	Flushing	Total Time
Product A	500 G	26.30	22.48	1.00	23.48
Product B	500 G	22.53	19.26	0.50	19.76
Product C	500 G	19.54	16.70	0.50	17.20
Product D	500 G	8.56	7.32	0.50	7.82
Product E	500 G	79.97	68.35	1.00	69.35
Product F	1 KG	32.22	18.52	1.00	19.52
Product G	1 KG	22.35	12.84	2.00	14.84
Product H	500 G	43.50	37.18	1.00	38.18
Product I	500 G	9.92	8.48	7.00	15.48
Product J	500 G	29.63	25.32	5.00	30.32
Product K	500 G	87.36	74.67	1.00	75.67
Product L	500 G	53.88	46.05	4.00	50.05
Product M	500 G	73.52	62.84	5.00	67.84
			420.0	29.5	449.50

Table 3: Current MS Excel model used for scheduling.

After the Easymix have been produced they start scheduling the Qualitech					
SKU		Units	Hours	Flushing	Total Time
Product N	1 KG	30000	13.16	0.29	13.45
Product O	1 KG	22200	9.74	0.29	10.03
Product P	1 KG	44600	19.56	0.29	19.85
Product Q	1 KG	46400	20.35	0.29	20.64
Product R	1 KG	38800	17.02	0.29	17.31
Product S	1 KG	6600	2.89	0.29	3.19
Product T	1 KG	4600	2.02	0.29	2.31
Product U	1 KG	4000	1.75	0.29	2.05
Product V	1 KG	3800	1.67	0.29	1.96
Product W	1 KG	3400	1.49	0.29	1.78
Product X	1 KG	4600	2.02	0.29	2.31
Product Y	1 KG	36600	16.05	0.29	16.34
Product Z	1 KG	29000	12.72	0.29	13.01
Product AA	1 KG	13400	5.88	0.29	6.17
Product BB	1 KG	0	0.00	0.00	0.00
Product CC	1 KG	0	0.00	0.00	0.00
			120.44	3.79	124.23

Table 4: Current MS Excel model used for scheduling.

2.5 Assumptions

The assumptions made in formulating the problem are:

1. Production is made-to-stock and there are no due dates assigned to jobs.
2. All jobs and machines are available at the beginning of the scheduling process.
3. The production lines consist of multiple stages. Each stage may have non-identical but uniform machines. The stages in this problem include: When the material is in the mixer and when the material is in the bins.
4. Setup time for jobs on each machine is dependent on the order in which the jobs are processed.
5. No job splitting is allowed. Once a job is being processed it must be completed on the same machine without stoppage.
6. No job preemption is allowed. No jobs may be interrupted for example joining of two jobs on one machine.

The next sections shall address the possible solutions for the Premier foods scheduling problem. A brief overview of strategies will be provided along with recommendations of the best strategies. The use of heuristics and metaheuristics will also be reviewed.

2.6 Possible solution strategies

An optimization model works like a hiker, searching for the smallest fluctuation in elevation per period but aiming to reach the top of the mountain. The model will begin at the bottom of the mountain gradually working upwards to the top. Just like hiking there is going to be elements interrupting the model. These will come in the form of constraints and local optimal points. Pure uphill or downhill approaches tend to fail due to the fact that the model needs to find out which peak is in fact the highest or which hole is the deepest. Good overviews of optimization models can be found in the literature (Cuthbert 1987).

2.6.1 Introduction to solution strategies

Optimization algorithms adjust the input variables of a process to obtain the optimal output result. Inputs of optimization problems include; variables, rules, processes and resources. These inputs have to be managed in order to find the optimal solution. Optimization forms the primary tool in the intellectual toolbox (Haupt and Haupt 2004).

Optimization is the process of improving, until no more improvements are possible. Whilst attempting to solve the Premier foods scheduling problem different strategies would have to be considered. Strategies with the most appropriate methodology in solving the problem will be considered. This section will provide an overview of the possible strategies such as optimal solutions strategies and other methods that have been used in the literature to solve similar problems.

Possible solution strategies are shown in the Figure 4. Indicating the relationship between these approaches and the sequence in which they are performed (Grobler 2008).

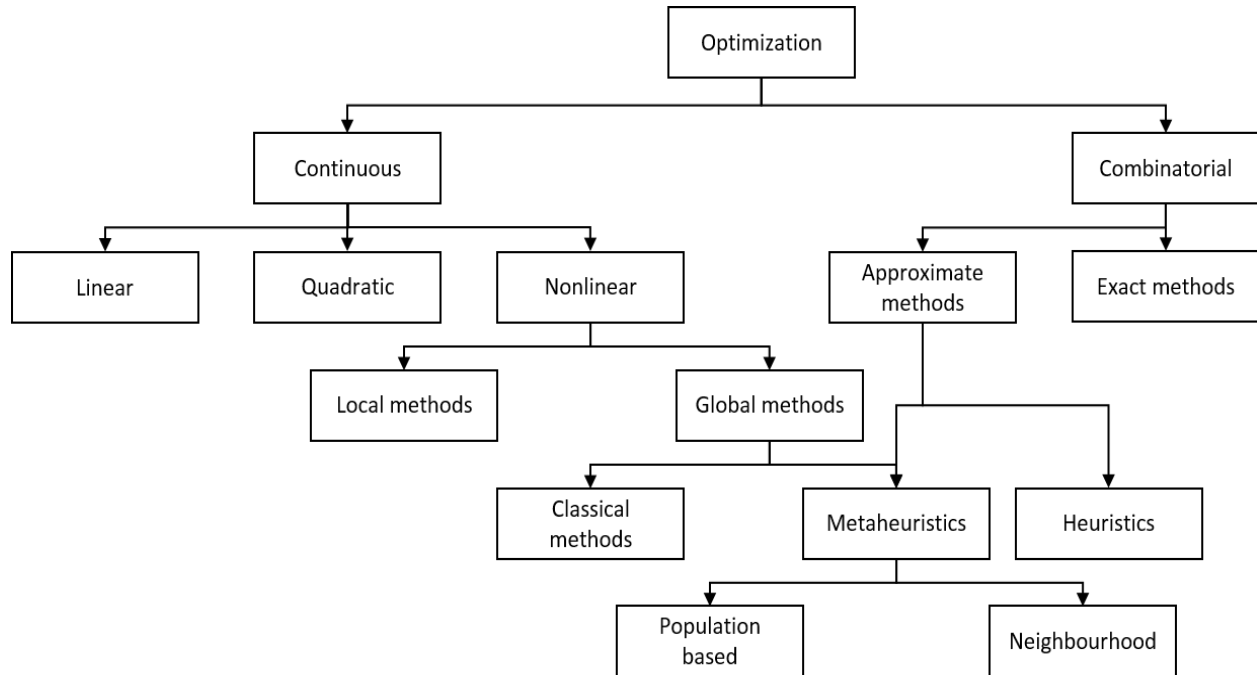


Figure 4: **Breakdown of the possible solution strategies (Grobler 2008).**

After the realization that this is an optimization problem the following question must be answered. Is this a continuous or combinatorial problem? Continuous problems have a infinite sample space where combinatorial or discrete problems have a finite known sample space. The Premier foods scheduling problem has a finite sample space which categorizes it as a combinatorial problem.

The Premier foods problem can be solved using exact methods based on mathematics, however these methods will not be as effective when considering the time it takes to calculate the optimal solution. In industry time is money. Premier foods do not have excessive time to solve a scheduling problem each day, week or month. The Premier foods scheduling problem will, therefore, not be solved using exact methods. The Premier foods scheduling problem is a np-hard problem, thus the problem cannot be solved to one optimal solution.

Finding the "best" solution The term "best" have been described by (Haupt and Haupt 2004), it implies that there are other alternative solutions, but they are not equal in quality. The best solution will depend on the type of problem, the method of solution and the specific tolerances allowed. Optimization algorithms can find either an exact solution or near optimal solutions. Exact solutions will be found where the solution space is relevantly small and all constraints can be addressed fully.

Near optimal solutions are found when the solution space is large and there may be more than one possible solution. The possible different solutions have to be compared to identify

the optimal solution. The optimal solution will not necessary adhere to all of the constraints, but it would be the best possible solution in a specified time period.

A number of general techniques that can be applied to the Premier foods problem are described below (Feo and Resende 1995). The first two methods are used to generate solutions for the constraints given. The exchanging methods is used to improve the current solution, optimizing the initial solution obtained.

- **Random methods:** The heuristic will return any random solution obtained to the user. The reason why this step is import is that the heuristic can be repeated multiple times. The different solutions can then be evaluated by the user allowing the user to choose the preferred solution. The advantages to this approach are that the heuristic can find hundreds of possible solutions in a given period. In the same period of time another heuristic may only find one. The random solution method provides the user with an almost limitless source of new solutions to improve. The negative side of using this method is that the solutions provided by a small number of random solutions will most likely be far from optimal.
- **Greedy methods:** The greedy method will initiate the program with no initial values assigned to the variables. As the program executes values are assigned to variables. The value that has been assigned to the variable is the value that minimizes the increase of the objective function. The greedy method has the property that once a decision is made, that decision will never be changed. The greedy method is more accurate than the random method. The disadvantage of the greedy method is the lack of randomization which leads to only one or a few solutions.
- **Exchanging methods:** The exchange method replaces a certain number of variables with predetermined values, this is called exchanging. The changing of certain variables will not change the entire solution, it will only change parts of it. This method is used to narrow down the solutions provided to the optimal solution or near optimal solution.

Another categorization of optimization Optimization can be divided into six main branches. Different optimization problems can be constrained or unconstrained and variables may be discrete or continuous. Table 5 summarizes the six main groups of optimization with an explanation of each regarding the category (Haupt and Haupt 2004).

Category	Description
Trial and error	The process of adjusting the input variables to obtain the desired output variable. An example may be, searching for the best signal with a radio antenna.
One dimensional	Only one variable to optimize.
Dynamic optimization	The Output of the system is a function of time, in contrast static is independent of time. An example may be the distance of a trip (Static) <i>vs</i> the fastest route (Dynamic).
Discrete / Continuous	Variables used to optimize a process can be combinatorial or continuous. Combinatorial variables have a finite number of values where continuous have infinite number of values.
Constraint variables	Constraints variables to a certain series of values where an unconstrained variable is allowed to take on any value.
Minimum seekers	Minimum seeking algorithms usually try to minimize the cost function to a minimum, they are generally based on mathematical equations. Minimum seekers is usually fast.

Table 5: **Categorization of optimization.**

2.6.2 Minimum seeking algorithms

Exhaustive search The exhaustive search (ES) approach tend to consider a relatively small sample. This approach requires a great number of cost function evaluations to be able to obtain the best solution. The methodology include creating a list of function values generated over the sampled variables. After the list has been created the minimum value in the list is identified as the optimal value.

The ES does the surveying in order to produce a topographic map of the solutions. This requires the model to check an extremely large finite solution space. To prevent the ES approach from getting stuck a small sample is required. The larger the sample size the greater the chance of getting stuck and the longer it would take to find the optimal solution. In contrast under sampling is also a problem due to the long solving times. To summarize ES approaches are practical for smaller samples with a small number of variables and a finite search space (Haupt and Haupt 2004).

