

https://dspace.org/

The application of meta heuristic global optimization algorithms to scheduling problems within the brewing industry

Nash, Graham

2008-05-21T10:05:09Z
http://hdl.handle.net/10539/4862

Downloaded from a DSpace repository. https://dspace.org/

The Application of Meta Heuristic Global Optimization Algorithms to Scheduling Problems within the Brewing Industry

Graham Nash

A thesis submitted to the Faculty of Science, University of the Witwatersrand, Johannesburg in fulfillment of the requirements for the the degree of Master of Science. Johannesburg, 2007

Declaration

I declare that this thesis is my own, unaided work. It is being submitted to the Faculty of Science, University of the Witwatersrand, Johannesburg. It has not been submitted before for any other degree or examination in any other University.

_____

Signature of candidate

_____

Date

Abstract

In this thesis we have presented a mathematical model of a scheduling problem which arises in the Brewing Industry. We have implemented two different types of global optimization algorithms to find the global minimum of the problem. Several instances of the scheduling problem are considered and the results thereof are presented. The results show that significant savings can be made if the global optimization techniques are used in brewery Industry.

Keywords: Simulated Annealing, Tabu Search, Flexible Flow Shop Problem, Brewery Scheduling

Acknowledgments

I would like to thank my supervisor, Professor Montaz Ali, for all his enthusiasm and support. Without his commitment to this thesis it would never have materialized.

I would also like to thank my parents, who inspired me to take on the challenge of studying once again, for all their support.

A special thanks to Ido Rieger who gave me the opportunity to work in the brewing environment, for his insights as well as encouragement to tackle real world optimization problems.

Lastly to Tonielle Eltze whose patience knew no bounds, thank you for your continued support in all that I do.

I would also like to acknowledge the financial support granted by the NRF.

Contents

1 Introduction **2**

2 Problem Definition **4**
2.1 Optimization within the Brewing Industry 4
2.1.1 Process Description 4
2.1.2 Scheduling 7
2.2 Scheduling Problems 9
2.2.1 The Mathematical Model of the Flexible Flow Shop Problem (FFS) . . 9
2.3 Scheduling Problems within the Brewing Industry 15

3 Combinatorial Optimization Techniques **17**
3.1 Tabu Search 17
3.1.1 TS Methodology 17
3.1.2 Memory Structures 20
3.1.3 TS Flowchart 23
3.1.4 The TS Algorithm 23
3.1.5 Intensification and Diversification 25
3.2 Simulated Annealing 26
3.2.1 The Metropolis Procedure 27
3.2.2 The Simulated Annealing (SA) Method 27
3.2.3 Cooling Schedules 28

4 Implementation of TS and SA to Brewing Optimization **32**
4.1 Phase 1: Filtration and Packaging 32
4.1.1 Scenario 1 39
4.1.2 Scenario 2 44
4.1.3 Scenario 3 45
4.1.4 Scenario 4 46

5 Appendix A **49**

List of Figures

2.1 Macro view of Brewing Resources 5

3.1 The Simple Descent Method 18
3.2 Optimal Solution Path obtained by the Simple Descent Method for Example 1 20
3.3 Sensitivity Analysis of Tenure for Example 1 22
3.4 The Flow Chart of Tabu Search 24
3.5 Simulated Annealing: High vs. Low T_0 31

4.1 Conceptual Layout of a Brewery 33
4.2 TS Solution Path for a single case of Scenario 1 41
4.3 SA Solution Path for a single case of Scenario 1 42
4.4 Initial and Optimized allocation of jobs for Filter 1 and Filter 2 in Scenario 1 . 43
4.5 Conceptual Layout of a Brewery 48

List of Tables

2.1 Examples of brand recipes 8

3.1 Thermodynamic Simulation vs. Combinatorial Optimization 26

4.1 SKU Weekly Demands 34
4.2 Pack Types 34
4.3 Filtration Plant Attributes 35
4.4 Packaging Line Attributes 35
4.5 Initial assignment of the Packaging line and Filtration Jobs 38
4.6 % of SKU allocated to Line 2 and Line 4 45
4.7 FV Data 46
4.8 Summary of Results 47
4.9 Results Summary of Scenarios 1 to 4 48

5.1 Scenario 1:Tabu Search Results - 100 Iterations 49
5.2 Scenario 1:Tabu Search Results - 150 Iterations 50
5.3 Scenario 1:Tabu Search Results - 200 Iterations 51

Chapter 1

Introduction

There are several scheduling problems that arise in the Brewing industry. In this thesis we will focus on the scheduling of the filtration and packaging regimes of a brewery. Optimized filtration and packaging schedules allow for better utilization of resources, thus enabling greater volumes of beer to be packaged.

The problem arises in that at any brewery there is a number of different packaging job's that need to be completed each week. These jobs are required to be filtered and then packaged, and have sequence dependent setup times. Due to a limited number of filtration and packaging equipment within any brewery there is a requirement for the optimized scheduling of jobs. This requirement becomes increasingly evident as the number of stock keeping units (SKUs) a brewery must produce increases.

From the results of the thesis we clearly demonstrate the need for such optimization in the brewing process. Although the problem presented is a true optimization problem, to the best of our knowledge, there are no research papers on the subject in current literature. Hence we intended to study the problem.

We have first described the problem and then presented a mathematical representation of it. We have used two different algorithms to optimize the problem. Several instances of the problem were considered and extensive numerical testing was done. Results are presented which clearly show that a brewery can make significant gains if scheduling is improved.

In Chapter 2 we present the background to the problem, giving a high level description of each component within the brewing process. We conclude the chapter by introducing the Flexible Flow Shop Problem and present the mathematical model thereof, as it is used to describe the filtration and packaging problem,

Chapter 3 introduces two combinatorial optimization techniques, namely, Simulated Annealing (SA) and Tabu Search (TS). We describe both algorithms in detail and use an example scheduling problem on which to perform tests in order to show the significance of various

parameter settings.

In Chapter 4 we implement SA and TS on a formulated brewery scheduling problem. A phased approach is adopted, as we apply the algorithms on a base problem and gradually increase the complexity of the problem.

Concluding remarks are made in Chapter 5.

Chapter 2

Problem Definition

We begin with a brief description of the brewing process in order to provide the reader with some insights into the background of the problem at hand. We then introduce the flexible flow shop problem and a mathematical model thereof, from which we are able to optimize components of the brewing problem.

2.1 Optimization within the Brewing Industry

There is a real need for optimization within the brewing industry. The optimized scheduling of operations within any process industry can allow one to maximize profits by minimizing the utilization of resources [1]. Within the brewing industry in particular, precise scheduling is required in order to ensure that demand is met; this is due to the high level of limited resources that must be shared whilst producing beer of different brands with different process times. The main resources within the brewery are represented in Figure 2.1.

2.1.1 Process Description

The Brewhouse

The brew house marks the beginning of the brewing process. It is here where the malt, hops, yeast and water, (the main ingredients required to brew beer), are combined together. The malt is ground with other adjuncts and added to water. During this process starch is converted into fermentable sugars and natural enzymes that convert proteins into the nutrients needed to assist the growth of yeast during fermentation. Hops is also added and during the transfer of brews from the brew house to the fermentation area, the yeast is added [2]. The brewlength of a brewhouse refers to the volume of a single brew, and is dependent on the capacity of the brewhouse.

The scheduling of the sequence and batch size of brews is a complex task, and efficient scheduling ensures the availability of beer for packaging. Inefficient scheduling at this stage of the process can affect the overall capacity of the brewery. For example, if one brews too large

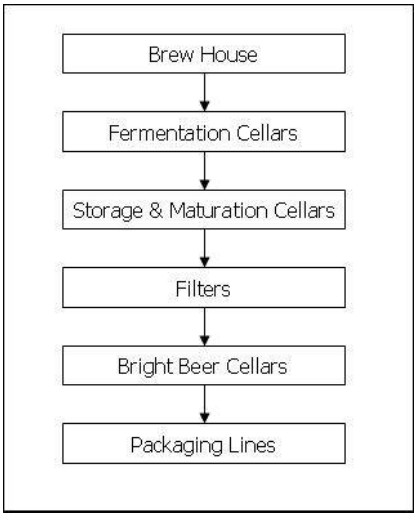


Figure 2.1: Macro view of Brewing Resources

a batch of a brand that requires long process times within the latter stages of the brewing process, a bottleneck will be created. Elements which need to be considered in this phase of the process are:

- Brew length,
- Brand specific process times,
- Cycle time of brands[1],
- Number of tanks available in fermentation/unitank cellars,[2]
- Brand restrictions for each stream of brew house,
- Connectivity,
- Equipment CIP (cleaning in progress) as well as different cleaning regimes for different brands, and
- Yeast availability and propagation[3].

[1]The cycle time is the longest time a brand is in a particular piece of brew house equipment.

[2]Fermentation and unitank cellars occur in the next step of the brewing process after brewing.

[3]Each brand requires a certain amount of yeast as part of its recipe, the efficient propagation thereof means that one must brew enough in order to produce yeast for further brews.

Fermentation and 'Storage and Maturation' Vessels

Beer produced in the brew house is first transferred to the fermentation vessels (FV) wherein the yeast begins the natural process of fermenting the sugars into alcohol and carbon dioxide. Yeast is drawn off the bottom of the tank once the fermentation process is complete. Beer is transferred from the FVs, via racking stations, to the 'storage and maturation' vessels (SVs)[4]. A racking station contains a centrifuge through which beer is moved in order to remove impurities. Within the SVs the fermentation process continues at a slower rate as the beer is stored at sub zero temperatures. Where unitanks (UNIs) are used, the whole process of fermentation and maturation occurs within a single tank [2]. Orders of magnitude of the Fermentation and 'Storage and Maturation' processes are presented in a later section.

The role of optimization in this phase of the brewing process is to minimize the amount of time that beer remains in the FV's after the fermentation process is complete, this amount of time is known as aging. Elements which need to be considered in this phase of the process are:

- Number of SVs and ready FVs,
- Tank capacities,[5]
- Connectivity,
- Equipment CIP.

Filtration, Bright Beer Cellars and Packaging Lines

In this part of the process beer is transferred from the SVs and UNIs to the 'bright beer' vessels (BBTs), via the filtration plants. The BBTs serve as buffer tanks for the packaging lines, and beer must remain there for a minimum number of hours, ie. 4 hours. The capacity of the filters are rated in hecto liters (Hl) per hour, and as with the SVs, UNIs and BBTs may be subject to connectivity constraints [2].

During packaging it is essential to maximize the utilization of the line, i.e to make sure one meets the demands in the demand week. Hence one needs to optimize the decision of how much of which brand to filter. These decisions should also minimize the number of brand and pack changes on the lines. Effective decisions should not only result in the volumes of the demand week being met, but also in freeing up tanks in the SVs so as to allow for beer to be transferred from the FVs. The constraints involved in this process are:

- Connectivity of BBTs, filters and packaging lines,

[4]This transfer does not necessarily need to be facilitated via racking stations, depending on the requirements of the brand.

[5]The capacity of a tank is typically described as the number of brews one can hold in that tank, the volume of the brew is determined by the brewlength.

- Brand restrictions for filters, BBTs and packaging lines,
- Maintenance period for packaging lines,[6]
- Washout for filters (*i.e After every z Hl the filter needs to be washed out*),
- Rate of filters (HL/hr),
- Capacity of BBTs and SVs,
- Minimum run lengths for filters and packaging lines (*Typically an hour of production is the minimum*),
- Equipment CIP.