Analytic Optimization Calculus provides many ways for finding the minimum value for a cost function. The method of solving a analytic problem will be to narrow down the variables, the extreme value can then be found by setting the first derivative of the cost function equal to zero. In order to classify the extreme value as a minimum or maximum the second derivative must be calculated. If the second derivative is greater than zero, the extreme value is a minimum, and contrarily if the value of the second derivative is less than zero it is a maximum. Two or more variables can be solved by calculating the gradient of the function and setting it equal to zero. The downside of using this approach is that the answer obtained does not indicate if this is the global minimum or global maximum. This approach is more elegant than the exhaustive search. A list can be created with all the minimum or maximum values and a search function can determine the global maximum or global minimum.

Analytic optimization is constrained to continuous functions (unless the derivatives are calculated numerically, but accuracy decreases). Similar to ES, if there are too many variables it can be difficult to find the extreme points. The gradient of the cost function leads the way when searching for global points. This approach does not deal well with cliffs or boundaries but is effective when the global point searched for is close (Haupt and Haupt 2004).

Many disadvantages of this approach make it an unlikely candidate to solve optimization problems. Although it is impractical, most numerical approaches are based on the calculus used in analytic optimization. Popular algorithms start with random points, calculate the gradients and head downhill slowly. The analytic approach head downhill fast, finding the local minimum and not the global minimum. The other disadvantage in using this approach is that analytic optimization does not work well with discrete variables.

Other candidate methods also categorized as minimum seeking algorithms is the Nelder-Mead downhill simplex method. This method was spurred with the invention of computers. Another approach was introduced as: optimization based on line minimization. A more in depth literature of these and the above models can be found in Haupt and Haupt (2004).

2.7 Metaheuristic

Metaheuristics form a major sub field of stochastic optimization (Luke 2013). Stochastic optimization forms part of the general algorithms which uses a degree of randomness for their approach. Metaheuristics are applicable to problems which at first is not clear, and can be used to solve problems with very little information. Very little heuristic information is available and the use of exhaustive search is not applicable as the sample space is too large. To be able to find the optimal solution the model will have to evaluate all the possible solutions found and assess how good they are.

One of the most popular metaheuristics is the genetic algorithm. The following section include a basic formulation for the genetic algorithm.

2.8 The genetic algorithm

Genetics along with natural selection forms the principle on which genetic algorithms (GA) are based. A GA allows a population of values to evolve under certain rules and constraints, until a steady state is reached. John Holland developed the GA method in the 1960's, popularization of the method only came when one of his students used it to solve difficult problems (Goldberg and Holland 1988).

Advantages of using a genetic algorithm as stated by Haupt and Haupt (2004) include:

- Able to optimize discrete as well as continuous variables,
- Does not require any derivative calculations,
- Searches a wide sample of cost functions,

- Suited for solving with parallel computers,
- Ability to jump out of a local extreme,
- Provides a series of optimum values, not just a single solution.

2.8.1 Simple genetic algorithm

Solving a well defined problem using a GA is as simple as these few steps described by Mitchell (1998).

1. Use a random generated population for the basis of the GA, the population consists of nl - bit chromosomes. These form the candidate solutions.
2. Calculate the degree of fitness $f(x)$ for each chromosome x in the population.
3. Repeat the preceding steps until n children have been produced:
 - (a) Select one pair of parent chromosomes from the current population, with a higher probability of selecting greater fitness candidates. Note that the same chromosome can be selected more than once to become a parent.
 - (b) Probability p_c (crossover probability), cross over the parent pair randomly chosen to form children. In the case where no crossover takes place two offspring form which are direct copies of each parent.
 - (c) Mutate the two children at each locus with the probability being p_m (mutation probability), place the resulting children in the new population. If the number of values n in the population is odd, one new member can be discarded at random.
4. Substitute the new population in the space of the old population.
5. Repeat step 2.

Every iteration of these five steps are called a generation. A typical GA will consist of 50 to 500 or more generations. All the generations are called a run. More detail is necessary for instance population size, probabilities of crossover and probabilities of mutation. The effectiveness of this approach is dependent on the availability of these details. In depth examples of GA's can be revised in Mitchell (1998).

To conclude the GA approach and search methods, all search methods start with a random candidate in the initial population. The candidate will be evaluated according to the fitness criteria, a decision will be made on whether to keep or discard the candidate. The model will search for further variants by using the best candidates until a near optimal solution is reached.

The next chapter will be covering the data analysis of the project. Input data in this project consists of the flushing times, demand and production rates.

Chapter 3

3 Data analysis

Data relevant to the problem were obtained using time studies and interviews. The flushing times of the system from one product to another were obtained by observation with a stopwatch. The flushing times hadz to be analysed in order to identify a relationship. The relationship of flushing time from one product to another is best described using a from to chart.

3.1 From to chart

Data are plotted on a from to chart (J L Riggs, Wiley, 1987) to indicate the relationship between the different products. For example, flushing time from Product A to Product B will be 2.5 hours. A realistic example of a from to chart is shown below. All the actual flushing times can be seen in Appendix C.

From to	Product A	Product B	Product C
Product A	-	2.5	0
Product B	0	-	0
Product C	0	0	-

Table 6: **Example of flushing times from Product (i) to (j).**

The rows and columns have exactly the same titles in the same sequence. The entries in the chart describes the time lapsed to flush the system, from Product i to Product j .

The data in the chart were used by the algorithm to identify the optimal process to produce next. A typical rule that could be implemented is to schedule the product with the smallest (from to) flushing time next.

3.2 Production rates and demands

Data relevant to the production rates, product demands, shift lengths, maintenance time, and down time due to fumigation of the plant where obtained by interviewing the production manager. This data will also be used for scheduling production.

Production rates refer to the number of units that can be produced by a single line. These production rates differ from each line as the size of the packages differ. This data will be used in the heuristic to calculate the time it would take to produce a certain number of products. Production rates are listed per line as shown in Table 7.

Production line	Production rate (Units/min)
Easy mix 500g	37
Easy mix 1kg	27
Qualitech	38

Table 7: **Production processing units per production line.**

The product demand per month were also available. Demand were used alongside the production rates to calculate the time it would take to produce the demand. Demands were available by product name as shown in Table 8. All of the actual demands can be seen in Appendix C.

Product	Demand (Tons)
Product A	370
Product B	400
Product C	310

Table 8: **Demand per product.**

Shift lengths are a fixed period given by the company. They currently run three shifts. The shifts are each eight hours long starting from six in the morning.

The following chapter shall cover the formulation of the Premier foods sequence dependent scheduling problem. Included in the next chapter is an in depth mathematical model followed by a psuedocode model. If necessary the python code is available in the appendices.

Chapter 4

4 Formulating and solving the Premier foods sequence dependent scheduling problem

This chapter provides a formulation for the Premier foods problem and represents various heuristics which can be used to solve the problem. The new model will decrease the number of days needed to produce both Easymix and Qualitech demand. The heuristics were also tested to decide which one is the closest to optimal.

4.1 Model formulation

The exclusion of exact methods indicates that the problem will be solved using heuristic approximation methods. In this way a solution will be obtained in a shorter time span. Although exact methods will not be used to solve the problem it is still beneficial to first obtain a mathematical model and then develop a heuristic program. The formulating of an exact mathematical model will provide a better understanding of the problem. The subsection that follows presents a mixed integer programming model with the objective of minimizing the make span of the problem provided by Sethanan n.d.

The mathematical model that follows is an exact model to solve the Premier foods schedule problem. The table below will be used to denote the objective function as well as the following constraints.

Type of variable	Notation	Description
Decision variables	$FT(j, i, s, m)$	Finish time of product i , family j on machine m of stage s .
	E	Make span (total time).
Binary decision variables	$x(j, i, s, m)$	= 1, if product i , family j is assigned to machine m of stage s . = 0, otherwise.
	$w(j, i, q, p, s, m)$	= 1, if product i , family j immediately precedes product p , family q on machine m of stage s . = 0, otherwise.
Parameters	i, p j, q s m f_j $m(s)$ N $M(s)$ S $PT(j, i, s, m)$ $ch(j, i, q, p, s)$	Product indices. Family indices. Stage index. Machine index. The number of products in family j . The number of machines on stage s . Total number of families. The set of machines in stage s ; $M(s) = \{1, 2, \dots, m(s)\}$ The number of stages in the production line. The processing time of product i , family j on machine m of stage s . The time units required to changeover from product i , family j to product p , family q at stage s .

Table 9: Notation used for Mixed Integer Programming model.

Objective function: Min E

Constraints: Completion time constraints: (Ensures that all products are scheduled and that the initial setup times for the machines are included in the processing time)

$$FT(j, i, 1, m) \geq ch(0, 0, j, i, s) + \{PT(j, i, 1, m) \times x(j, i, 1, m)\} \quad (4)$$

where $j = 1, 2, \dots, N$; $i = 1, 2, \dots, f_j$; and $m = 1, 2, \dots, m(1)$

Stage link constraints: Ensures that the completion time of the preceding time is greater than the completion time in the previous stage. The difference must be equal to the amount of processing time required for the current stage.

$$FT(j, i, s, m) \geq FT(j, i, s - 1, mp) + \{PT(j, i, s, m) \times x(j, i, s, m)\} \quad (5)$$

where $j = 1, 2, \dots, N$; $i = 1, 2, \dots, f_j$, $s = 2, 3, \dots, S$, $m = 1, 2, \dots, m(s)$, and $mp = 1, 2, \dots, m(s - 1)$

Constraints for product sequencing on all the S stages:

$$FT(j, i, s, m) - FT(q, p, s, m) - ch(q, p, j, i, s) + (V)(1 - w(q, p, j, i, s, m)) \quad (6)$$

$$\geq \{PT(j, i, s, m) \times x(j, i, s, m)\}$$

where $j, q = 1, 2, \dots, N; i = 1, 2, \dots, f_j, p = 1, 2, \dots, f_q; s = 1, 2, \dots, S, m = 1, 2, \dots, m(s)$, and V is a very large positive value.

Sequence completion time constraint: Ensures that the make span is equal to or greater than the completion time of each of the products in the last stage.

$$FT(j, i, S, m) \leq E \quad (7)$$

where $j = 1, 2, \dots, N; i = 1, 2, \dots, f_j$; and $m = 1, 2, \dots, m(S)$

The following constraint ensures that each product is processed on exactly one machine in each stage.

$$\sum_{m=1}^{m(s)} x(j, i, s, m) = 1 \quad (8)$$

where $j = 1, 2, \dots, N; i = 1, 2, \dots, f_j$; and $s = 1, 2, \dots, S$

Except for the first product, a product must be scheduled directly after another product.

$$x(j, i, s, m) - w(j, i, 0, 0, s, m) - \sum_{q=1}^N \sum_{p=1}^{f_j} w(j, i, q, p, s, m) = 0 \quad (9)$$

where $j = 1, 2, \dots, N; i = 1, 2, \dots, f_j$; and $s = 1, 2, \dots, S$, and $m = 1, 2, \dots, m(s)$

A machine may only have one first and one last product.

$$\sum_{q=1}^N \sum_{p=1}^{f_j} w(0, 0, q, p, s, m) = 1 \quad (10)$$

where $s = 1, 2, \dots, S$; and $m = 1, 2, \dots, m(s)$

Although an optimal solution can be obtained using the exact method, the computation time would be extremely long. In order to obtain the best solution as close to the optimal solution, heuristic algorithms were developed. The solution would be categorized as a near optimal solution and could be obtained in a much shorter time period.

Approximation methods include the use of heuristics, metaheuristics and classical methods. The approach followed in solving the Premier foods scheduling problem involves the use of heuristics. The following section includes the pseudocode for the Premier foods sequence dependent scheduling problem.

4.2 Heuristic pseudocode and description

This section of the report will cover the pseudocode for solving the Premier foods scheduling problem. The code will be accommodated with a brief description as to why this method is appropriate and other possible alternatives. The different methods will be described in the direct order in which they will be implemented.

All the heuristics proposed in this report will first schedule the Easymix range of products. The Easymix range is scheduled to incur the least amount of flushing times. Easymix, being the main product range, is therefore scheduled first. The Easymix demand are greater and has to be delivered as soon as possible. Qualitech is almost a tenth of the demand of Easymix and is a special product, not main stream or fast moving product. After the Easymix range has been scheduled the model will calculate the time the mixer is idle. The time the mixer is idle will then be the total mixer idle time during the production of each individual Easymix product. After the scheduling of the Easymix range the different mixer idle times will be stored in a list.

The algorithm will now use one of the heuristic rules to determine the order in which the Qualitech range will be produced. These rules include random, smallest inter-changeover times, smallest demands first, largest demands first and largest inter-changeover times. These rules will be described in more depth later.

After the algorithm has generated an order in which to schedule the Qualitech range the model will then start to schedule the Qualitech demands into the mixer idle time slots, calculated earlier. If all the demands cannot be scheduled in the open slots, the model will simply schedule the remaining Qualitech demands after the Easymix products as is done in the current MS Excel model. In the model this time will be labeled as "extra time", as this time is incurred at the end of the schedule.

4.2.1 New heuristic pseudocode

The table below gives the descriptions for the symbols used in the pseudocode.

Symbol	Definition
E_{D1}	Easymix 500g demands
E_{D2}	Easymix 1kg demands
Q_D	Qualitech demands
E_F	Easymix flushing times
Q_F	Qualitech flushing times
E_{P1}	Easymix production rate for 500g
E_{P2}	Easymix production rate for 1kg
Q_P	Qualitech production rate
M_T	Mixing time per product
E_{PT1}	Production time needed for Easymix 500g per product
E_{PT2}	Production time needed for Easymix 1kg per product
M_I	Mixer idle time
Q_T	Time needed to produce Qualitech demand per product
$Xtra$	Extra time needed to produce Qualitech
TP	Total production time needed to produce all the demands

Table 10: Symbols used in heuristic pseudocode.

```

//Initialization phase
1  Input data  $\leftarrow$  (demands, flushingtimes, production – rates);
2  Store data  $\rightarrow$  [list];

//Loop until the terminal condition
3  for  $i$  in range(0,len( $E_{P1,2}$ ):
4     $M_T \leftarrow E_{P1,2}(9min)$ ;
5    Mixing time list.append( $M_T$ );
6  end

//Determine whether the 500g or 1kg of the same products production time is greater
7  Use  $max[E_{PT1,2}]$ ;
8   $max[E_{PT1,2}] \leftarrow E_{D1,2} / E_{P1,2} + E_F$ ;
9   $M_I \leftarrow \sum E_{P1,2} - \sum M_T$ ;
10 list.append( $M_I$ );

//Scheduling Qualitech products in mixer idle time slots
11  $Q_P \leftarrow 38$ ;
12 %  $M_I \leftarrow 85\%$ ;
13  $Q_D \leftarrow list.(Q_D)$ ;
14  $Q_F \leftarrow Q_F.array$  ;

//Use heuristic rule to generate Qualitech production order
15 import heuristic rule.();

```

```

//Calculate time needed to produce Qualitech products
16  for i in range(0,len(QD));
17    QT ← QD / QP;
18  end;

//Fit the Qualitech demands in the mixer idle times
19  for k in range(0, len(QD));
20    if QD ≤ MI;
21      list.append(QD in MI);
22    if MI == 0;
23      list.append(QT);
24  end;

//Calculate total time needed to produce products
25  EPT1,2 ← ∑ EPT1,2;
26  QT ← ∑ QT;
27  Xtra ← QT - EPT1,2;
28  Xtra ← ∑ Xtra;
29  TP ← ∑[EPT1,2 + QT] + Xtra;
30  return TP;

```

For further information of the pseudocode, the python code along with in depth descriptions is provided in Appendix C.

4.2.2 Random Qualitech production order

The random production ordering rule will simply shuffle the order in which the Qualitech products will be produced. All of the 16 Qualitech products will be scheduled randomly. Below is the pseudocode for this rule as it is used in the main model. The random rule will be used as a baseline. In order to determine whether the solutions obtained from other rules are of better quality, they will be tested against the random rule's minimum value.

```

//Produce random order for Qualitech products
1  Qualitech-production-order ← [1, ..., 16];
2  random.shuffle(Qualitech-production-order);

```

4.2.3 Order Qualitech range so that the total flushing time is a minimum

Ordering the Qualitech range from one to sixteen will be equal to the minimum total flushing time as the products are currently scheduled to minimize flushing times. they are producing the products from limited flavor to the product with the strongest flavor. For example pancake will be low in flavor and chocolate mint will be rich in flavor.

The flushing time required from a product with limited flavor to a product with more flavor is small. While the opposite is true for a product with adequate flavor would incur a large flushing time. The pseudocode for this rule is simple as the rule only needs to schedule the product from one to sixteen. See below for the pseudocode.

```

//Produce order for Qualitech products with minimum flushing times
1  QD ← [1, ..., 16];
2  for i in range(0, 16);
3      Order-number ← i;
4      Qualitech-production-order.append(i);
5  end;

```

4.2.4 Order Qualitech range that the total flushing time is a maximum

As discussed above, to flush after a product with more flavor would incur a large flushing time. Thus, in order to create a production order with the largest flushing times the products will need to be ordered from sixteen to one. This implies that the production order will now run from the product with the most flavor to the product with the least amount of flavor.

The pseudocode for this rule is simply the reverse order from the previous rule, the products is in order from sixteen to one. See below for pseudocode.

```

//Produce order for Qualitech products with maximum flushing times
1  QD ← [1, ..., 16];
2  for i in range(0, 16);
3      Order-number = i;
4      Qualitech-production-order.append(i);
5  reverse.Qualitech-production-order;
6  end;

```

4.2.5 Order Qualitech range from smallest demand to largest demand

This rule will enforce products with the smallest demands to be scheduled first. The pseudocode for this rule is shown below. This rule will ignore the other rule of smallest flushing times and will only consider the demands.

```

//Produce order for Qualitech products for smallest demands first
1  QD ← [1, ..., 16];
2  for i in range(0, 16);
3      minimum ← min(QD);
4      Qualitech-production-order.append(minimum);
5      QD.remove(minimum);
6  end;

```

4.2.6 Order Qualitech range from largest demand to smallest demand

This rule will enforce products with the largest demands to be scheduled first. The pseudocode for this rule is shown below. This rule will ignore the other rule of smallest flushing times and will only consider the demands.

```
//Produce order for Qualitech products for largest demands first  
1   $Q_D \leftarrow [1, \dots, 16]$ ;  
2  for  $i$  in range(0, 16);  
3      maximum  $\leftarrow \max(Q_D)$ ;  
4      Qualitech-production-order.append(maximum);  
5       $Q_D$ .remove(maximum);  
6  end;
```

Chapter 5 will look at the results obtained for the five heuristic rules. These rules have been tested against thirty random demand sets. The best rule will then be analysed to its robustness with input data.

Chapter 5

5 Results

5.1 Current heuristic

This section will include results for every rule used to minimize the total period needed to produce the demand. Later in this section the best alternative will be explored and further analysis on the best alternative will follow. The current model will be able to complete all of the demands for Easymix as well as Qualitech in a period of 23.906 days. The newly developed model must be able to complete the same number of jobs and demands in a shorter period.

5.1.1 Random ordering of the Qualitech products

The random order generator rule has been run thirty times (central limit theorem) in order to obtain the average, median, minimum and maximum values for the total time to produce the Qualitech range, extra time needed and the total duration of the whole schedule. These values are shown in Table 19 in Appendix D.

The results from the random iterations shows that the minimum time that can be reached using random ordering over 30 iteration is 19.362 days which is an improvement of 4.54 days from the current scheduling model.

For the other rules it is only necessary to run these rules once due to their deterministic nature.

5.1.2 Scheduling Qualitech to obtain the minimum flushing times

This heuristic has been run with the rule of scheduling the Qualitech range to incur the smallest amount of flushes. Table 11 shows the results obtained.

Qualitech production order	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
Total time to produce Qualitech range [min]	7722.5
Number of extra days needed	1.34
Total production days required	20.07

Table 11: **New model using the minimum flushing times rule.**

The results from using the minimum number of flushing times rule shows that the minimum time in which all the products can be produced is 20.07 days, which is an improvement of 3.84 days from the current scheduling model. Although it is better than the current schedule it is a little longer than the minimum obtained by random scheduling.

5.1.3 Scheduling Qualitech to obtain the maximum flushing times

Qualitech production order	15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0
Total time to produce Qualitech range [min]	7785
Number of extra days needed	1.02
Total production days required	19.75

Table 12: **New model using the maximum flushing times rule.**

The results above have been obtained with the rule of scheduling the Qualitech range to incur the maximum amount of flushes.

The results from using the maximum amount of flushing times shows that the minimum time in which all the products can be produced is 19.75 days, which is an improvement of 4.16 days from the current scheduling model. Although it is better than the current schedule it is a little longer than the minimum obtained by random scheduling. This scheduling rule is also better than the smallest flushing time rule.

5.1.4 Scheduling the smallest demands for Qualitech first

This heuristic has been run with the rule of scheduling the Qualitech range with the smallest demands first. Table 13 shows the results obtained.

Qualitech production order	15,14,9,8,7,6,10,5,13,1,12,0,11,4,2,3
Total time to produce Qualitech range [min]	7783
Number of extra days needed	0.97
Total production days required	19.7

Table 13: **New model scheduling the smallest demands first.**

The results from scheduling the smallest demands first shows that the minimum time in which all the products can be produced is 19.7 days, which is an improvement of 4.21 days from the current scheduling model. Although it is better than the current schedule it is a little longer than the minimum obtained by random scheduling. It is also better than both the smallest and largest flushing times rules.

5.1.5 Scheduling the largest demands for Qualitech first

This heuristic is based on the rule of scheduling the Qualitech range with the largest demands first. Table 14 shows the results obtained.

Qualitech production order	3,2,4,11,0,12,1,13,5,10,6,7,8,9,14,15
Total time to produce Qualitech range [min]	7759.5
Number of extra days needed	1.31
Total production days required	20.03

Table 14: **New model scheduling the largest demands first.**

The results from scheduling the largest demands first shows that the minimum time in which all the products can be produced is 20.03 days which is an improvement of 3.88 days from the current scheduling model. Although it is better than the current schedule it is the worst scheduling rule out of the five.

To summarize the results Table 15 indicates the different rules used and their corresponding results. These results include the number of days that each rule saves over the current model and the total number of days needed to produce the demand of all the products.

Results			
Heuristic used to order Qualitech	Total time needed for Qualitech range [min]	Extra production days needed	Total production days needed
Random			
Average	7795.00	0.99074	19.72004
Median	7795.00	0.95788	19.68718
Min	7778.50	0.63256	19.36185
Max	7816.50	1.36766	20.09696
Smallest flushing times	7722.50	1.34000	20.07000
Longest flushing times	7785.00	1.02000	19.75000
Smallest demands first	7783.00	0.97000	19.70000
Largest demands first	7759.50	1.31000	20.03000

Table 15: New model summary of results.