2.1.2 Scheduling

It is common for the scheduling of a brewery to originate from a six-month forecast produced by the management of a brewery. The forecast is continuously revised, and a rolling six week forecast is used, from which the weekly plan to be implemented is based. Hence the brewery receives a volume demand by stock keeping unit (SKU), for example 3500Hl brand A to be packaged in 500ml bottles. The following schedules need to be created based on the demand:

- Brewing (Brew House to FV),
- Racking (FV to SV),
- Filtration (SV to BBT),
- Packaging (BBT to Packaging Line).

Before the brewing schedule can be created, the number of brews need to be calculated based on the forecasted demands. Each brand has a particular recipe with regard to a dilution factor and process times. The process times indicate the length of time that the beer is required to remain in any particular stage of the process, ie. fermentation, storage and maturation. Packaging volumes are quoted at standard (diluted) gravity and need to be converted to volumes in high gravity (concentrated) form when establishing the number of brews required. A brand specific dilution factor is used to obtain the high gravity requirements from the demands quoted at standard gravity. To demonstrate the process, let us consider two hypothetical brands: brand A and brand B. The recipes for brands A and B are described in Table 2.1. The second column of Table 2.1 presents the dilution factor, whilst columns three to six present the individual and total process times.

[6]Scheduled maintenance periods are allocated to the packaging lines

Brand	Dilution Factor (1Hl:1Hl)	Process Time (Days)			
		FV	SV	UNI	Total (incl. Brewing)
A	1.231	9	7	16	17
B	1.625	10	20	30	31

Table 2.1: Examples of brand recipes

Thus, if the packaging demand for brands A and B in Table 2.1 are required for the week starting at t_0, (i.e. day zero), then they must be brewed at t_{-17} and t_{-31} respectively. If the packaging demand for brand A was 1231Hl then the volume to be brewed, without taking losses into consideration would be 1000Hl, due to the dilution factor of 1.231. Hence to calculate the number of brews required, the following equation may be used [2]:

$$NB = \frac{D}{1 - \frac{L}{100}} \times \frac{1}{(DF \times R)} \tag{2.1}$$

where,

- NB: Number of brews required,
- D: Demand Volume,
- DF: Dilution Factor
- L: Losses [7] and
- R: Brew Length.

The number of brew is therefore given by $NB = 11.1 \cong 11$, for the following parameter values: $D = 1231Hl$, $DF = 1.231$, $L = 10$ and $R = 100Hl$.

[7] This is the percentage of beer lost through the process from the brew house to packaging.

2.2 Scheduling Problems

The application of scheduling ranges from manufacturing to the services industry. Despite there are numerous types of scheduling problems, they are often grouped into the following five main classes as described by Jungwattanakit et al [3].

Workshops with only one machine: there is only one machine which is to be used when scheduling a given number of jobs, subject to specified constraints.

Flowshop: there is more than one machine and each job must be processed on each of the machines. The number of operations, $j = 1, ..., k$, for each job must be equal to the number of machines, the j^{th} operation of each job being processed on machine j, (hence there are k stages).

Jobshop: this problem is the same as that of the flowshop problem, but in this instance, each job has an individual processing order assigned for its operations.

Openshop: this problem is the same as that of the flowshop problem, however the processing order of the jobs is arbitrary and any ordering will do.

Mixed workshop: there is a subset of jobs to be completed that must follow a pre-determined path as well as other jobs which must be scheduled in order to minimize the objective function.

The scheduling problem of the brewery introduced in section 2.1.1 falls into the flexible flowshop (FFS) environment, which is a generalization of the classical flowshop problem. Within this environment, there are k stages, where any given stage may only consist of one machine, but in at least one stage there will exist more than one machine. The jobs to be scheduled are required to visit each of the stages from 1 to k. A machine can only process at most one operation of a particular job at a time. Thus the problem involves the assignment of the jobs to a machine at each stage and sequencing the jobs assigned to the machine so that some optimality criteria are minimized [4].

2.2.1 The Mathematical Model of the Flexible Flow Shop Problem (FFS)

In this section we present a number of features of FFS, after which we present the mathematical model of FFS.

FFS with Parallel Non-identical Machines

The flexible flow shop problem, as described in [4], has received much attention throughout the literature, but the focus of these studies are mainly concentrated on problems with identical machines. In many applications it is rare that identical machines will exist. In the case of a real world problem, such as a brewery, it is common that new machines will run side by side older less efficient machines. For example, within many brewery's filtration plants, there may be both new 'candle' filters as well as the older 'plate and frame' filters in operation. This is also true with regard to packaging lines, where the cost of replacing a line is high. The older machines will perform the same operations as the newer machines, but will in general have longer processing times.

Jobs and Their Operations

In the typical flexible flow shop problem there exists a set of n independent jobs, $j \in \{1, 2, \ldots, n\}$, with due dates, $d_1, \ldots, d_n$ and we assume that the release dates, $r_1, \ldots, r_n$, of the jobs are non-negative. Each job j consists of $k (k \geq 2)$ different operations which have to be processed.

Machine Dependent Processing Times

Each job j has a processing size ps_j^t, (with units in Hl), for each stage t, $t \in \{1, \ldots, k\}$ and at each stage t, there is a set of m^t unrelated parallel machines, $i \in \{1, 2, \ldots, m^t\}$. Thus machine i at stage t can process job j at the relative speed of v_{ij}^t, (with units in Hl/hr). Hence, the processing time p_{ij}^t of job j on machine i at stage t is equal to ps_j^t / v_{ij}^t,(with units in hrs).

Setup Times

Setup time refers to the amount of time required to prepare machinery before a given job is to be processed. Two types of setup times are considered in the problem, machine-dependent (changeover) setup times and sequence-dependent setup times. Machine-dependent setup times depend on the machine to which the job is assigned, and is assumed only to occur when a job is assigned to be the first job processed on a particular machine at some stage. This means that the changeover time ch_{ij}^t of job j depends on machine i performing it if job j is assigned to the first position of the machine at stage t.

Sequence-dependent setup times can be described as the time considered between jobs, hence the setup time s_{lj}^t between job l and job j at stage t, where job l is processed directly before job j on the same machine, may have different values depending on both the job completed and the job to be processed [4]. In some instances sequence dependent setup times could depend on machine i, however we assume that it does not.

Problem Assumptions

The problem assumes the following:

- Processing of jobs cannot be interrupted.
- The operations of a job have to take place sequentially.
- No machine can process more than one job at a time.
- No job can be processed by more than one machine at a time.
- In some of the stages there may be only one machine, but there must exist at least one stage with multiple machines.
- All setup times are known constants.

The Objective Function

The makespan of the problem refers to the time it takes to complete all n jobs [5], and will be equal to the completion time of the last job to leave the system. Thus if the completion time of job j is defined to be C_j, the makespan may be defined as,

$$C_{MAX} = max_{j\in\{1,\dots n\}}\{C_j\}. \tag{2.2}$$

Let us define $U_j = 1$ if the due date for job j is smaller than the completion time C_j of job j, otherwise $U_j = 0$. Thus the total number of tardy jobs can be defined as,

$$\eta_{Ta} = \sum_{j=1}^{n} U_j. \tag{2.3}$$

The objective of the problem is to minimize a weighted average of the makespan and number of tardy jobs. Thus, the objective value is defined by

$$\lambda C_{MAX} + (1-\lambda)\eta_{Ta}. \tag{2.4}$$

Notation and Formulation of a Mathematical Model

Jungwattanakit et al [4] describe the problem with a 0-1 mixed integer linear programming formulation for FFS. The model is claimed to solve problems with up to six jobs and four stages in acceptable time using an entry level Intel Pentium 4 2GHz PC.

The following notation is used,

Indices:

j, l**:** job index, $j, l = 1, 2, 3, \ldots, n$.

t**:** stage index, $t = 1, 2, 3, \ldots, k$.

i**:** machine index, $i = 1, 2, 3, \ldots, m^t$ at stage t.

Data:

r_j**:** release date of job j.

d_j**:** due date of job j.

L_j **:** the amount of time that passes between a job j's due date and its completion, $L_j > 0$.

B**:** a constant greater than the sum of all completion and setup times.

s^t_{jl}**:** setup time between job j and job l at stage t where $j \neq l$.

ch^t_{ij}**:** setup time of job j if job j is assigned to machine i at the first position at stage t.

ps^t_j**:** processing size of job j at stage t.

v^t_{ij}**:** relative speed of machine i at stage t for job j.

a^t_i**:** time when machine i at stage t becomes available.

Variables:

X^t_{ijl}**:** a Boolean variable; $X^t_{ijl} = 1$ if job j is scheduled immediately before job l on machine i at stage t, and $X^t_{ijl} = 0$ otherwise.

O^t_j**:** operating time of job j at stage t.

C^t_j**:** completion time of job j at stage t.

C_{MAX}**:** the makespan.

U_j**:** a Boolean variable; $U_j = 1$ if the job is tardy, and $U_j = 0$ otherwise

Ta_j**:** tardiness of job j.

η_{Ta}**:** the total number of tardy jobs in the schedule.

The Optimization Problem

The problem can be formulated mathematically as follows:

$$minimize \qquad \lambda C_{MAX} + (1-\lambda)\eta_{Ta} \tag{2.5}$$

subject to:

$$\sum_{i=1}^{m^t}\sum_{j=0}^{n} X_{ijl}^t = 1, \qquad \forall t, l \tag{2.6}$$

$$\sum_{i=1}^{m^t}\sum_{l=1}^{n+1} X_{ijl}^t = 1, \qquad \forall t, j \tag{2.7}$$

$$\sum_{l=1}^{n+1} X_{i0l}^t = 1, \qquad \forall t, i \tag{2.8}$$

$$\sum_{j=0}^{n} X_{ij(n+1)}^t = 1, \qquad \forall t, i \tag{2.9}$$

$$X_{ijj}^t = 0, \qquad \forall t, i, j \tag{2.10}$$

$$\sum_{j=0}^{n} X_{ijl}^t = \sum_{j=1}^{n+1} X_{ilj}^t, \qquad \forall t, i, j \tag{2.11}$$

$$X_{ijl}^t \in \{0,1\}, \qquad \forall t, i, j, l; j = 0; l = n+1 \tag{2.12}$$

$$O_j^t = \sum_{i=1}^{m^t}\sum_{l=1}^{n+1} \frac{ps_j^t}{v_i^t j} X_{ijl}^t, \qquad \forall t, j \tag{2.13}$$

$$C_l^t - C_j^t \geq s_{jl}^t + O_j^t + ((\sum_{l=1}^{m^t} X_{ijl}^t) - 1)B, \qquad \forall t, j, l; j \neq l \tag{2.14}$$

$$C_j^t \geq 0, \qquad \forall t, j \tag{2.15}$$

$$C_l^t - C_j^{t-1} \geq \sum_{i=1}^{m^t}\sum_{j=1}^{n} X_{ijl}^t s_{jl}^t + \sum_{i=1}^{m^t} ch_{il}^t X_{i0l}^t + O_j^t, \qquad \forall t, l \tag{2.16}$$

$$C_j^0 = r_j, \qquad \forall j \tag{2.17}$$

$$C_j^t \geq \sum_{i=1}^{m^t} a_i^t X_{i0j}^t + \sum_{i=1}^{m^t} ch_{il}^t X_{i0l}^t + O_j^t, \qquad \forall t, l \tag{2.18}$$

$$C_{MAX} \geq C_j^t, \qquad \forall t, l \tag{2.19}$$

$$Ta_j \geq C_j^t, \qquad \forall t, l \tag{2.20}$$

$$Ta_j \geq 0, \qquad \forall t, l \tag{2.21}$$

$$U_j \leq BTa_j, \qquad \forall t, l \tag{2.22}$$

$$BU_j \geq Ta_j, \qquad \forall t, l \tag{2.23}$$

$$\eta_{Ta} = \sum_{j=1}^{n} U_j, \qquad \forall t, l \tag{2.24}$$

$$U_j \in \{0,1\}, \qquad \forall t, l \tag{2.25}$$

Constraints (2.6) to (2.12) are responsible for ensuring the feasibility of the partial schedule on each machine at each stage:

- Constraints (2.6) and (2.7) ensure that only one job is assigned to any particular position at each stage by having only one discrete proceeding and following job.
- Constraints (2.8) and (2.9) ensure that only one job is assigned to the first and last positions, respectively, on each machine at each stage.
- Constraint (2.10) ensures that after the job has finished a stage, it cannot be reprocessed at the same stage.
- Constraint (2.11) forces that a consistent sequence is built at each stage.
- Constraint (2.12) sets X_{ijl}^{t} to a binary variable.