Number of days saved from current scheduling model			
	New	Current	Days saved
Random			
Average	19.72004	23.90555657	4.18552
Median	19.68718	23.90555657	4.21838
Min	19.36185	23.90555657	4.54371
Max	20.09696	23.90555657	3.80860
Smallest flushing times	20.07000	23.90555657	3.83556
Longest flushing times	19.75000	23.90555657	4.15556
Smallest demands first	19.70000	23.90555657	4.20556
Largest demands first	20.03000	23.90555657	3.87556

Table 16: New model summary of results with days saved.

In the next section a number of random demands have been tested using all of the rules above in order to exploit the best heuristic rule. The findings will be discussed in the section to follow.

5.2 Finding the best heuristic rule based on total number of production days

The five heuristic models were tested with 30 random demand scenarios, with each heuristic using the same 30 demand scenarios. Table 17 shows the results of these runs with their respective averages, medians, minimums and maximums. These random demands were created using the range of the actual demand, maximum minus minimum, and assigning a random value in that range to each product.

From Table 17 it is clear that the largest flushing times heuristic is the best in terms of minimum total duration. This may sound counter intuitive at first but the reason why this is true is because the flushing times for Qualitech ranges from ten to twenty minutes. This is not a very large range of flushing times, therefore, for this data the flushing times did not play such a major role in which heuristic to use. It is recommended to run these heuristic rules every time the initial data changes, this include flushing times, demands and production rates changes.

Total production days needed					
	Random rule	Smallest flushing times first	Largest flushing times first	Smallest demand first	Largest demand first
	25.44	25.06	25.18	25.13	25.44
	29.5	29.5	29.5	29.5	29.5
	22.81	22.81	22.81	22.81	22.81
	23.41	23.59	23.48	23.55	23.41
	25	24.97	24.46	24.44	25
	25.11	24.7	24.63	24.63	25.11
	25.32	25.2	25.24	25.2	25.32
	21	20.89	20.83	21.17	21
	23.71	23.76	23.47	23.7	23.71
	21.57	21.53	20.53	21	21.57
	23.25	23.46	23.28	23.46	23.25
	22.43	22.39	22.39	22.39	22.43
	20.95	21.16	21.05	21.26	20.95
	26.76	26.75	26.65	26.65	26.76
	25.21	25.21	25.21	25.21	25.21
	25.49	25.36	25.36	25.36	25.49
	26.48	26.48	26.48	26.48	26.48
	25.48	25.47	25.13	25.1	25.48
	23.24	23.25	22.81	22.8	23.24
	22.58	22.52	22.53	22.45	22.58
	23.69	23.69	23.69	23.69	23.69
	17.26	17.39	17.08	17.19	17.26
	23.64	23.4	23.6	23.84	23.64
	23.59	23.53	23.09	23.4	23.59
	22.11	21.77	21.65	21.77	22.11
	26.56	26.51	26.59	26.51	26.56
	25.81	25.81	25.81	25.88	25.81
	27.08	27.08	27.08	27.08	27.08
	24.01	23.98	23.85	23.85	24.01
	25.92	26.15	26.07	26.4	25.92
Average	24.1470	24.1123	23.9843	24.0633	24.1470
Median	23.8600	23.8700	23.7700	23.8450	23.8600
Min	17.2600	17.3900	17.0800	17.1900	17.2600
Max	29.5000	29.5000	29.5000	29.5000	29.5000

Table 17: New model summary of results.

Alternative criteria that have to be considered with the implementation of the best rule include the environmental impact, impact on production, social impact on employees and financial impact. Implementing the new heuristic based scheduling will have a low positive impact on the immediate environment as the plant will be operational for shorter periods of the month, leading to less pollution and a decrease in natural resources used. The impact

on the production will not be affected as the new heuristic scheduling algorithm can be implemented without any delays in production. The employees of the company are going to be affected with the implementation of the new heuristic as overtime will be eliminated and shorter periods of work will be needed. The management and employees of the company will have to communicate clearly how they are going to address this problem as employees cant afford a decrease in their salary. The company must look for alternative departments to utilize these workers.

The financial impact of implementing the heuristic will be beneficial to the company. The fixed costs of raw materials and labor will remain constant but the variable expenses such as electricity, water and overtime will decrease. The plant can be saving up to R 653.00 on electricity per day, R 430.00 on water per day and R 18 000 per day on overtime. That leads to a total savings of R 19 083.00 per day. The new heuristic will eliminate four days of production, that is equal to a total saving of R 76 332.00 for 4 days saved.

In summary, the "largest flushing times first" rule will be analyzed on its robustness to input data. Input data such as demands, production rates and flushing times may influence the results. Therefore a sensitivity analysis will be conducted on the largest flushing times heuristic in the next section. The Gantt chart below shows the schedule of the best solution. The blue lines represents the production of Easymix and the red lines represents the production of Qualitech.

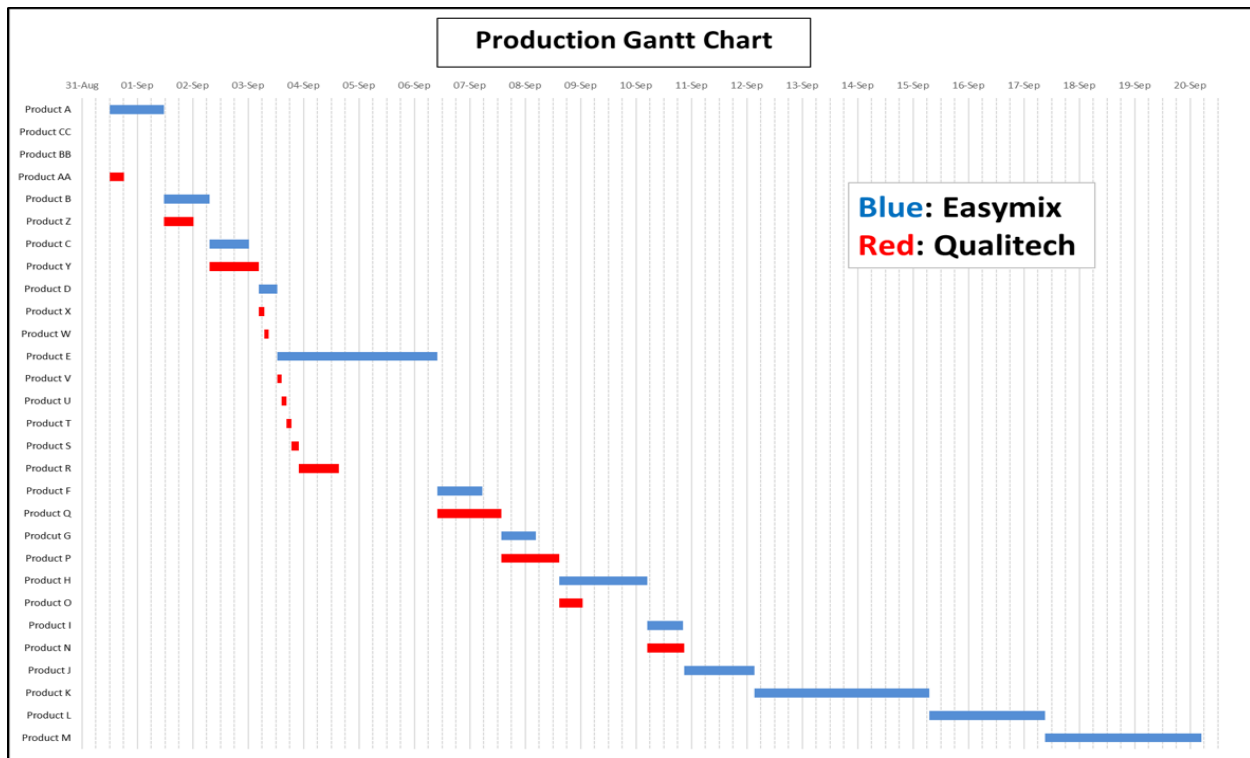


Figure 5: Actual schedule for largest flushing times.

Chapter 6

6 Verification and validation

6.1 Verification

In this subsection the current model that the company is using and the newly developed heuristic will be ran in parallel to ensure that they can produce the same answer. For validation of the heuristic the input data for Qualitech demands have to be set equal to zero. The reason for this is that the current model schedules Qualitech after the Easymix range, whereas the new heuristic schedules the Qualitech in between the Easymix range. If the current model and the new heuristic produce the same answer the heuristic is valid.

The current model and the new heuristic model have been tested using the normal demands and input data as received. The results are shown in the table below.

	A	B	C	D	E	F
1	Production rate	T/H		Days Required Easymix		
2	500G	1.17		18.73 Days	Calculation	=F21/24
3	1kg	1.74		Hours required		
4	Qualitech	Units/min		1123.76 Hours	Calculation	=D4*60
5	1KG	38				
6	Start production for the month with Easymix 500g range					
7	SKU		Tons	Hours	Flushing	Total Time
8	Product A	500 G	26.3	=C8/B\$2	1	=D8+E8
9	Product B	500 G	22.53	=C9/B\$2	0.5	=D9+E9
10	Product C	500 G	19.54	=C10/B\$2	0.5	=D10+E10
11	Product D	500 G	8.56	=C11/B\$2	0.5	=D11+E11
12	Product E	500 G	79.97	=C12/B\$2	1	=D12+E12
13	Product F	1 KG	32.22	=C13/B\$3	1	=D13+E13
14	Product G	1 KG	22.35	=C14/B\$3	2	=D14+E14
15	Product H	500 G	43.5	=C15/B\$2	1	=D15+E15
16	Product I	500 G	9.92	=C16/B\$2	7	=D16+E16
17	Product J	500 G	29.63	=C17/B\$2	5	=D17+E17
18	Product K	500 G	87.36	=C18/B\$2	1	=D18+E18
19	Product L	500 G	53.88	=C19/B\$2	4	=D19+E19
20	Product M	500 G	73.52	=C20/B\$2	5	=D20+E20
21	Totals [hours]			=SUM(D8:D20)	=SUM(E8:E20)	=SUM(F8:F20)

Table 18: Current model formulas for computing total time needed per month.

The following data in the figure below is directly imported from the heuristic model using python. The total number of days needed to produce Easymix are the same, this concludes that the heuristic model is valid.

```

>>> runfile('C:/Users/use/Desktop/OneDrive/Semester 2 2016/01 BPJ 420/97 Pyth
on projek/01 Main files/001 Main file 03_08_2016.py', wdir=r'C:/Users/use/Des
ktop/OneDrive/Semester 2 2016/01 BPJ 420/97 Python projek/01 Main files')
Product A need 22.48 hours in order to fulfill the demands
Product B need 19.26 hours in order to fulfill the demands
Product C need 16.7 hours in order to fulfill the demands
Product D need 7.32 hours in order to fulfill the demands
Product E need 68.35 hours in order to fulfill the demands
Product F need 18.52 hours in order to fulfill the demands
Product G need 12.84 hours in order to fulfill the demands
Product H need 37.18 hours in order to fulfill the demands
Product I need 8.48 hours in order to fulfill the demands
Product J need 25.32 hours in order to fulfill the demands
Product K need 74.67 hours in order to fulfill the demands
Product L need 46.05 hours in order to fulfill the demands
Product M need 62.84 hours in order to fulfill the demands
The total amount of days needed for Easymix = 18.73

```

Figure 6: Heuristic output for producing Easymix range.

6.2 Sensitivity analysis

The sensitivity analysis will be used to evaluate the robustness of the model against fluctuations in input data. Input data that will be tested in the sensitivity analysis include:

1. Demands,
2. Production rates,
3. Mixer times and
4. Qualitech flushing times.

The heuristic was repeatedly executed using different data sets. These data sets were obtained by multiplying the original data with fractions to replicate decreases and increases in the monthly input data. These twenty fractional values include:

[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2]

The fractions 0.1 to 0.9 will represent the scenario where the demands, production rates, flushing times and mixing times are only [10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%] respectively of the normal amount. The fraction equal to 1 will represent the normal running conditions and the fractions 1.1 to 2 will mean that all the demands, production rates, flushing times and mixing times are increased to [110%, 120%, 130%, 140%, 150%, 160%, 170%, 180%, 190%, 200%] respectively.

The total time of the schedule will be calculated as before by using the largest flushing times heuristic. The fraction equal to one represents normal conditions and will be used as a reference point. Negative percentage differences indicates that the total time of the fraction under consideration is worse than the normal running conditions. The percentage total time difference is calculated as follows:

$$\%Difference = \frac{Totaltime_{100\%} - Totaltime_{X\%}}{Totaltime_{100\%}} \quad (11)$$

6.2.1 Sensitivity analysis on the product demands

The sensitivity of the heuristic against fluctuations in product demands can be vital as demands will vary every month. The heuristic must be able to adapt to fluctuations in demand with ease in order for it to be implemented in the real world environment.

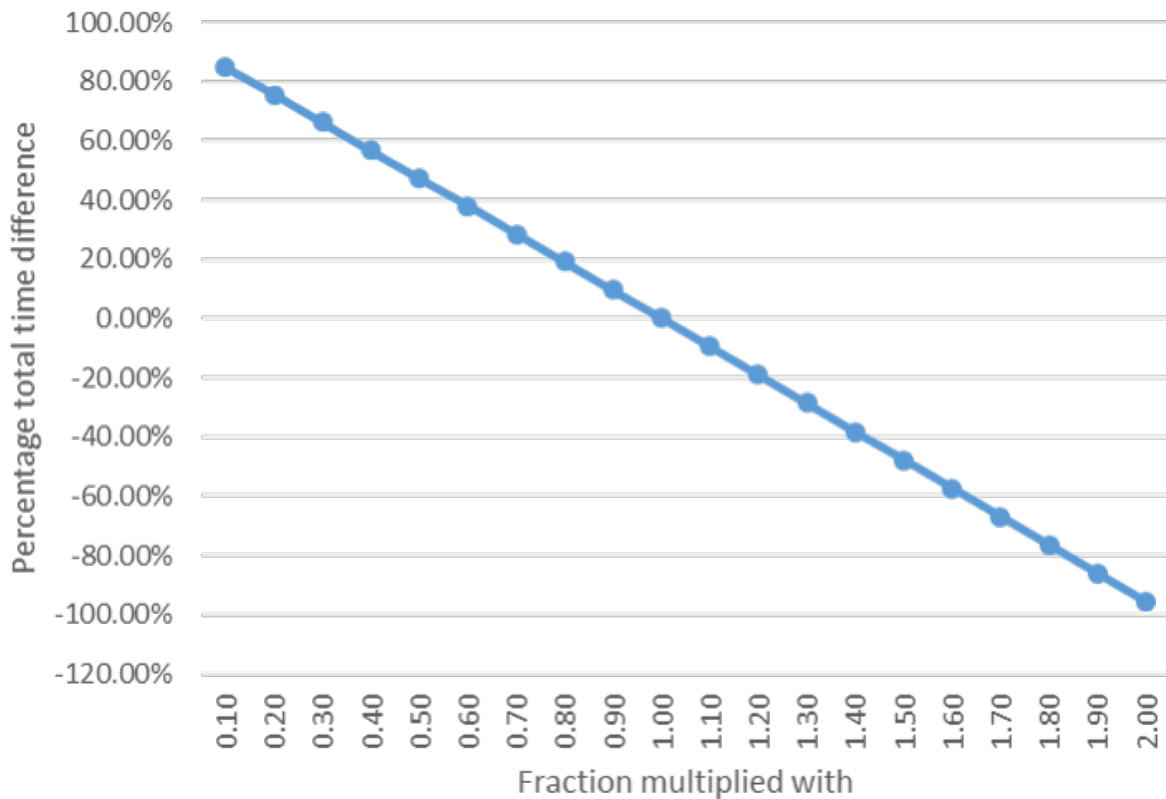


Figure 7: Sensitivity analysis on product demands.

Figure 7 clearly indicates that the heuristic is not robust to fluctuations in demand. If the demand decrease to only 10% the difference in total time decreases by almost 85%. The heuristics total time is directly related to the fluctuation in demands. If the demands rise so will the total time needed. The inverse is also true. The relationship between demands and number of production days required is linear with a medium to high gradient as seen from Figure 7.

The next analysis performed on the heuristic scheduling the largest flushing times consists of fluctuations in production rates. Changes in the machines production rates are rare. Implementations to improve the production rates are expensive and are normally planned over a large period of time, but if the company is looking at improving their productivity of the machines the relevant impact can be observed using this analysis.

6.2.2 Sensitivity analysis on production rates.

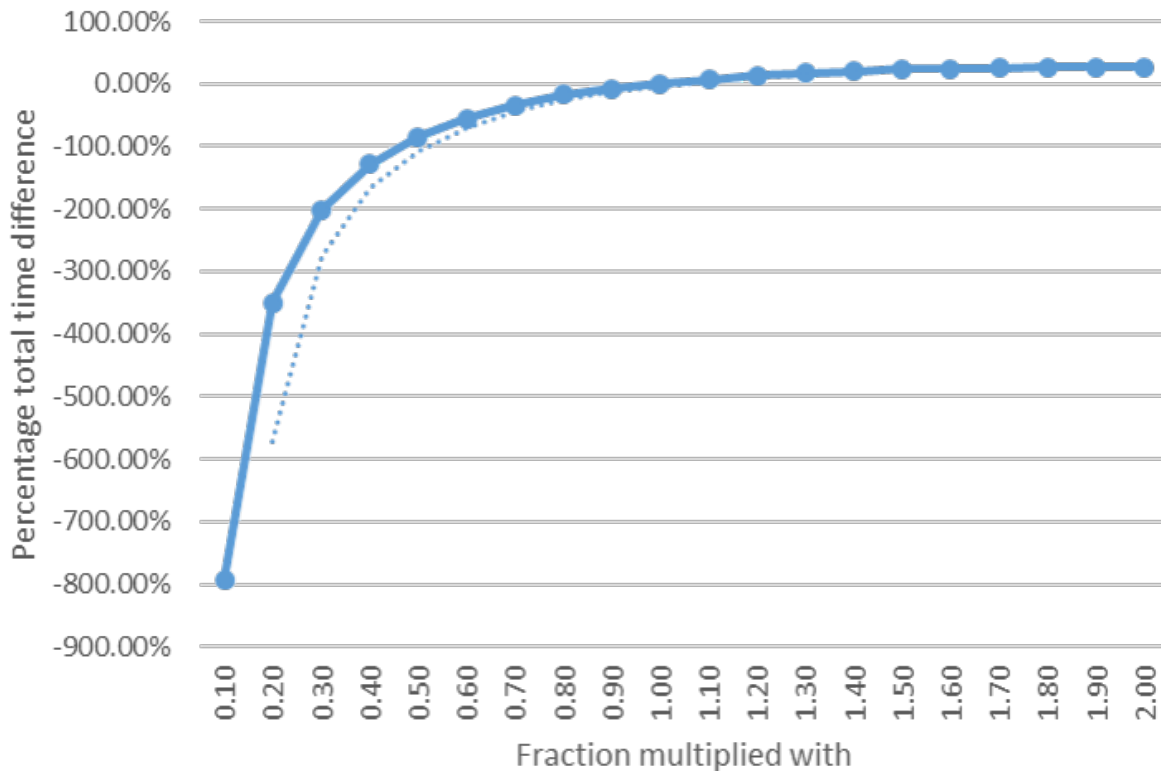


Figure 8: **Sensitivity analysis on production rates.**

The largest flushing time heuristic is sensitive to fluctuations in production times. As seen from Figure 8 a decline in production rates have a critical impact on the number of production days required, if the production rate decreases by 50% the total number of days needed increases by 85%. The heuristic is less sensitive to production rate increases, since the production rate increases with 50% the total number of days only decrease by 23%. Production rates have an exponential relationship with the number of production days needed.

6.2.3 Sensitivity analysis on mixer mixing times.

In Figure 9 the fraction of 10% indicates that the new mixing time is only 10% of the original time. The largest flushing time heuristic is robust against changes in mixing times. This relationship is important as the company is considering installing another mixer in parallel to the current one. This scenario of installing another mixer is related to the fraction

represented by 50%, meaning the mixer only uses half the time. The installation of a secondary mixer would not have a large impact as the total days needed with two mixers is only 3.39% better compared to one mixer. The relationship between mixing time and the number of production days needed is almost linear with a few exceptions at very low efficiency and very high efficiency.

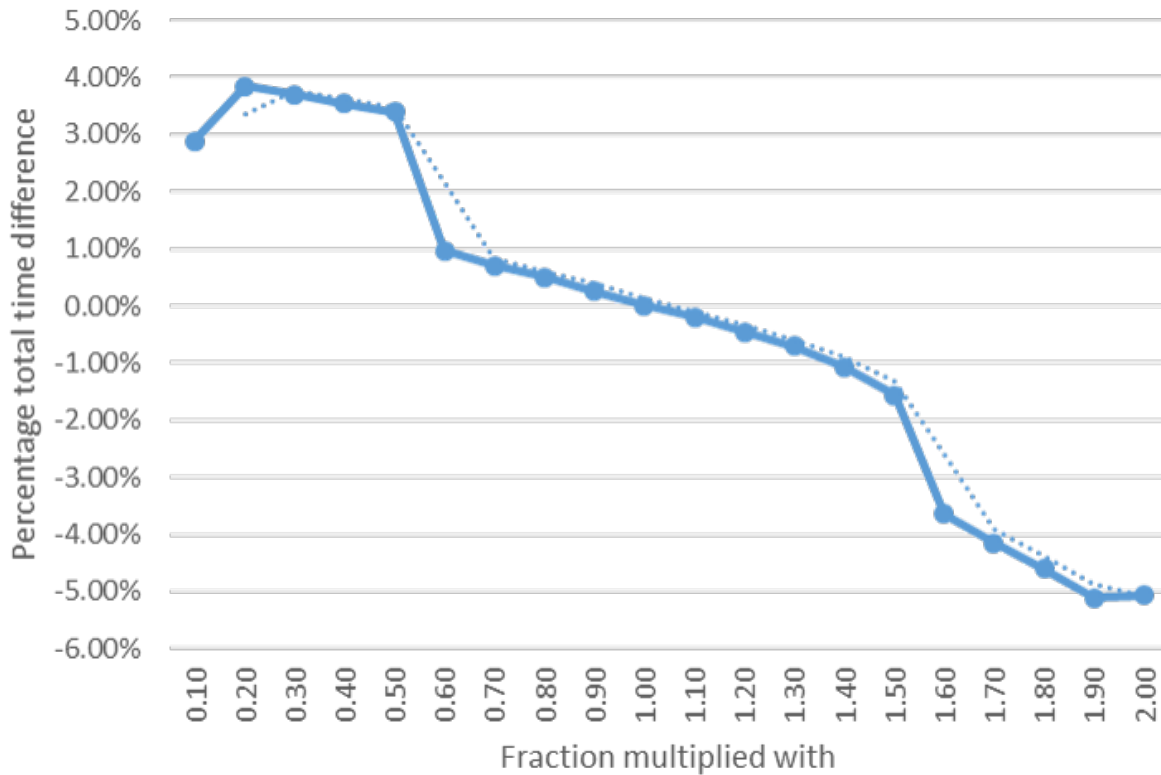


Figure 9: Sensitivity analysis on mixer mixing times.