Constraint (2.13) determines the machine dependent operating time of every job and constraints (2.14) to (2.18) determine the completion time of every job at stage t:

- Constraint (2.14) ensures that when job j follows job l on the same machine, it may only begin once job j is processed and the required setup time of the machine is completed. B represents a constant greater than the sum of all completion and setup times.
- Constraint (2.15) ensures that the completion time for each job is a positive value.
- Constraint (2.16) ensures that a job cannot be processed on stage $t+1$ before it has completed stage t.
- Constraint (2.17) ensures that a job cannot begin to be processed at stage 1 before its release date.
- Constraint (2.18) ensures that the first job to be processed on a machine cannot begin to be processed before the machine is available.

Constraints (2.19) links the makespan decision variable, whist constraints (2.20) and (2.21) determine the value of the tardiness:

- Constraint (2.20) determines the correct value of lateness (L_j)
- Constraint (2.21) specifies only the positive lateness as the tardiness ($Ta_j = max\{0, C_j^k - d_j\}$)

Constraints (2.22) to (2.25) link the decision variable to the number of tardy jobs, that is, if tardiness is larger than zero, the job is tardy; otherwise the job is not tardy.

Approaches to FFS

The flexible flow shop problem is a NP-hard problem, so algorithms that will find an optimal solution in polynomial time are unlikely to exist, on the other hand for smaller problems, one might be able to solve the problem exactly in reasonable time. Thus heuristic methods such as those based on the local neighborhood search are used to find approximate solutions e.g. TS and SA. Unlike procedures that rely on iterative improvements, procedures employing local neighborhood search will accept inferior solutions for further neighborhood searches, in order to escape local optima and to increase the chance of finding the global optimum [6]. Jones et al [7] state that seventy percent of articles utilize genetic algorithms to solve the flexible flow shop problem, twenty-four percent simulated annealing and only six percent make use of the tabu search.

2.3 Scheduling Problems within the Brewing Industry

The scheduling problems within the brewing industry that arise as a result of planning a filtration and packaging regime may be classified as a FFS problems as the constraints involved are very similar to FFS. However, instead of using makespan as part of our objective function we consider the total operating time incurred by the packaging lines. Hence our objective function is given by:

$$\lambda C_{TOTAL} + (1 - \lambda)\eta_{Ta}, \tag{2.26}$$

where,

$$C_{TOTAL} = \sum_{i=1}^{m^t} C_{MAXi}. \tag{2.27}$$

The sum in (2.27) is over all packaging lines and t is the final stage of the problem, i.e. t represents the packaging phase. Thus C_{TOTAL} is the sum of the completion times of the last job on each packaging line.

We also redefine ps_j^t to be the volume of job j processed at stage t and make the assumption that the processing of jobs can be interrupted. This is due to the fact that a job may be interrupted when processed on a filter or packaging line due to cleaning requirements. Interruptions may occur for the following reasons:

1. If a piece of machinery does not process a job for a specified period of time, a number of hours are needed for cleaning before processing any given job.

2. If a piece of machinery runs for a specified amount of time, the machine will require to be stopped and a number of hours may be needed for cleaning. The current job on the machine needs to be stopped, and the remainder of the job may only commence after the cleaning process has been completed.

Since the integrity of the packaging and filtration schedule must be maintained, constraints (2.6) to (2.12) are necessary. The completion time of every job needs to be determined at each stage of the problem, hence constraints (2.13) to (2.18) are vital. In order to ensure that the schedule is feasible constraints (2.19) to (2.25) are required to determine whether a job is tardy and the value of the tardiness.

Chapter 3

Combinatorial Optimization Techniques

In this chapter we introduce two different algorithms; they are Tabu Search (TS) and Simulated Annealing (SA). The algorithms are described in detail and in addition a simple sequencing problem is formulated to illustrate the various attributes and characteristics of each of the algorithms.

3.1 Tabu Search

Tabu search [8] is a meta-heuristic optimization algorithm that has been widely used in both academia and industry. To our knowledge it has not been implemented at a brewery but has been developed to solve similar type flow shop problems [9]. In the past, it has been primarily used to solve discrete optimization problems, although recently there have been versions designed for continuous optimization problems [10]. However, in this thesis we will implement it on a discrete scheduling problem.

TS is able to guide and modify the search process in such a way that solutions are generated surpassing the confines of local optimality. Hence TS is described as a 'meta-heuristic' approach [8].

3.1.1 TS Methodology

TS is an iterative algorithm which searches neighborhoods of the current solution in a restrictive manner for the global optimum. In order to understand the difference between a simple descent method and TS we first present the simple decent method, followed by TS. The goal of the simple descent method is to minimize an objective function $F(x), x \in X$. Each $x \in X$ has an associated neighborhood $N(x) \subset X$, and each solution $x' \in N(x)$ can be reached from x by an operation called a move. Hence the simple descent method iteratively considers moves to neighbor solutions that improve the current objective function value ending when no more

improving solutions can be found. The final solution found by the simple descent method is called the local optimum [8]. Figure 3.1 illustrates the flow chart of the simple descent method.

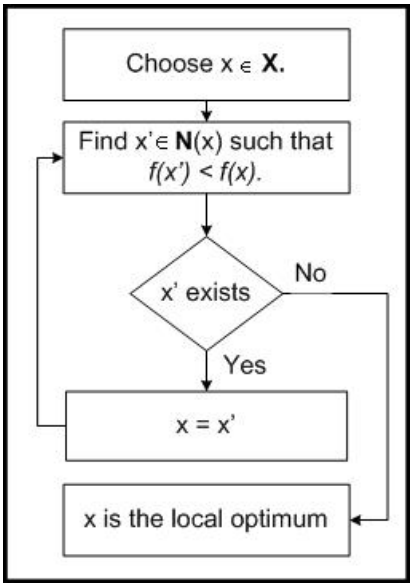


Figure 3.1: The Simple Descent Method

The simple descent method is an iterative method which attempts to improve the current solution by searching the neighborhood of the current solution. The simple descent method never accepts a solution which is 'worse' than the current solution.

In contrast to the simple descent method which is in most instances confined to only find local optima, TS finds near global or global optima. The solution space through which it searches is reduced through the use of tabu lists that record moves leading to 'less desirable' solutions. The so called tabu tenure is used to define the period of time for which an element of the tabu list is recorded, (the concept of tabu tenure is expanded on in Section 3.1.2). A 'less desirable' solution may however be accepted if it meets certain so called aspiration criteria, thus avoiding a local optimum.

Example 1

Let us consider a simple travelling salesman problem consisting of destinations $j = 1, \ldots, 10$. The objective of the problem is to minimize,

$$F(x) = \Sigma_{j=1}^{9} s_{x_j x_{j+1}}, \tag{3.1}$$

where each term in (3.1) depends on x, and x_j represents the job directly before x_{j+1}. Since the example only consists of one stage, it is not necessary to denote t. We use a simple descent algorithm with a random initial starting solution

$$x^0 = (10, 1, 9, 2, 8, 3, 7, 4, 6, 5),$$

with $F(x^0) = 44$. The sequence dependent setup times or distance, $s_{x_j x_{j+1}}$ are defined as:

$$s_{x_j x_{j+1}} = \begin{pmatrix} 0 & 0 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ & 0 & 0 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ & & 0 & 0 & 2 & 3 & 4 & 5 & 6 & 7 \\ & & & 0 & 0 & 2 & 3 & 4 & 5 & 6 \\ & & & & 0 & 0 & 2 & 3 & 4 & 5 \\ & & & & & 0 & 0 & 2 & 3 & 4 \\ & & & & & & 0 & 0 & 2 & 3 \\ & & & & & & & 0 & 0 & 2 \\ & & & & & & & & 0 & 0 \\ & & & & & & & & & 0 \end{pmatrix}$$

The global minimum value of the problem at $x^* = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$ is 0. After the simple descent method was applied for 1000 iterations or function calls, only 5 successful moves were obtained with the best value being a local minimum of 19. Figure 3.2 represents the solution path, where the x-axis representing the number of iterations or function calls performed, whist the y-axis represents the value of the objective function. TS was also implemented on the problem using a tenure of 3, the global optimum was found at iteration 91.

Formal explanation of TS

Let us first consider the problem of minimizing/maximizing $F(x)$ subject to $x \in X$, where X denotes the solution space. As in an ordinary local search; TS begins to iteratively move from one solution to the next until a chosen termination criterion is satisfied. A neighborhood, $N(x) \subseteq X$, exists for each $x \in X$, with each solution $x' \in N(x)$[1]. Unlike some conceptually

[1]The term 'move' describes the operation made to reach a solution

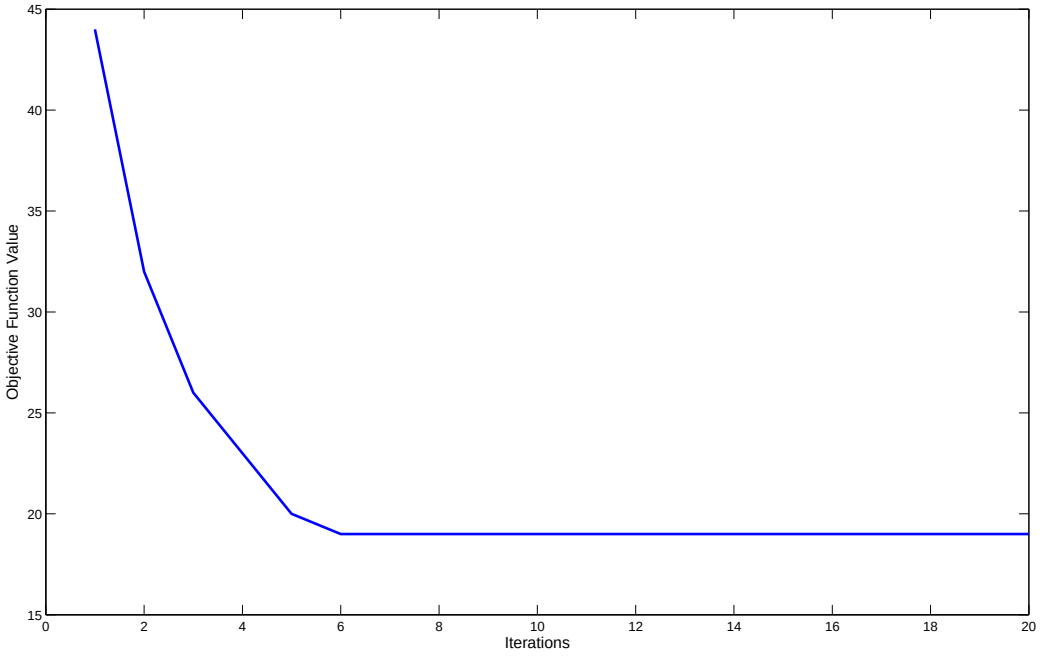


Figure 3.2: Optimal Solution Path obtained by the Simple Descent Method for Example 1

expensive algorithms where the neighborhood is searched completely for an optimum, or algorithms resulting in local optimality as they only allow moves to be made to neighborhood solutions which improve on $F(x)$; TS uses flexible memory structures that narrow down the examination of the neighborhood $N(x)$.