6.2.4 Sensitivity analysis on Qualitech flushing times.

This section is important because it is not usually suggested to schedule orders in such a way that the largest flushing times occur. This heuristic rule performs the best because the range of flushing times between products is very small. When inspecting the Qualitech flushing times it can be seen that the longest flushing time is 29 min and the shortest is 10 min, resulting in a range of only 19 min. In the scheduling for a month with an average of 43200 minutes, a range of 19 minutes does not have a big impact.

Figure 10 shows that the heuristic is robust to fluctuations in Qualitech flushing times. This result is positive as flushing times may vary from shift to shift. The largest percentage difference in total time is very small equal 0.2%. Fluctuations in Qualitech flushing times is not of real concern when using this heuristic. The relationship between Qualitech flushing times and number of production days needed is linear with a very low gradient.

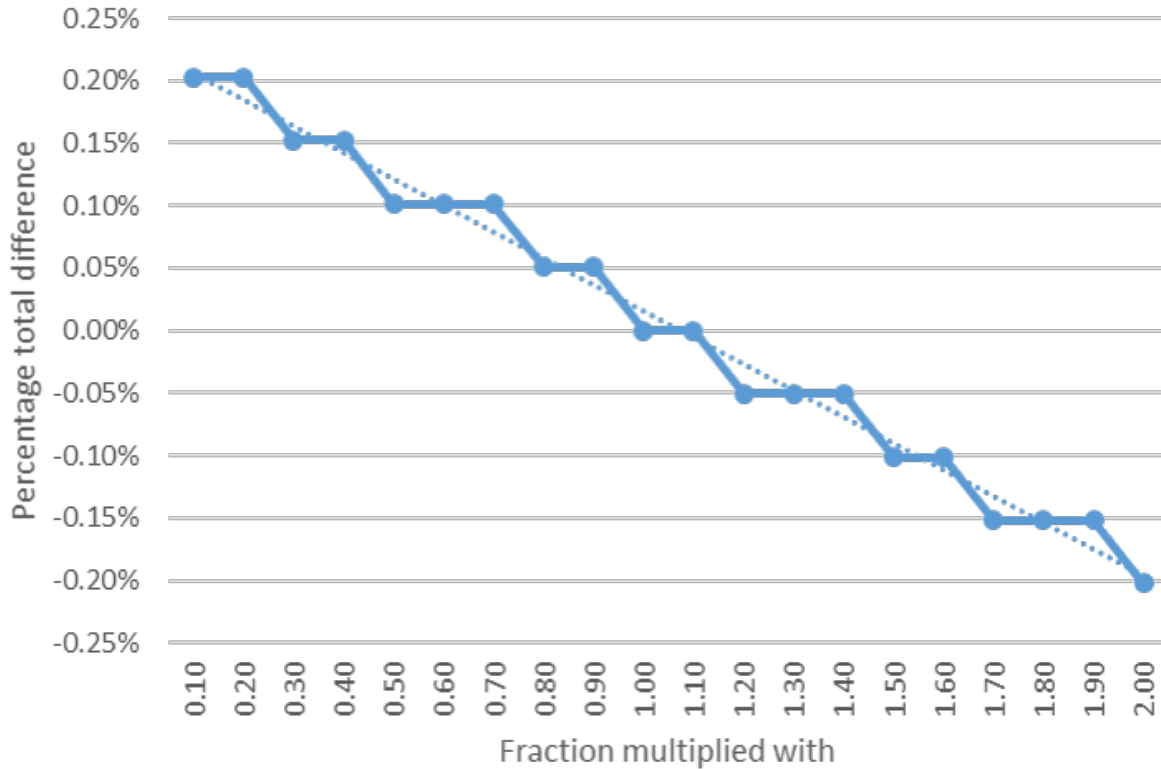


Figure 10: Sensitivity analysis on Qualitech flushing times.

The next analysis section was only used to identify the possibility of finding a better solution over a greater period of time spent searching. In this analysis the random heuristic was used to identify better solutions.

6.2.5 Further analysis on the baseline heuristic

In order to run through all the possible solutions, which will be 16 ! (sixteen faculty) solutions will take approximately 160727 Days. For that reason it was decided to rather run the random heuristic rule 1000 times to see if a lower value can be obtained. The figure shown below shows the results obtained from running the random rule over 1 000 iterations.

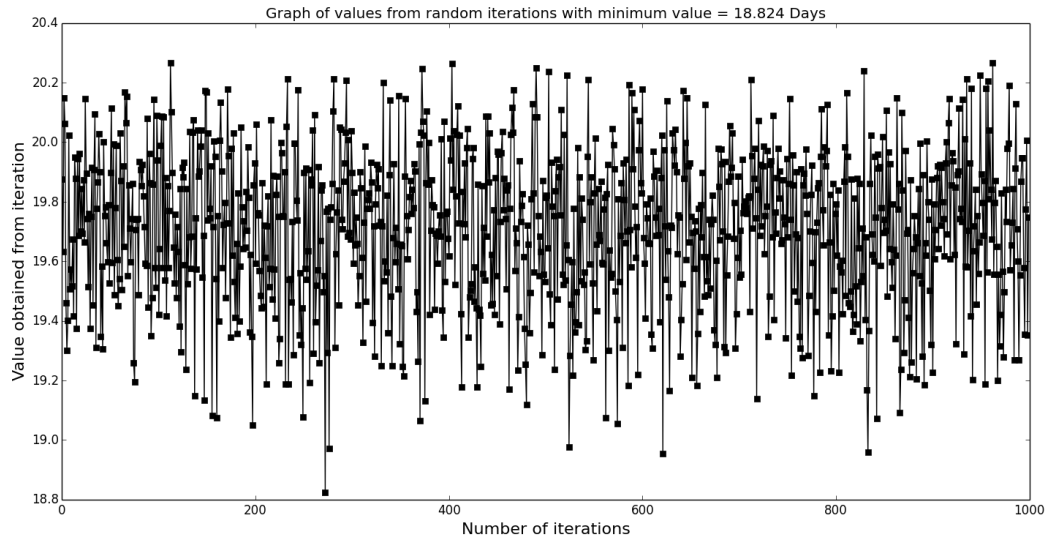


Figure 11: Results from running the random rule over 1000 iterations.

The minimum obtained from the 1000 iterations is almost one day shorter than the largest flushing time heuristic. In summary the largest flushing time first heuristic will provide the best solution for the program run time. If time is not of concern the random heuristic can be executed as many times as possible to obtain an even better solution.

6.3 Project achievements

The project has successfully satisfied the goal of delivering not only a heuristic algorithm that will schedule all the demands for one month but also four other heuristics which can be used later as the input data changes.

The heuristic chosen as the best can easily be implemented by the company. No additional resources have to be bought. The production manager will just have to follow the production order given by the heuristic. The quality of the solution is high. The solution have been tested against real world data and delivered excellent results. The current run time for the best heuristic rule is 0.0005 seconds. No time is wasted on waiting for a schedule. The project was a success as the newly developed heuristic model can produce the same number of demand by interspersing one product run with other product runs. Producing the demands in 19.75 days whereas the current model will schedule it over 24.162 days, giving the company 4.41 days of savings. The 4.41 days of savings will result in savings on manufacturing overhead such as electricity, and costs related to manufacturing, it will also eliminate the need for overtime workers which is a significant expense.

The heuristic proposed to the company will be able to operate in a live environment. It has been tested against the current schedules and various sensitivity analysis have been conducted. The heuristic can be implemented with ease and will be able to handle real world conditions without any problems.

7 Conclusion

7.1 Summary and impact of project

Scheduling is a key factor for delivering a quality and reliable product at the right time. Scheduling tools enable companies around the world to minimize non-value added factors like setup times, setup cost and changeovers. This report considered a parallel machine scheduling problem, with sequence dependent setup times for the production of flour products. The primary objective of the schedule is to minimize the total production time.

Five heuristic algorithms were developed. Results were obtained by running all of the heuristic rules over thirty random demand scenarios. The best heuristic rule was obtained by scheduling the largest flushing times first, the best rule was then tested with respect to its robustness to change and sensitivity to fluctuations in input data. The project is scoped from the beginning of the product production processes, raw materials and mixing, to the point where all the value adding processes are completed and a quality output product is achieved. The heuristic chosen as the best can easily be implemented by the company. No additional resources have to be bought. The production manager will just have to follow the production order given by the heuristic. The quality of the solution is high. The solution have been tested against real world data and delivered excellent results. The current run time for the best heuristic rule is 0.0005 seconds. No time is wasted on waiting for a schedule.

The financial impact of implementing the heuristic will save the company up to R653.00 on electricity per day, R430.00 on water per day and R18 000.00 per day on overtime. That leads to a total savings of R19 083.00 per day. The new heuristic will eliminate four days of production per month, which is equal to a total saving of R76 332.00 over four days. The employees of the company are going to be affected with the implementation of the new heuristic as overtime will be eliminated and shorter periods of work will be needed. The management and employees of the company will have to communicate clearly how they are going to address this problem as employees cannot afford a decrease in their salary. The company must look for alternative departments to utilize these workers.

7.2 Future work

Researchers or the company can still improve on the current heuristic. The methods of improving the current heuristic will include the use of a genetic algorithm (GA). Using a GA will help exploit the most optimal solution in a specified time frame. A GA will be applicable to use as the sample size is very large and the user does not have an idea of what the optimal solution looks like.

The model can also be optimized using a different approach from the start. The model can be developed to schedule the Easymix and Qualitech range in parallel. Currently the Easymix is scheduled first with the current method and then Qualitech is scheduled in the mixer idle time. This method can be improved by scheduling the Easymix as well as the Qualitech using different heuristic rules. This approach can also be complimented with the use of a GA to find the optimal solution.

References

- Barringer, H Paul (1997). “Availability, reliability, maintainability, and capability”. In: Chang, Pei-Chann, Jih-Chang Hsieh, and Yen-Wen Wang (2003). “Genetic algorithms applied in BOPP film scheduling problems: minimizing total absolute deviation and setup times”. In: *Applied Soft Computing* 3.2, pp. 139–148.
- Cuthbert, Thomas R (1987). *Optimization using personal computers: with applications to electrical networks*. John Wiley & Sons, Inc.
- Feo, Thomas A and Mauricio GC Resende (1995). “Greedy randomized adaptive search procedures”. In: *Journal of global optimization* 6.2, pp. 109–133.
- Goldberg, David E and John H Holland (1988). “Genetic algorithms and machine learning”. In: *Machine learning* 3.2, pp. 95–99.
- Graham, R.L. et al. (1979). “Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey”. In: *Discrete Optimization II, Proceedings of the Advanced Research Institute on Discrete Optimization and Systems Applications of the Systems Science Panel of NATO and of the Discrete Optimization Symposium co-sponsored by IBM Canada and SIAM Banff, Aha. and Vancouver*, pp. 287–326.
- Grobler, Jacomine (2008). “Particle swarm optimization and differential evolution for multi objective multiple machine scheduling”. PhD thesis. University of Pretoria.
- Gupta, Jatinder ND and Samarn Chantaravarapan (2008). “Single machine group scheduling with family setups to minimize total tardiness”. In: *International Journal of Production Research* 46.6, pp. 1707–1722.
- Haupt, Randy L and Sue Ellen Haupt (2004). *Practical genetic algorithms*. John Wiley & Sons.
- Kır, Sena and Harun Re ,sit Yazgan (2016). “A sequence dependent single machine scheduling problem with fuzzy axiomatic design for the penalty costs”. In: *Computers & Industrial Engineering* 92, pp. 95–104.
- Kolisch, Rainer and S`onke Hartmann (1999). *Classification and computational analysis*. Springer.
- Laguna, Manuel and Jos´e Luis Gonz´alez Velarde (1991). “A search heuristic for just-in-time scheduling in parallel machines”. In: *Journal of Intelligent manufacturing* 2.4, pp. 253–260.
- Luke, Sean (2013). *Essentials of Metaheuristics*. second. Lulu.
- Moghaddas, R and M Houshm (2008). “Job-shop scheduling problem with sequence dependent setup times”. In: *Premier foods*. <http://www.premierfoods.co.za>. Accessed: 2016-03-18.
- Sethanan, Kanchana. “Scheduling flexible flowshops with sequence dependent setup times”. PhD thesis. West Virginia University.
- T`kindt Vincent Billaut, Jean-Charles (2006). *Multicriteria scheduling*. Multicriteria scheduling. Springer.
- Zandieh, M., S.M.T. Fatemi Ghomi, and S.M. Moattar Husseini (2006). “An immune algorithm approach to hybrid flow shops scheduling with sequence-dependent setup times”. In: *Applied Mathematics and Computation* 180.1, pp. 111–127.

Appendix A: Project sponsorship form

Department of Industrial & Systems Engineering Final Year Projects

Identification and Responsibility of Project Sponsors


Final Year Projects may be published by the University of Pretoria on *UPSpace* and may thus be freely available on the Internet. These publications portray the quality of education at the University, but they have the potential of exposing sensitive company information. It is important that both students and company representatives or sponsors are aware of such implications.

Key responsibilities of Project Sponsors:

A project sponsor is the key contact person within the company. This person should thus be able to provide guidance to the student throughout the project. The sponsor is also very likely to gain from the success of the project. The project sponsor has the following important responsibilities:

1. Confirm his/her role as project sponsor, duly authorised by the company. Multiple sponsors can be appointed, but this is not advised. The duly completed form will be considered as acceptance of sponsor role.
2. Review and approve the Project Proposal, ensuring that it clearly defines the problem to be investigated by the student and that the project aim, scope, deliverables and approach is acceptable from the company's perspective.
3. Review the Final Project Report (delivered during the second semester), ensuring that information is accurate and that the solution addresses the problems and/or design requirements of the defined project.
4. Acknowledges the intended publication of the Project Report on UP Space.
5. Ensures that any sensitive, confidential information or intellectual property of the company is not disclosed in the Final Project Report.

Project Sponsor Details:

Company:	Premier foods
Project Description:	Multiple machine scheduling problem with independent product setup and flushing times
Student Name:	Marco Croucamp
Student number:	13035101
Student Signature:	<i>Mroucamp</i>
Sponsor Name:	<i>ERWIN POTGIETER</i>
Designation:	<i>Mill MANAGER</i>
E-mail:	<i>erwin.potgieter@premierfmcg.com</i>
Tel No:	<i>012 8490500</i>
Cell No:	<i>0825656738</i>
Fax No:	<i>0866698537</i>
Sponsor Signature:	

Appendix B: Reflection on learning

Independent learning process

Literature study The literature study has taught me new and effective ways of searching for appropriate articles, journals and books. The lecturer provided general guidelines for literature searches, but I identified the problem classification on my own and conducted the literature review independently. After I had learned the skill to search more effectively through the literature it was necessary for me to find articles which was applicable to the problem. I then realized that the classification of the problem based on problem classification literature was vital. Not only did this step help with the search for relevant articles but also improved my understanding of the problem. At this time I knew how to search for appropriate literature and I knew what my problem was. The problem was that the company is not using any scheduling tool in scheduling the production for the month. The next hurdle I had to cross was finding reliable articles, journals and books. I turned to the UP's library services and through them I searched Google scholar for the specific problem.

Problem statement After the problem had been identified a few interviews with the project sponsor and the production manager were necessary to obtain the information regarding the problem. The challenge was finding literature which would help me classify the type of problem I had, after finding the literature needed to classify my problem it was possible to find similar problems in articles. Using my problem and relevant literature I was able to investigate the different approaches and methods used to solve these types of problems. Articles related to my problem helped me define my scope and taught me the process of defining a problem statement given the objectives and constraints linked to the problem. I have learned that by setting the scope early in the project helps eliminating the irrelevant information that sometimes comes along with similar problems.

Responsibilities outside formal instructions

Learning professional typesetting methods was necessary to create a professional report. A program which assisted me with the professionalism of the report included Latex. The program had to be learned in order to use it efficiently. Report layout is one of the easiest ways to improve the professionalism of the report, thus the layout of scientific reports were examined in order to learn which layout is appropriate for my project. Extensive coding is used in this project with the development of a heuristic model. Different programming languages will be examined during the project. External resources such as coding platforms like Matlab requires basic understanding of the program itself.

Methods of solving similar problems include the use of algorithms, heuristics and heuristics combined with metaheuristics. These methods haven't been covered in depth in educational courses, these methods had to be examined for better understanding and to be able to use them. Different types of these methods were also considered and studied on a high level. Latex and heuristics are not addressed in the undergraduate IE syllabus. The understanding of how heuristics work and how to use heuristics were self study done by me. Latex is not a requirement for this project so the learning of Latex was self study. Another big decision

in the project was made without guided assistance. I selected to use the Genetic algorithm as the way of finding the optimal solution for the problem.

Certain skills required by the workplace have been learned over the past few months. These skills include the means of communication between project managers, mentors, workers and myself. Communication plays a vital role in any project. I learned the importance of keeping the communication on a professional level. The company taught me the importance of accuracy when considering the data.

Presenting the project to the company and university is important for the success of this project, different tools and techniques to be used for presenting was learned for example, the KISS principle, keep it simple and short.

Social and ethical impact of the project

The social impact that this project may have on the workers within the company is that at this stage the company is running a lot of production over weekends which counts as overtime. The reason why the workers prefer this is because overtime is paid at a ratio of 1.5 of the week day rate. If this project is able to develop a schedule which can assure the managers that production is only necessary on weekdays it will have an impact on the workers salary for the month. This may cause workers to be against the implementation of this project. On the positive side, this project is only focusing on scheduling, which assures the workers that no one will be retrenched or shifted to another department. It will also increase the work-life balance of the workers.

Ethically the only impact that the project could have is the quality of the product may decrease with an increase in production rate. In order to keep the quality I will ensure that the actual production time remains the same. The reason why this could be an ethical situation is because the company is producing food related products. Other than that there are no ethical implications.

Future learning

The methods of implementing such a project must be examined. Where to begin with the implementation, how much time is necessary for the implementation and how to encourage the workers that they will also benefit from this project, will be considered.

Appendix C: Actual python code for base heuristic used

```

import time
start = time.clock()
"""
Marco Croucamp
13035101
marcocroucamp@gmail.com

The following section is used to generate the scheduling for production of the
Easymix products (500g and 1kg). Later in this code will be the scheduling of
the Qualitechs products
"""
import numpy as np
#Put demand in production sequence
Easymix_500g_demands = [26.30,22.53,19.54,8.56,79.97,0,0,43.5,9.92,
                        29.63,87.36,53.88,73.52]
Easymix_1kg_demands = [38.21,0,0,0,93.61,32.22,22.35,
                        49.05,0,0,91.71,0,43.93]
Flushing_times = [60,30,30,30,60,60,120,60,420,300,60,240,300]

#Amount of minutes it takes to mix 1 ton of raw material.
Mixing_1_ton = 9.5

#Production rate per machine. Ton/hour
Rate_500g = 1.17
Rate_1kg = 1.74

Mixing_times = [] #Total mixing time required per product of easymix.
Production_time_per_product = [] #Total production time per product of easymix.
Mixer_idle_time_per_products= [] #Mixer idle time during production of Easymix.
Atoz = ["A","B","C","D","E","F","G","H","I","J","K","L","M"]
#Running a for loop through all the Easymix products to fill in lists above.
for i in range(0,len(Easymix_500g_demands)):
    #Calculating the amount of mixing time needed for the demand given.
    Mixing_time_500g = Easymix_500g_demands[i] * Mixing_1_ton
    Mixing_time_1kg = Easymix_1kg_demands[i] * Mixing_1_ton

    #Appending the amount of time required for mixing into a list.
    Mixing_times.append(Mixing_time_500g + Mixing_time_1kg)

    #Determining whether the 500g or 1kg production of the same product
    #will take longer,and then using the longest time for further calculations
    if Easymix_500g_demands[i]/Rate_500g > Easymix_1kg_demands[i]/Rate_1kg:
        Production_time = Easymix_500g_demands[i]/Rate_500g*60
    else:
        Production_time = Easymix_1kg_demands[i]/Rate_1kg*60

    #Calculating the total production time per product demand plus flushing
    Production_time_per_product.append(Production_time + Flushing_times[i])

    #Calculation of the mixer idle time per product.
    Mixer_idle_time = Production_time_per_product[i] - Mixing_times[i]
    Mixer_idle_time_per_products.append(Mixer_idle_time)
    print "Product " + str(Atoz[i]) + " need " +
          str(round(Production_time/60,2)) + " hours in order to fulfill the demands"
"""
The next section of code is used to schedule the Qualitech products in
between the Easymix products, in the idle spaces of the mixer.
"""
Units_per_min = 38 #Rate at which the qualitech machine produces.
Percentage_mixer_available = 0.85 # % of mixer idle time actually useable.

#Calculate the actual amount of minutes the mixer is available for poduction
Mixer_available_for_qualitech = [i*Percentage_mixer_available for i in