Memory structures result in the neighborhood $N(x)$ being modified to $N^*(x)$ where $N^*(x) \subseteq N(x)$. Elements of $N(x)$ that are excluded from $N^*(x)$ are classified as tabu, whilst those within $N^*(x)$ make up candidate solutions [8, 11].

At times, due to defined 'aspiration criteria', a solution within the Tabu list may be chosen allowing the TS to achieve better results, relying on the supposition that a bad strategic choice can yield more information than a good random choice.

3.1.2 Memory Structures

Memory structures used in TS may be classified as either short or long term memory. In TS strategies based on short term memory considerations, $N^*(x)$ typically is a subset of $N(x)$, and the tabu classification serves to identify elements of $N(x)$ excluded from $N^*(x)$. In TS strategies that include longer term memory considerations, $N^*(x)$ may also be expanded to include solutions not normally found in $N(x)$. Characterized in this way, TS may be viewed

as a dynamic neighborhood method. This means that the neighborhood of x is not a static set, but rather a set that can change according to the history of the search [8].

Short Term Memory

A TS strategy involving only short term memory may result in a solution x being visited more than once, but it is likely that the corresponding neighborhood $N^*(x)$ will be different each time.

Recency-based memory, is the most commonly used short term memory structure used in the tabu search algorithm. When using recency based memory, selected attributes that occur in solutions recently visited are labelled 'tabu-active'. Thus these solutions within N^* are prevented from being visited again for a specified number of iterations. Tabu tenure is used to define the period of recency based memory. Generally the iteration q at which a solution becomes tabu is recorded and then released at iteration $q + A$, where A is the equal to the tenure. There is no rule designed that provides the optimal tenure for particular classes of problems, and an effective tenure usually depends on the size of the neighborhood and structure of the problem at hand. If the tabu tenure is too small it may result in repeated objective function values; those that are too large can be recognized by a resulting deterioration in the quality of the solutions found [12].

Figure 3.3 represents the solution paths created by implementing TS on Example 1 with a tenure of 2 and 20 respectively. The x-axis of Figure 3.3 represents the number of iterations performed, whist the y-axis represents the value of the objective function. One can clearly see how with a tenure of 20 the quality of the values deteriorates towards the end, as well as the repetition of objective function values throughout the solution path that results with a tenure of 2.

Long Term Memory

When including longer term memories in TS, the likelihood of duplicating a previous neighborhood upon revisiting a solution is reduced. Hence making choices that repeatedly visit only a limited subset X is all but non-existent. Frequency-based memory is a type of long term memory, and is conceptually perceived as ratios. In general the numerators of these ratios could either represent a transition measure or a residence measure. A transition measure would express the number of iterations that an attribute enters or leaves the solutions generated; whilst a residence measure would express the number of iterations an attribute belongs to a solutions generated [8]. The denominators of the ratios may be,

- the total number of iterations,
- the sum or average of the numerators,
- the maximum value of the numerators.

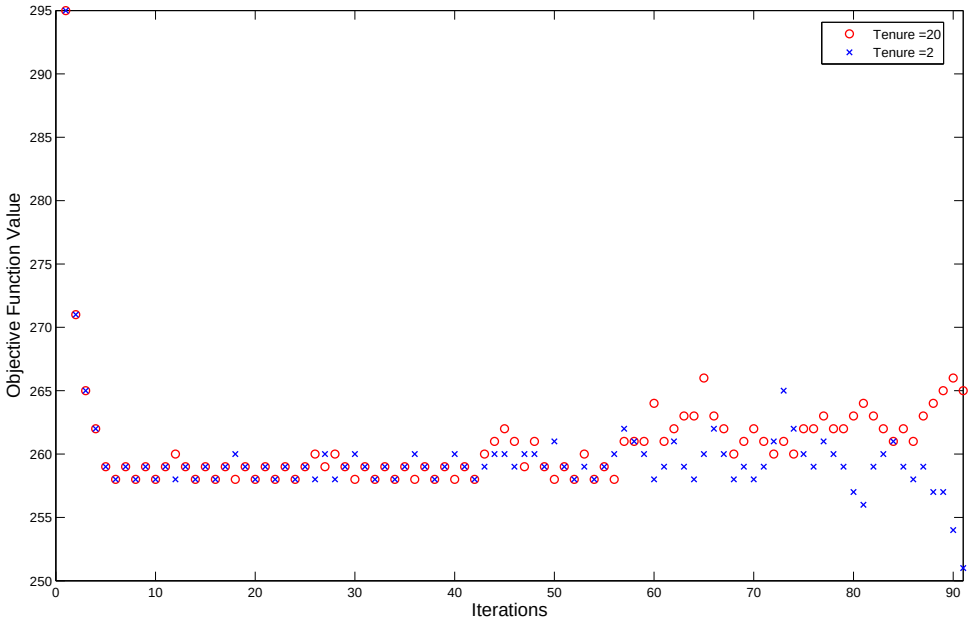


Figure 3.3: Sensitivity Analysis of Tenure for Example 1

Quality and Influence

The dimension of quality in TS memory refers to the ability to differentiate the merit of solutions visited in the search. For instance, memory can be used to identify elements that occur often in good solutions.

Explicit memory can be thought of as high quality memory as it records complete solutions, typically consisting of elite solutions visited during the search.

Influence refers to the impact of the choices made during the search, not only in terms of quality but also in terms of structure. Recording information about the influence of choices on a particular solution incorporates an additional level of learning. Influence is an underlying theme in attributive memory, which is used for guiding purposes. This type of memory records information about solution attributes that change in moving from one solution to the next. For example in a graph or network setting, attributes can consist of nodes or arcs that are added, dropped or repositioned by the moving mechanism. In production scheduling, the index of jobs may be used as attributes to inhibit or encourage the method to follow certain search directions [5].

3.1.3 TS Flowchart

TS begins by establishing a candidate list of moves from an initial solution. If a candidate solution is better than the current best solution or if it satisfies given criteria known as 'Aspiration Criteria', it is accepted as the new best solution. The move performed is recorded in a tabu list to prevent the algorithm from revisiting past solutions for a specified number of iterations. The algorithm ends when a given termination criteria is satisfied, for example a maximum number of iterations performed. The tabu search method is represented as a flow chart in Figure 3.4.

3.1.4 The TS Algorithm

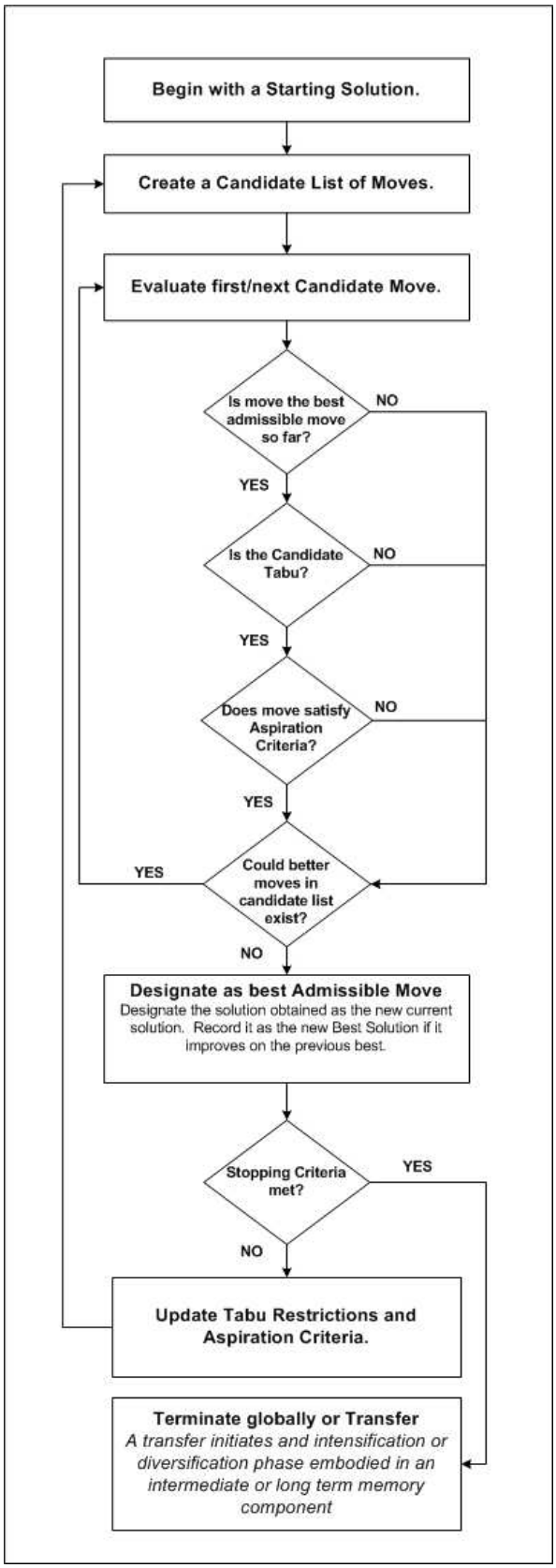
Here we present a step by step description of the TS algorithm based on the flowchart given in Figure 3.4. We have used the format of the algorithm as presented in [12], let:

1. H represent a selective history of the solutions encountered during the search.
2. $N(H, x^{now})$ represent the modified neighborhood that replaces $N(x^{now})$.
3. $Candidate_N(x^{now})$ refer to the candidate subset.

Step 1 (Initialization)
(A) Select a starting solution $x^{now} \in X$
(B) Record the best known solution by setting $x^{best} = x^{now}$ and define $bestcost = F(x^{best})$.
(C) Set history record H empty.

Step 2 (Choice and Determination)
Determine $Candidate_N(x^{now})$ as a subset of $N(H, x^{now})$. Select x^{next} from $Candidate_N(x^{now})$ to minimize $F(x)|_H$ over this set. (x^{next} is called a highest evaluation element of $Candidate_N(x^{now})$.) Terminate by a chosen cut-off rule.

Step 3 (Update)
Re-set $x^{now} = x^{next}$, and if $F(x^{now}) < bestcost$, perform B of Step 1, then return to step 2 and additionally update the history record H.