```

```

Mixer_idle_time_per_products ]

#Qualitech demands (In units)
Qualitech_demands = [30000,22200,44600,46400,38800,6600,4600,4000,3800,3400,
                    4600,36600,29000,13400,0,0]

#Qualitech flushing times in minutes
Qualitech_flushing_array =
    np.array([[0,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10],
             [15,0,10,10,10,10,10,10,10,10,10,10,10,10,10,10],
             [16,15,0,10,10,10,10,10,10,10,10,10,10,10,10,10],
             [17,16,15,0,10,10,10,10,10,10,10,10,10,10,10,10],
             [18,17,16,15,0,10,10,10,10,10,10,10,10,10,10,10],
             [19,18,17,16,15,0,10,10,10,10,10,10,10,10,10,10],
             [20,19,18,17,16,15,0,10,10,10,10,10,10,10,10,10],
             [21,20,19,18,17,16,15,0,10,10,10,10,10,10,10,10],
             [22,21,20,19,18,17,16,15,0,10,10,10,10,10,10,10],
             [23,22,21,20,19,18,17,16,15,0,10,10,10,10,10,10],
             [24,23,22,21,20,19,18,17,16,15,0,10,10,10,10,10],
             [25,24,23,22,21,20,19,18,17,16,15,0,10,10,10,10],
             [26,25,24,23,22,21,20,19,18,17,16,15,0,10,10,10],
             [27,26,25,24,23,22,21,20,19,18,17,16,15,0,10,10],
             [28,27,26,25,24,23,22,21,20,19,18,17,16,15,0,10],
             [29,28,27,26,25,24,23,22,21,20,19,18,17,16,15,0],])

Min_order = [0]
Min_time =[100]
x_vals = []
y_vals = []
for hardloop in range(0,1000,1):
    "Order in which qualitech will be produced"
    Qualitech_production_order = []

    #Insert function here
    import random
    Qualitech_production_order = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
    random.shuffle(Qualitech_production_order)

    #Insert function here
    #print ""
    #print "Qualitech_production_order"
    #print str(Qualitech_production_order)[1:-1]

    #Production time needed per qualitech product (Flushing included)
    Time_to_produce_qualitech = []

    for i in range(0,len(Qualitech_demands)):

        #Calculate production time needed per qualitech product
        if Qualitech_demands[Qualitech_production_order[i]] == 0:
            Flushing_time = 0
        elif i == 0:
            Flushing_time = 17.5
        elif i > 0:
            Flushing_time = Qualitech_flushing_array[
                Qualitech_production_order[i-1],Qualitech_production_order[i]]
        else:
            Flushing_time = 17.5
        Time_to_produce_qualitechs = (Qualitech_demands[
            Qualitech_production_order[i]]/Units per min) + Flushing_time
        Time_to_produce_qualitech.append(Time_to_produce_qualitechs)
    Time_to_produce_qualitech_new = []

    #print ""

```



```

#print "Mixer time available for Qualitech"
Mixer_available_for_Qualitech = [ round(elem, 2) for elem in
                                Mixer_available_for_qualitech ]
#print str(Mixer_available_for_Qualitech)[1:-1]

n = 0
l = 0
while n < 16:
    if l > 12:
        Append = Time_to_produce_qualitech[n]
        n = n + 1
    elif Time_to_produce_qualitech[n] <= Mixer_available_for_qualitech[l]
        and n < 15:
        if Time_to_produce_qualitech[n] + Time_to_produce_qualitech[n+1]
            <= Mixer_available_for_qualitech[l] and n < 14:
            if Time_to_produce_qualitech[n] +
                Time_to_produce_qualitech[n+1] +
                Time_to_produce_qualitech[n+2]
                <= Mixer_available_for_qualitech[l] and n < 13:
                if Time_to_produce_qualitech[n] +
                    Time_to_produce_qualitech[n+1] +
                    Time_to_produce_qualitech[n+2] +
                    Time_to_produce_qualitech[n+3] <=
                    Mixer_available_for_qualitech[l] and n < 12:
                    if Time_to_produce_qualitech[n] +
                        Time_to_produce_qualitech[n+1] +
                        Time_to_produce_qualitech[n+2] +
                        Time_to_produce_qualitech[n+3] +
                        Time_to_produce_qualitech[n+4] <=
                        Mixer_available_for_qualitech[l] and n < 11:
                        if Time_to_produce_qualitech[n] +
                            Time_to_produce_qualitech[n+1] +
                            Time_to_produce_qualitech[n+2] +
                            Time_to_produce_qualitech[n+3] +
                            Time_to_produce_qualitech[n+4] +
                            Time_to_produce_qualitech[n+5] <=
                            Mixer_available_for_qualitech[l] and n < 10:
                            if Time_to_produce_qualitech[n] +
                                Time_to_produce_qualitech[n+1] +
                                Time_to_produce_qualitech[n+2] +
                                Time_to_produce_qualitech[n+3] +
                                Time_to_produce_qualitech[n+4] +
                                Time_to_produce_qualitech[n+5] +
                                Time_to_produce_qualitech[n+6] <=
                                Mixer_available_for_qualitech[l] and n < 9:
                                if Time_to_produce_qualitech[n] +
                                    Time_to_produce_qualitech[n+1] +
                                    Time_to_produce_qualitech[n+2] +
                                    Time_to_produce_qualitech[n+3] +
                                    Time_to_produce_qualitech[n+4] +
                                    Time_to_produce_qualitech[n+5] +
                                    Time_to_produce_qualitech[n+6] +
                                    Time_to_produce_qualitech[n+7] <=
                                    Mixer_available_for_qualitech[l]
                                    and n < 8:
                                    Append = (Time_to_produce_qualitech[n]
                                        + Time_to_produce_qualitech[n+1] +
                                        Time_to_produce_qualitech[n+2] +
                                        Time_to_produce_qualitech[n+3] +
                                        Time_to_produce_qualitech[n+4] +
                                        Time_to_produce_qualitech[n+5] +
                                        Time_to_produce_qualitech[n+6] +
                                        Time_to_produce_qualitech[n+7])

```

```

        n = n + 8
    else:
        Append = (Time_to_produce_qualitech[n]
        + Time_to_produce_qualitech[n+1] +
        Time_to_produce_qualitech[n+2] +
        Time_to_produce_qualitech[n+3] +
        Time_to_produce_qualitech[n+4] +
        Time_to_produce_qualitech[n+5] +
        Time_to_produce_qualitech[n+6])
        n = n + 7
    else:
        Append = (Time_to_produce_qualitech[n] +
        Time_to_produce_qualitech[n+1] +
        Time_to_produce_qualitech[n+2] +
        Time_to_produce_qualitech[n+3] +
        Time_to_produce_qualitech[n+4] +
        Time_to_produce_qualitech[n+5])
        n = n + 6
    else:
        Append = (Time_to_produce_qualitech[n] +
        Time_to_produce_qualitech[n+1] +
        Time_to_produce_qualitech[n+2] +
        Time_to_produce_qualitech[n+3] +
        Time_to_produce_qualitech[n+4])
        n = n + 5
    else:
        Append = (Time_to_produce_qualitech[n] +
        Time_to_produce_qualitech[n+1] +
        Time_to_produce_qualitech[n+2] +
        Time_to_produce_qualitech[n+3])
        n = n + 4
    else:
        Append = (Time_to_produce_qualitech[n] +
        Time_to_produce_qualitech[n+1] +
        Time_to_produce_qualitech[n+2])
        n = n + 3
    else:
        Append = (Time_to_produce_qualitech[n] +
        Time_to_produce_qualitech[n+1])
        n = n + 2
    else:
        Append = (Time_to_produce_qualitech[n])
        n = n + 1
    Time_to_produce_qualitech_new.append(Append)
    l = l + 1

Extra_time_list = []
for i in range(0,len(Time_to_produce_qualitech_new)):
    if Time_to_produce_qualitech_new[i] >
        Mixer_available_for_qualitech[i]:
        Extra_time_list.append(Time_to_produce_qualitech_new[i]-
        Mixer_available_for_qualitech[i])
    else:
        Extra_time_list.append(0)

if sum(Time_to_produce_qualitech_new) >
    sum(Mixer_available_for_qualitech)*Percentage_mixer_available:
    Extra_time = sum(Time_to_produce_qualitech_new) -
    sum(Mixer_available_for_qualitech)*Percentage_mixer_available
    + sum(Extra_time_list)

```

```

else: Extra_time = sum(Extra_time_list)

#print ""
#print "Time_to_produce_qualitech_new"
Time_to_produce_Qualitech_new = [ round(elem, 2)
                                for elem in Time_to_produce_qualitech_new ]
#print str(Time_to_produce_Qualitech_new)[1:-1]

#print""
#print "Amount of extra days needed"
#print round(Extra_time/(60*24),2)

Total_production_time_for_Q_and_E = sum(Production_time_per_product
                                        /(60*24) + Extra_time/(60*24))

#print ""
#print "Total production days required to produce Easymix and Qualitech"
#print round(Total_production_time_for_Q_and_E,2)

#'''For random only'''
Time_to_produce_Qualitech_new.append(Extra_time/(60*24))
Time_to_produce_Qualitech_new.append(Total_production_time_for_Q_and_E)
#print str(Time_to_produce_Qualitech_new)[1:-1]

if Total_production_time_for_Q_and_E < Min_time[0]:
    Min_time.pop(0)
    Min_time.append(Total_production_time_for_Q_and_E)
    Min_order.pop(0)
    Min_order.append(Qualitech_production_order)
else:
    Nothing = 0

x_vals.append(hardloop)
y_vals.append(Total_production_time_for_Q_and_E)

from matplotlib import pyplot as plot
Min_time = Min_time[0]
Min_time = round(Min_time,3)
plot.figure(1)
plot.plot(x_vals,y_vals,'k-s')
plot.ylabel('Value obtained from iteration',fontsize=16)
plot.xlabel('Number of iterations',fontsize=16)
plot.title('Graph of values from random iterations with minimum value = '
          + str(Min_time) + ' Days')

plot.show()
print Min_order
end = (time.clock() - start)
print end

```

```
'''Random ordering heuristic function'''
import random
Qualitech_production_order = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
random.shuffle(Qualitech_production_order)

'''Smallest flushing times ordering heuristic function'''
for k in range(0,len(Qualitech_demands)):
    Order_nr = k
    Qualitech_production_order.append(k)

'''Largest flushing times ordering heuristic function'''
for k in range(0,len(Qualitech_demands)):
    Order_nr = k
    Qualitech_production_order.append(k)
Qualitech_production_order.reverse()

'''Smallest demands first ordering heuristic function'''
Qualitech_demands_copy = list(Qualitech_demands)
for i in range(0,len(Qualitech_demands)):
    val = min(Qualitech_demands_copy)
    product_nr = Qualitech_demands.index(val)
    Qualitech_production_order.append(product_nr)
    Qualitech_demands_copy.remove(val)

'''Largest demands first ordering heuristic function'''
Qualitech_demands_copy = list(Qualitech_demands)
for i in range(0,len(Qualitech_demands)):
    val = max(Qualitech_demands_copy)
    product_nr = Qualitech_demands.index(val)
    Qualitech_production_order.append(product_nr)
    Qualitech_demands_copy.remove(val)
```

Appendix D: Random heuristic 30 runs

	Total time required for qualitech	Extra days needed	Total production days needed
	7803.50	1.06	19.79
	7806.50	0.92	19.65
	7778.50	1.33	20.06
	7781.50	0.91	19.64
	7796.00	1.02	19.75
	7805.50	0.93	19.66
	7794.50	0.83	19.56
	7816.50	1.31	20.04
	7806.50	0.77	19.50
	7787.50	0.87	19.60
	7783.50	0.81	19.54
	7803.50	0.75	19.48
	7795.50	1.34	20.07
	7784.00	0.70	19.43
	7796.50	1.16	19.89
	7790.50	1.37	20.10
	7794.50	1.27	20.00
	7792.50	0.98	19.71
	7797.50	0.80	19.53
	7782.50	0.95	19.68
	7810.50	0.73	19.46
	7795.50	1.08	19.81
	7790.50	1.28	20.01
	7785.50	0.93	19.66
	7792.50	1.12	19.85
	7805.50	0.97	19.70
	7784.50	0.97	19.70
	7799.50	0.85	19.58
	7798.50	0.63	19.36
	7790.50	1.07	19.80
AVERAGE	7795.00	0.99	19.72
MEDIAN	7795.00	0.96	19.69
MIN	7778.50	0.63	19.36
MAX	7816.50	1.37	20.10

Table 19: New model using the Random rule for 30 iterations.