Begin with a Starting Solution.
Create a Candidate List of Moves.
Evaluate first/next Candidate Move.
Is move the best admissible move so far?
NO
YES
Is the Candidate Tabu?
NO
YES
Does move satisfy Aspiration Criteria?
NO
YES
Could better moves in candidate list exist?
YES
NO
Designate as best Admissible Move
Designate the solution obtained as the new current solution. Record it as the new Best Solution if it improves on the previous best.
Stopping Criteria met?
YES
NO
Update Tabu Restrictions and Aspiration Criteria.
Terminate globally or Transfer
A transfer initiates and intensification or diversification phase embodied in an intermediate or long term memory component

3.1.5 Intensification and Diversification

The following section offers some insights into commonly used strategies within the tabu search.

Intensification Strategies

Intensification strategies are introduced in the search process to perform thorough searches in attractive regions of the solution space. Hence the use of intensification strategies within the tabu search algorithm is to alter decision rules in such a way that they promote the desirable qualities of previous solutions in newly generated solutions. This can be accomplished in both the long and short term of the solution life cycle.

Intensification strategies can be implemented through a number of methods. A common form of an intensification strategy is to keep a sequential list, (of a fixed length), which records a new solution at the end of it, (if it is the best solution found at the time). Then when the strategy is required to be implemented, the current last member of the list is always chosen, (and removed), as the starting point from which to resume the search. In addition to the solution being recorded, associated short term memory of the tabu search at that point is also required to be recorded. Hence when the search is resumed, the first move previously taken from the solution is prohibited, thus a new solution path can be followed.

Diversification Strategies

Diversification strategies are often introduced to prevent searching processes from cycling, hence they drive the search to new regions. This kind of strategy is evident in a number of different algorithms. Within tabu search, diversification is implicitly created to some extent by short term memory structures but is reinforced by longer term memory structures.

Thermodynamic Simulation	Combinatorial Optimization
System States	Feasible Solutions
Energy	Cost
Change of State	Neighboring Solutions
Temperature	Control Parameter
Frozen State	Heuristic Solution

Table 3.1: Thermodynamic Simulation vs. Combinatorial Optimization

3.2 Simulated Annealing

The simulated annealing algorithm has been a widely used method of solving discrete optimization problems since the 1980's. The underlying philosophies governing simulated annealing stem from the physical annealing process. In the field of condensed matter physics, annealing is defined to be the thermal process used to obtain low energy states of a solid in a heat bath. This process consists of two fundamental steps:

1. The temperature of the heat bath is increased to the boiling point of the solid in question.
2. The temperature of the heat bath is carefully decreased, until the particles of the melted solid arrange themselves in the ground state of the solid.

During the first step, the particles of the melted solid arrange themselves randomly. However in step two when the ground state of the solid is achieved, it is characterized by the particles of the sold being arranged in a highly structured lattice, such that the energy of the system is minimal. The structural properties of the cooled solid depend on the rate at which the cooling was implemented. For instance it has been found that large crystals can be formed when a slow cooling phase is implemented, but that imperfections occur when rapid cooling schedules are implemented.

Back in 1953, Metropolis et al [13] presented an algorithm to simulate the system of particles in a temperature of the physical annealing process. The algorithm is known as the Metropolis algorithm. It has been shown by Metropolis et al. that the thermal equilibrium of the system of particles of a temperature can be achieved by the Metropolis algorithm. It was later proposed by Kirkpatrick et al, [14] that this type of simulation could be used to search the feasible solutions of an optimization problem, with the objective being to converge to an optimal solution.

Reeves [12] describes how both Kirkpatrick at al [14] and Cerny [15] have both independently shown how the original Metropolis algorithm can be applied to optimization problems by mapping the elements of the physical cooling processes onto the elements of a combinatorial optimization problem as shown in the following table,

Simulated Annealing for continuous variable problem has also been suggested by a number of authors, see [16] and the references therein.

3.2.1 The Metropolis Procedure

In 1953, Metropolis et al.[13] used the Monte Carlo method now known as the Metropolis algorithm to simulate the collection of particles in thermal equilibrium at a given temperature T. The Metropolis algorithm generates a sequence of states of the system of particles or atoms in the following way. Given a current state or solution, x, of the system of particles with corresponding energy $F(x)$, the system is perturbed to a new state y with energy $F(y)$. If the change, $\Delta F = F(y) - F(x)$, represents a reduction in the energy value then the new state or solution y is accepted. If the change ΔF represents an increase in the energy value, then the new state is accepted with probability $\exp(-(\Delta F/k_B T)$, where T is the surrounding temperature and k_B is the Boltzmann constant. The acceptance rule described above is called the Metropolis criterion and the algorithm that goes with it, is known as the Metropolis algorithm.

In the physical annealing, a thermal equilibrium is reached at each temperature if the lowering of the temperature is done sufficiently slowly. Similarly, in the case of the Metropolis algorithm, a thermal equilibrium can be achieved by generating a large number of transitions at a given temperature. At thermal equilibrium, the probability that the system of particles is in state, x, with energy $F(x)$ is given by the Boltzmann distribution, i.e.,

$$P_T\{X = x\} = \frac{1}{Z(T)} \exp\Big(\frac{-F(x)}{k_B T}\Big), \tag{3.2}$$

where X is a random variable denoting the current state of the system of particles and $Z(T)$ is defined as

$$Z(T) = \sum_y \exp\Big(\frac{-F(y)}{k_B T}\Big). \tag{3.3}$$

3.2.2 The Simulated Annealing (SA) Method

In 1983, Kirkpatrick et al. [14] designed the simulated annealing algorithm for optimization problems by simulating the physical annealing process. The simulation was achievable since the Metropolis algorithm was in place. The formulation of the optimization algorithm using the above analogy consists of a series of Metropolis chains used at different values of decreasing temperatures. In this formulation, the system state corresponds to the feasible solution, the energy of the state corresponds to the objective function to be optimized, and the ground state corresponds to the global minimizer.

The general SA consists of two loops. In the inner loop, a number of points in a Markov chain (a Markov chain is a sequence of trial solutions) in the configuration space is suggested and from which some are accepted. A trial solution is accepted if it only satisfies the Metropolis criterion. On the other hand, in the outer loop, the temperature is progressively decreased

using a cooling schedule which will be discussed in this section. The original SA algorithm was intended for discrete optimization problem. The general description of SA algorithm is as follows:

Simulated annealing algorithms therefore are characterized by a point-wise search through the feasible region. Trial points are generated according to some distribution over the search region and a move to a better point is always accepted. As described previously local searches such as the simple descent method are likely to be confined to local optima. Simulated annealing avoids this by accepting 'worse' solutions subject to probability that decreases as the algorithm progresses. The proceeding condition is used to accept a 'worse' solution, $\hat{x}$, at some temperature T_k, k is the temperature counter. The probability of acceptance is,

$$\exp\left(\frac{F(x) - F(\hat{x})}{T_k}\right). \tag{3.4}$$

The SA Algorithm

The SA algorithm [17] may be expressed as follows:

Step 1 Select an initial solution x.

Step 2 Select an initial temperature $T_0 > 0$. It is important to note that the initial temperature T_0 is usually large, so that most cost increasing trials are accepted and there is little chance of the algorithm being confined to local optima.

Step 3 For k iterations, randomly select $\hat{x} \in N(x)$. Let $\Delta F = F(\hat{x}) - F(x)$, if $\Delta F < 0$ then $x = \hat{x}$. Otherwise let $x = \hat{x}$ with probability $\exp(\frac{-\Delta F}{T_k})$.

Step 4 Set $T_{k+1} = \alpha \times T_k$. If T_k meets the termination criteria, end the algorithm, otherwise set $k = k + 1$ and return to Step 3.

3.2.3 Cooling Schedules

As already mentioned, a cooling schedule[2] refers to the method by which the probability with which the simulated annealing algorithm accepts a solution is decreased [17].

In general a cooling schedule consists of assigning the temperature parameter T an initial value T_0. Once T_0 has been established a method is then required for reducing T as well as for establishing how many trials should be attempted at each value of T. The algorithm will end subject to some defined stopping criterion. Thus the four key components of a cooling schedule are:

- An initial value of the control parameter T,

[2]Also referred to as an annealing schedule in some literature.

- a finite length of each homogeneous Markov chain,
- a decrement function for lowering T,
- a final value of the control parameter, T_k, specified by a stopping criterion.

An Initial Value of the Control Parameter T

If one begins the process of simulated annealing with an initial temperature set so high that almost all solutions are accepted, the result is that a series of random solutions will be produced, all of which could have been a starting solution. Thus it is recommended by most researchers that a 'moderately high' temperature is used to begin with, or that the cooling should be quite rapid during this phase of the algorithm. It is important to note however that if the initial temperature is set too low, very little movement will be possible. Hence the probability of the algorithm being confined to local optima is increased.

An initial temperature value can be obtained by generating a number of trials, say m_0, and requiring that the initial acceptance $\chi_0 = \chi(T_0)$ be close to 1, where $\chi(T)$ is defined as the ratio between the number of accepted transitions and the number of proposed transitions. The expected acceptance ratio χ is approximately given by,

$$\chi = \frac{m_1 + m_2 \times exp(-\overline{\Delta F}^{(+)}/T)}{m_1 + m_2} \tag{3.5}$$

which can be rewritten as:

$$T = \overline{\Delta F}^{(+)} \left(\ln \left(\frac{m_2}{m_2\chi - m_1(1-\chi)} \right) \right)^{-1}, \tag{3.6}$$

where m_1 an m_2 denote the number of trials ($m_1 + m_2 = m_0$) with $\Delta F_{x\hat{x}} \leq 0$ and $\Delta F_{x\hat{x}} > 0$ respectively, and $\Delta\overline{F(x)}^{+}$ the average value of those $\Delta F_{x\hat{x}}$-values, for which $\Delta F_{x\hat{x}} > 0$ ($\Delta F_{x\hat{x}} = F(\hat{x}) - F(x)$).

T_0 is determined as follows: First, T_0 is given some arbitrary value. Next the algorithm is executed for a fixed number of transitions or moves, say m_0, and after each transition, with χ_0 set to 0.95, is used to update the current value of T_0; m_1 and m_2 now corresponds to the number m_1 of cost-decreasing and number m_2 of cost-increasing transitions obtained so far. Numerical experience shows that in this way T_0 reaches a stable value after a small number of transitions. This value is then taken as the initial value of the control parameter.

Markov Chain Length

One is able to fix the length of the Markov chains using a number that is related to the size of the neighborhoods of the problem in question. The length should be large enough to enable

the algorithm to explore the neighborhood of a given point thoroughly. A straightforward selection, therefore, is given by the following relation,

$$L = L_0 \times n, \tag{3.7}$$

where n denotes the dimension of the problem, i.e. the number of machines, and L_0 is a constant called the standard length. Note that the choice of L is problem specific. At each temperature, the Metropolis algorithm attempts to reach equilibrium by means of perturbations and rearrangement. The total number of transitions at the same temperature T that constitutes a homogeneous Markov chain length is given by the parameter L.

Lowering the Control Parameter T

The most common technique used to the control parameter T is,

$$T_{k+1} = \alpha T_k. \tag{3.8}$$

where T_k and T_{k+1} are the temperature at the beginning and at the end of the cooling schedule at temperature change counter k. The 'cooling rate' α, typically has values ranging from 0.8 to 0.99 [5]. It is important to note that the probability of getting trapped in local optima is higher when using a greater decrement rate. On the other hand, the more computation time is spent when the slower decrement rate is used.

An alternative approach to decrease the control parameter T is suggested by Aarts and Korst [17],

$$T_{k+1} = \frac{T_k}{1 + (T_k(1+\delta)/3\sigma_{T_k})} \tag{3.9}$$

where $\sigma(T_k)$ denotes the standard deviation of the values of the objective function values on the Markov chain at T_k. The constant δ controls the cooling rate and is a measure of the desired closeness to equilibrium. Small values ($\delta < 1$) produce slow convergence.

Termination Criteria

A stopping criterion is required to terminate the cooling process. One may choose to stop the algorithm after a fixed number of iterations, or after some number of iterations without an improvement in the objective value. However it is common for the simulated annealing algorithm to end after a specific level of final temperature of the system is reached, for example,

$$T_k \leq \epsilon \tag{3.10}$$

where ϵ is a small number.

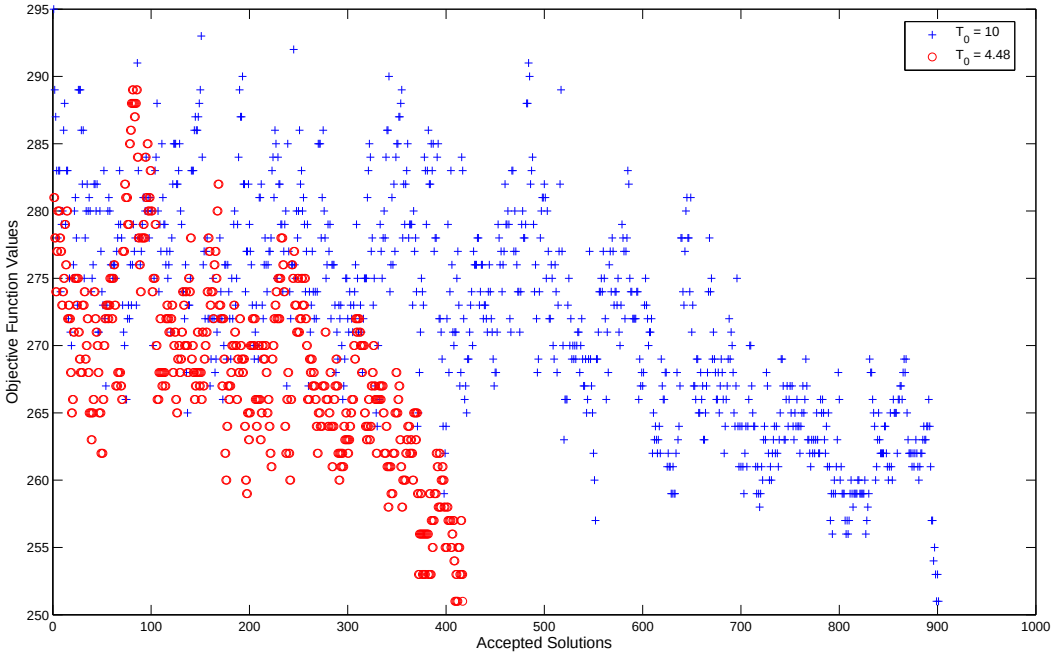


Figure 3.5: Simulated Annealing: High vs. Low T_0

Implementation of SA to Example 1

Let us once again consider the travelling salesman problem 3.1 introduced earlier in the chapter. Figure 3.5 illustrates the solution paths when simulated annealing is applied with a randomly chosen high initial temperature, $T_0 = 10$, versus a derived low initial temperature $T_0 = 4.48$. The x-axis in Figure 3.5 represents the number of accepted solutions whilst the y-axis represents the relative objective function values. The derived temperature was calculated using (3.6) with $m_0 = 40$. In both cases we used the temperature decrement rule (3.8) with $\alpha = 0.95$, the length of the Markov chains of 45 and ended the algorithm when $T < 10^{-3}$.

Figure 3.5 clearly shows the advantage of using a calculated initial temperature as the global optima is obtained in far fewer iterations or function calls.

Chapter 4

Implementation of TS and SA to Brewing Optimization

In this chapter we demonstrate how the scheduling problems within the brewing industry can be optimized using the tabu search algorithm and simulated annealing algorithm. We begin by first considering selected components of the problem, (defining them using the notation introduced in Chapters 1 and 2), and gradually extend the problem through a series of phases as we proceed.

4.1 Phase 1: Filtration and Packaging

We first consider the problem of scheduling the packaging regime of a hypothetical brewery for a week. In order to do this one has to consider that the beer to be packaged must be sourced from the storage and maturation tanks (SVs). Thus the beer must be transferred from the SVs via the filtration plants to the bright beer tanks (BBTs) and then to the various packaging lines. Thus we have 2 fundamental stages ($t = 1, 2$) to consider, the allocation of how the required volumes of beer will be filtered as well as how they are to be packaged.

Let us consider the conceptual layout of the brewery in Figure (4.1). There are 3 filtration plants and 6 packaging lines, hence $m^1 = 3$ and $m^2 = 6$ as well as 3 storage and maturation cellars SV1, SV2 and SV3 consisting of 42, 6 and 14 tanks respectively.

We begin with a description of the problem in this phase. A stock keeping unit (SKU) is a combination of a brand B and a pack type PT. In the problem we will consider, there are 9 brands and 4 different pack types. The brewery is required to schedule a demand for 15 SKU's represented in Table 4.1. Columns 3 and 4 of Table 4.1 represent the brand and pack type respectively of each SKU. The volumes are quoted at standard gravity.

The packaging type, PT, of an SKU describes the type of container the beer is to be packaged in; Table 4.2 represents the packs considered as well as the volumes PTV_j thereof.

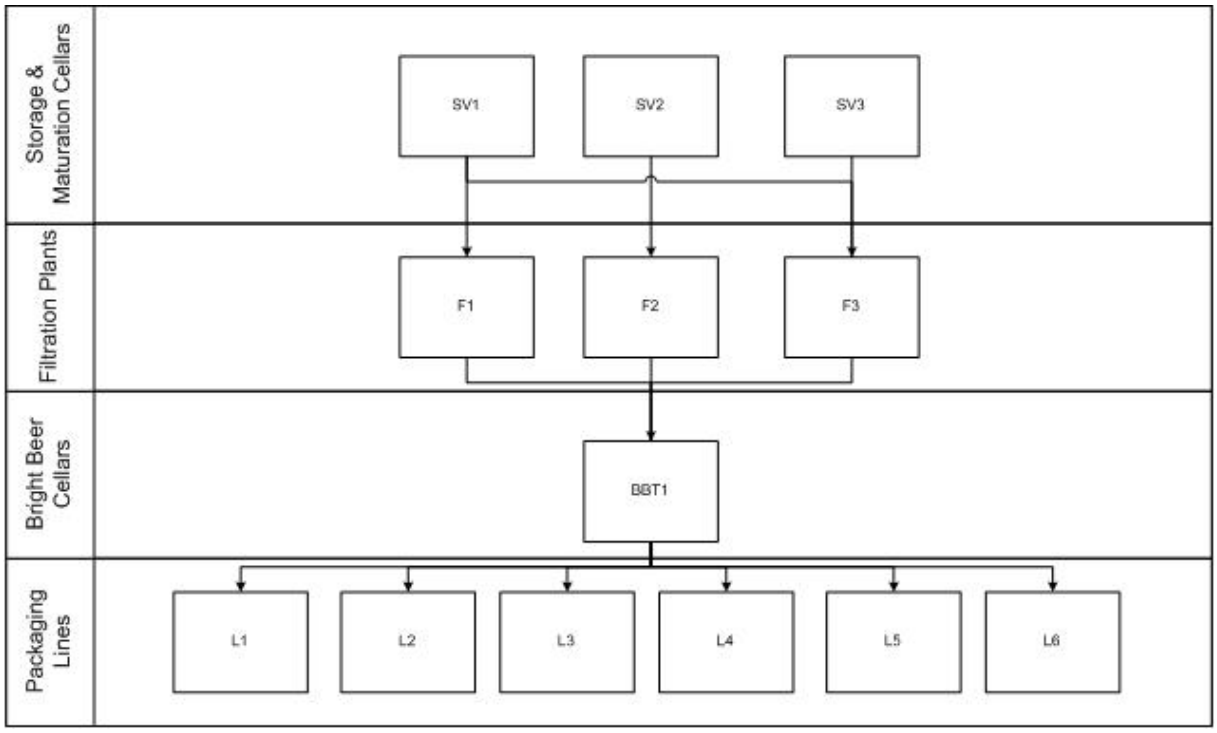


Figure 4.1: Conceptual Layout of a Brewery

#	SKU	Brand	Pack Type	Volume (Hl)
1	11	1	1	180
2	14	1	4	12782
3	23	2	3	3602
4	24	2	4	13574
5	32	3	2	10959
6	44	4	4	865
7	54	5	4	3732
8	64	6	4	2371
9	71	7	1	370
10	73	7	3	3674
11	74	7	4	6802
12	81	8	1	580
13	83	8	3	3616
14	84	8	4	7415
15	92	9	2	1506

Table 4.1: SKU Weekly Demands

Pack Type	Description	PTV_j (l)
1	Keg	30
2	Bottle	0.33
3	Can	0.5
4	Bottle	0.5

Table 4.2: Pack Types

For example SKU 83 is comprised of brand $B = 8$ and pack type $PT = 3$, (a 0.5l bottle). For programming purposes, one can extract the types of brand and packaging type using,

$$B = \frac{SKU - mod(SKU, 10)}{10}, \tag{4.1}$$

$$PT = mod(SKU, 10). \tag{4.2}$$

The attributes of the filtration plants to be considered in the problem are illustrated in Table 4.3. The 'washout volume' in Table 4.3 represents the volume allowed to be filtered after which the filter is required to be cleaned. The relative speed of the filtration plants v_{ij}^1 is independent of job j and we assume that when they are in operation they are 100% efficient. Hence we have, $v_{ij}^1 = 550$, 600 and 300 for $i = 1, 2$ and 3 respectively. We assume that the only brand that filters 1 and 2 cannot process is brand 9, and that brand 9 is exclusively processed on filter 3.

Filter	**1**	**2**	**3**
Capacity (Hl/hr)	550	600	300
Washout Vol (Hl)	2500	2500	1500

Table 4.3: Filtration Plant Attributes

The capacities and efficiencies, (E_i), of each packaging line is illustrated in Table 4.4. Efficiency is given as a percentage.

Line	**1**	**2**	**3**	**4**	**5**	**6**
Capacity ($bphr_i$)	24000	50000	23000	60000	70000	60000
$E_i(\%)$	75	85	90	85	85	90

Table 4.4: Packaging Line Attributes

Note that the capacities of the packaging lines are quoted in the number of bottles per hour (bphr). In this case there are 6 machines, $i = 1, \ldots, 6$. Thus in order to obtain their capacities in Hl/hr, taking into consideration efficiencies as well as the SKU in question we use:

$$v_{ij}^2 = \frac{bphr_i \times PTV_j \times \frac{E_i}{100}}{100}, i = 1, \ldots, 6; \forall j = 1 \ldots n, \tag{4.3}$$

where $bphr_i$ is the number of bottles per hour packaged by machine i and PTV_j is the volume of the pack type of job j; the denominator of a 100 is used to obtain a relative speed of Hl/hr.

The sequence dependent setup times for each SKU take both the brand and pack type of a job into consideration. Matrix Q represents brand dependent setup times, both the row and column indices represent the 9 different brands. Matrix R represents the pack dependent setup times used in the problem, both the row and column indices represent the 4 different packaging types. The matrices are given by,

$$Q_{j_B l_B} = \begin{pmatrix} 0 & 1.3 & 3 & 1.3 & 1.3 & 1.3 & 1.3 & 1.3 & 3 \\ & 0 & 3 & 1.3 & 1.3 & 1.3 & 1.3 & 1.3 & 3 \\ & & 0 & 3 & 3 & 3 & 3 & 3 & 3 \\ & & & 0 & 1.3 & 1.3 & 1.3 & 1.3 & 3 \\ & & & & 0 & 1.3 & 1.3 & 1.3 & 3 \\ & & & & & 0 & 1.15 & 1.15 & 3 \\ & & & & & & 0 & 1.15 & 3 \\ & & & & & & & 0 & 3 \\ & & & & & & & & 0 \end{pmatrix},$$

and

$$R_{j_{PT}l_{PT}} = \begin{pmatrix} 0 & 1000 & 1000 & 1000 \\ & 0 & 1000 & 1.15 \\ & & 0 & 1000 \\ & & & 0 \end{pmatrix}.$$

Hence if we were to consider the occurrence of two consecutive jobs j and l of SKU 32, (brand = 3, pack =2), and SKU 44, (brand = 4, pack =4), respectively; the sequence dependent setup time s^t_{jl} at $t = 2$ required would be,

$$\begin{aligned} s^2_{jl} &= Q_{j_B l_B} + R_{j_{PT} l_{PT}} && (4.4) \\ s^2_{jl} &= Q_{34} + R_{24} && (4.5) \\ s^2_{jl} &= 3 + 1.15 && (4.6) \\ s^2_{jl} &= 4.15hrs && (4.7) \end{aligned}$$

We assume that in this problem the setup time of job j if job j is assigned to the first position[1] on machine i at stage t,

$$ch^t_{ij} = 0 \quad \forall t = 1, \ldots, k \quad and \quad \forall i = 1, \ldots, m^t. \tag{4.8}$$

However, the following interruptions will occur:

- If a packaging line or filter does not process a job within 24hrs, a $3hr$ cleaning process will be completed before any new job is processed on the relevant piece of equipment.
- If a packaging line runs for more than $24hrs$ or a filtration plant filters the equivalent of it's washout volume, the respective piece of equipment will be stopped and a cleaning process will begin. In the case of a packaging line $3hrs$ of cleaning will be allocated whilst $0.2hrs$ of cleaning will be allocated to the filters. The current job on the given machine will be stopped, and the remainder of the job will only commence after the cleaning process has been completed.

The packaging demands in Table 4.1 are sourced from the SV tanks. Although there are 62 SV tanks, in our problem there are 42 tanks within the SV's from which we can supply the total demand. Thus there are 42 jobs that are required to be scheduled, i.e. $j = 1, \ldots, 42$.

In Table 4.5 we present all 42 jobs and their attributes:

Column 1 - j: The job number,$j = 1, \ldots, 42$.

Column 2 - SKU_j: The type of SKU that the job will be packaged as.

[1]By this we mean that job j is the first job to be processed by machine i.

Column 3 - $Cellar_j$**:** The SV cellar from which the job will be sourced.

Column 4 - $Tank_j$**:** The SV tank within the relevant cellar from which the job will be sourced.

Column 5 - ps_j^t**:** The volumes of jobs $j = 1, \ldots, 42$ are quoted at standard gravity.

Column 6 - r_j**:** The release dates r_j indicate earliest time, in hours, at which a job may begin to be processed.

Column 7 - Packaging Line: The initial assignment of the packaging line to process the job.

Column 8 - Filter: The initial assignment of the filter to process the job.

If one adds the volumes, ps_j^t, of all tanks that pertain to a particular SKU, this will equal the corresponding SKU's packaging volume in Table 4.1.

j	SKU_j	$Cellar_j$	$Tank_j$	ps_j^t	r_j	**Packaging Line**	**Filter**
1	14	1	1	1000	0	5	1
2	14	1	2	781	0	5	2
3	14	1	3	1000	41	5	1
4	14	1	4	1000	41	5	2
5	14	1	5	1000	41	5	1
6	14	1	6	1000	41	5	2
7	14	1	7	1000	41	5	1
8	14	1	8	995	41	5	2
9	14	1	9	1000	65	5	1
10	14	1	10	1007	89	5	2
11	11	2	1	180	0	7	1
12	14	2	2	2999	17	5	2
13	24	1	11	594	0	5	1
14	24	1	12	690	0	5	2
15	24	1	13	690	0	5	1
16	23	1	14	685	0	4	2
17	23	2	3	2917	0	4	1
18	24	3	1	3919	0	5	2
19	24	3	2	3760	0	5	1
20	24	3	3	3921	17	5	2
21	32	1	15	993	0	6	1
22	32	1	16	1000	0	6	2
23	32	1	17	1000	0	6	1
24	32	2	4	1966	0	6	2
25	32	2	5	3000	17	6	1
26	32	2	6	3000	41	6	2
27	44	1	18	865	89	5	1
28	54	3	4	2641	81.8	5	2
29	54	3	5	1091	89	5	1
30	64	1	19	657	0	5	2
31	64	1	20	861	17	5	1
32	64	1	21	853	17	5	2
33	71	3	6	370	0	7	1
34	74	3	7	3902	0	3	2
35	74	3	8	2900	17	3	1
36	73	3	9	3674	0	4	2
37	83	3	10	3616	17	4	1
38	81	3	11	580	41	7	2
39	84	3	12	4015	65	3	1
40	84	3	13	3400	65	3	2
41	92	1	22	652	17	1	3
42	92	1	23	854	17	1	3

Table 4.5: Initial assignment of the Packaging line and Filtration Jobs

4.1.1 Scenario 1

In this scenario we assume that the initial assignment of the packaging lines to each job, as represented in Table 4.5, is fixed. Hence the total processing time will consist of both the filtration and packaging process, but only moves with regard to the sequence of the jobs and the allocation of the filters will be executed. Thus the assignment of the packaging lines remain as given in Table 4.5. For the purposes of this exercise, we will ignore any physical constraints associated to the BBTs and assume that the BBT tanks are emptied at the rate of the packaging line it feeds. We will however ensure that after any given filtration job, a constant number of hours, i.e. $4hrs$, are added to the completion time of the job before it can be packaged. In this way we ensure that the amount of time that the job would have spent in the BBT's is accounted for. The 'First In First Out' (FIFO) philosophy is also followed where applicable.

We assume that the filtration plants $i = 1, \ldots, m^1$, as well as the packaging lines, $i = 1, \ldots, m^2$ are available immediately, i.e. $a_i^t = 0, \forall t, i$ where a_i^t represents the time at which machine i at stage t is available. We also assume that all jobs are required to complete within 1 week (168 hours). Our objective is to minimize the sum of the completion times of the last job on each packaging line. Therefore the objective function to be minimized is:

$$\lambda C_{TOTAL} + (1 - \lambda)\eta_{Ta}, \tag{4.9}$$

where λ is given.

In Scenario 1 as well as proceeding scenarios[2], we restrict the number of jobs that may be allocated to each filter as follows:

- Filter 1: 20 jobs may be allocated.
- Filter 2: 20 jobs may be allocated.
- Filter 3: 2 jobs may be allocated.

It is important to note that only the number of jobs are restricted and no restrictions are placed on the filters in regard to the brand of a particular job, nor the volume of the job. These restrictions were imposed for realism.

Results for TS on Scenario 1

In the implementation of TS to Scenario 1, a diversification strategy was implemented to prevent cycling by using a long term memory structure which kept record of accepted objective function values from the beginning of the algorithm. After a given number of iterations, eg. 10, the last 5 accepted objective function values are examined, if the standard deviation of

[2]Scenarios 2 and 3

these values was below 0.1 a random member of the candidate list is selected as the next best solution and the search is resumed. A number of tests were conducted to see the effect on the objective function value as a result of,

- the number of iterations performed,
- the length of tenure (Tenure) imposed,
- the number of iterations performed before the diversification strategy (D.S.) was implemented.

The results of the tests are summarized in Tables 5.1, 5.2 and 5.3 respectively in Appendix A.

The tests revealed that the best solutions were found when a relatively small tenure of 5 was used and when the diversification strategy was implemented after 15 iterations. As expected, an increased number of iterations yielded better results. The best solution found was that of 591.0355, see Table 5.3, which was found after 131 iterations, a tenure of 5 and commencing the diversification strategy after 15 iterations, the solution path is represented in Figure 4.2. Hence the number of iterations performed does not ensure optimality but rather increases the likelihood thereof.

In Figure 4.2 the x-axis represents the number of iterations and the y-axis represents the value of the objective function. One can clearly see in Figure 4.2 where the diversification strategy is utilized, most noticeably after iteration 90. The large spike in the solution path indicates where a random solution from the candidate list was accepted. After iteration 90, one can see how the values of the objective function consistently decrease to the global or near global optima.

Results for SA on Scenario 1

A number of tests were conducted to see the effect on the objective function value as a result of implementing different cooling schedules. In particular we have used,

- $m_0 = m_1 + m_2 = 200$ and $m_0 = m_1 + m_2 = 270$
- $\alpha = 0.95$ and $\alpha = 0.85$

The best results for scenario 1 were obtained using a cooling schedule comprised of:

- An initial value of the control parameter $T = 26.7983$ (using equation (3.6), $m_0 = m_1 + m_2 = 270$).
- Markov chains with a length of $L = 550$,
- A decrement function, $T_{t+1} = \alpha T, \alpha = 0.95$,

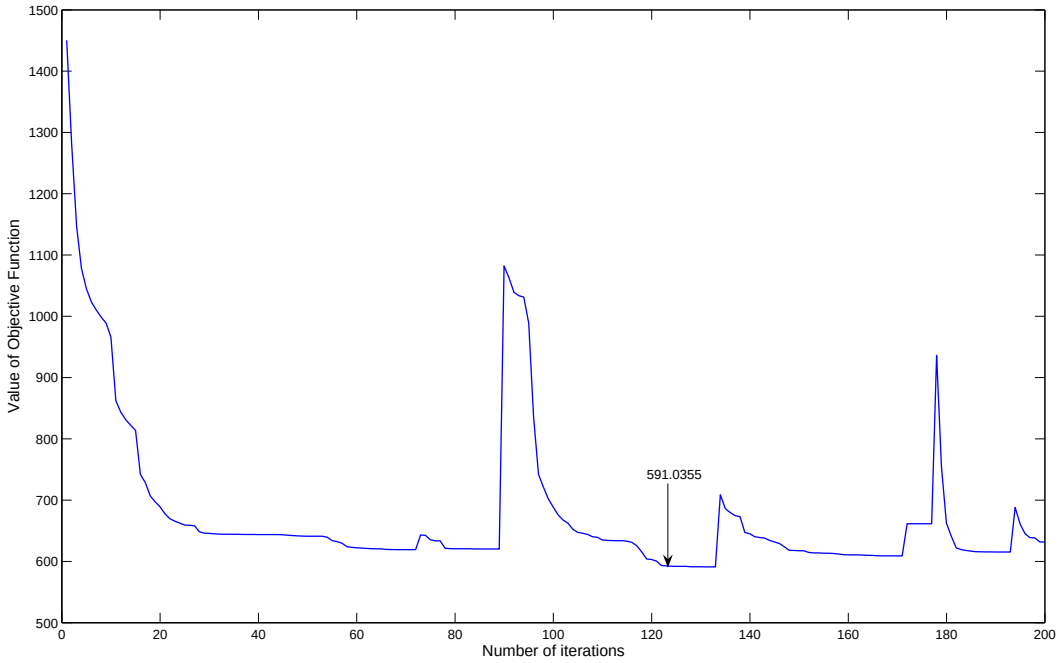


Figure 4.2: TS Solution Path for a single case of Scenario 1

- A stopping criterion such that the method ends when $T \leq 0.001$.

The algorithm yielded an optimal solution of 587.3726. The solution path is illustrated in Figure 4.3, where the x-axis represents the number of accepted solutions found and the y-axis represents the value of the objective function.

The best results obtained by SA and TS show that SA is the best performer in scenario 1. We would like to mention that the results obtained by the other SA runs were not inferior to the best results of TS. It is therefore clear that SA is the best performer in the problem described by scenario 1.

A comparison of the number of function evaluations corresponding to the best run of SA and TS confirms the superiority of SA over TS in scenario 1. TS made approximately 170000 function calls as oppose to SA which made approximately 108000.

To visualize the features of the optimized solution obtained by SA, we present two more figures. Figure 4.4 illustrates by means of gannt charts, the events occurring on filters 1 and 2 for both the optimized and initial solutions. The events are comprised of a combination of the processing of jobs and cleaning processes and brand changes. Thus the completion of a particular job may be described by two events, i.e. if job j was interrupted due to a cleaning

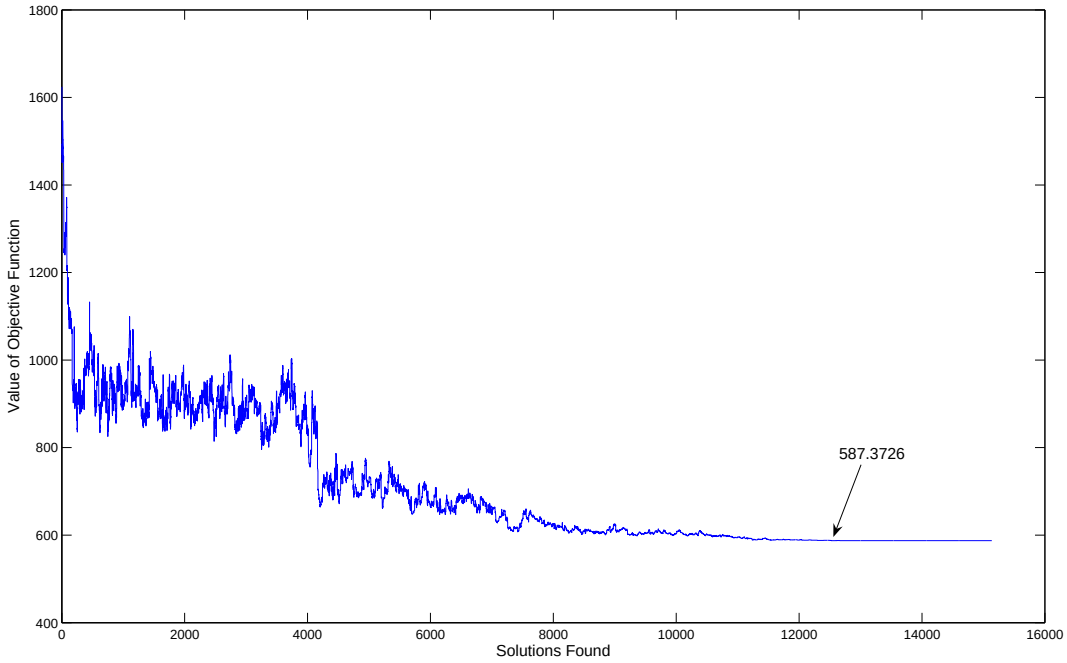


Figure 4.3: SA Solution Path for a single case of Scenario 1

procedure, then it would be described by an event representing the initial volume filtered, followed by the cleaning event, and finally an event representing the remainder of job j to be filtered. The first sub-figures (in the left hand side) in Figure 4.4 represent the initial solution's allocation of jobs to Filter 1 and Filter 2 respectively; whilst the second sub-figures (in the right hand side) represent the optimized solution's allocation of jobs to the filters. The x-axis in these figures represent time in hours, whist the y-axis represents events.

It is evident that in the optimized solution the lines are more efficiently used as there is less idle time between jobs.

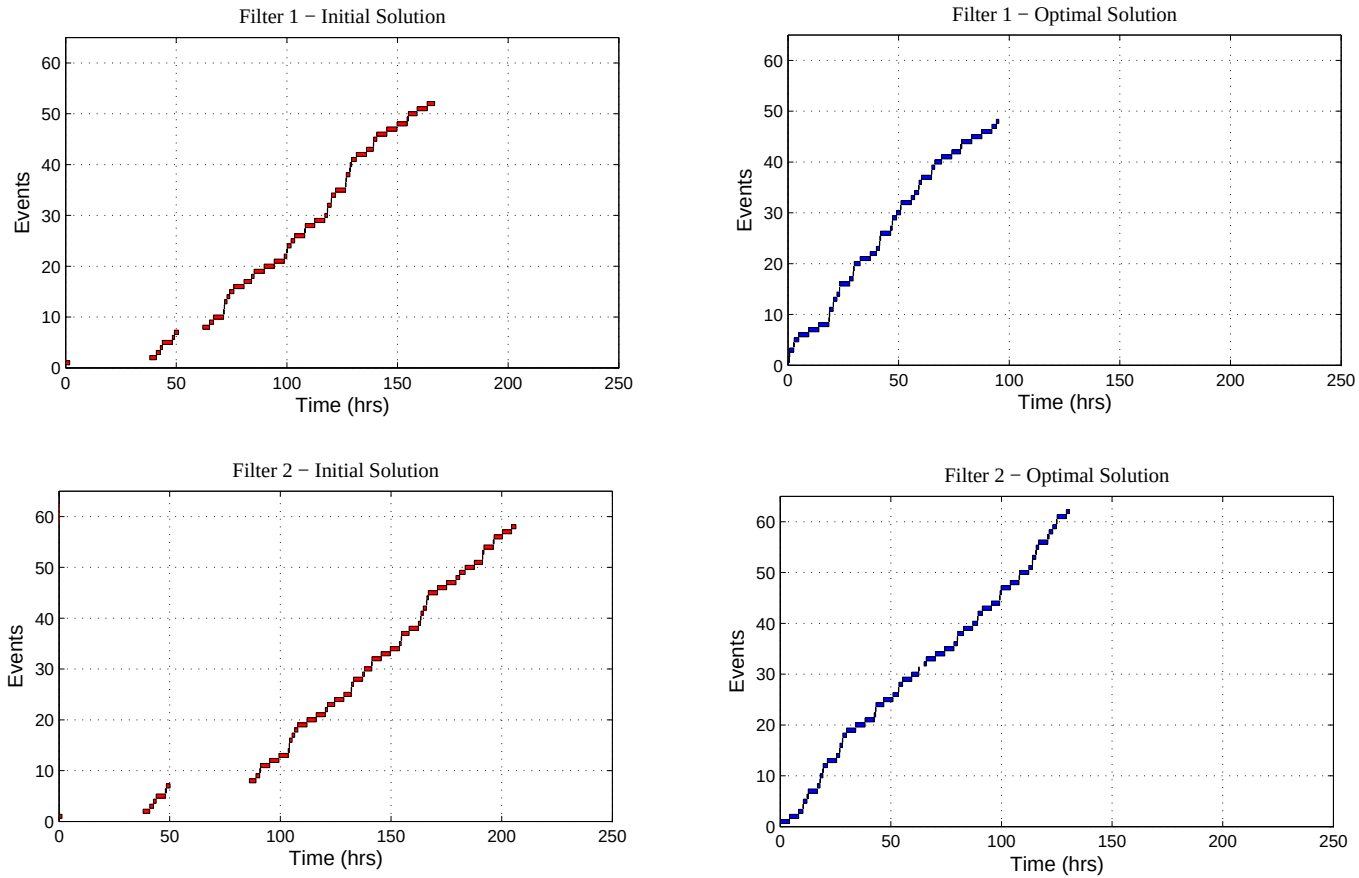


Figure 4.4: Initial and Optimized allocation of jobs for Filter 1 and Filter 2 in Scenario 1

4.1.2 Scenario 2

Scenario 2 is an extension of Scenario 1. The difference being that in Scenario 2 moves will be executed with regard to the sequence of the jobs and the allocation of the filters as well as the as the assignment of the jobs to the packaging lines. For this problem, a number of SKU's may be packaged on either packaging line 2 or 4 as is illustrated (in bold) in the 'Line Restrictions' matrix $LR(SKU, PL)$, where the rows represent the various SKU (see Table 4.1), $SKU = 1, \ldots, 15$ and the columns represent the packaging lines, $PL = 1, \ldots, 6$. From the matrix LR we can see that only lines 2 and 4 share SKU's, namely 14, 24, 54, 64, 74 and 84, that can be allocated to either line. The matrix $LR(SKU, PL)$ is given by:

$$LR(SKU, PL) = \begin{pmatrix} 0 & \mathbf{0} & 0 & \mathbf{0} & 0 & 1 \\ 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{0} & 1 & \mathbf{0} & 0 & 0 \\ 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{0} & 0 & \mathbf{0} & 1 & 0 \\ 0 & \mathbf{0} & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{0} & 0 & \mathbf{0} & 0 & 1 \\ 0 & \mathbf{0} & 1 & \mathbf{0} & 0 & 0 \\ 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{0} & 0 & \mathbf{0} & 0 & 1 \\ 0 & \mathbf{0} & 1 & \mathbf{0} & 0 & 0 \\ 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & 0 \\ 1 & \mathbf{0} & 0 & \mathbf{0} & 0 & 0 \end{pmatrix}$$

In the previous problem presented in scenario 1, we have conducted a series of runs for SA and TS where good parameter values were obtained. We have used these parameter values for the problem presented in scenario 2.

TS was implemented with a tenure of 5 and the diversification strategy was implemented after 15 iterations. The best result obtained by TS was 584.6004 after 200 iterations (170000 function calls)

The SA algorithm was therefore implemented with a cooling schedule comprised of:

- An initial value of the control parameter $T = 26.7983$ (using equation (3.6), $m_0 = m_1 + m_2 = 270$).
- Markov chains with a length of $L = 550$,
- A decrement function, $T_{t+1} = \alpha T, \alpha = 0.95$,
- A stopping criterion such that the method ends when $T \leq 0.001$.

SA performed 108000 function calls, with a best result obtained of 572.6229.

Table 4.6 shows the % allocation of the total volume of each SKU to lines 2 and 4 in the initial solution, and that of the result of each algorithm.

SKU	**Initial Solution**		**TS**		**SA**	
	Line 2	Line 4	Line 2	Line 4	Line 2	Line 4
14	53%	47%	71%	29%	54%	46%
24	32%	68%	55%	45%	51%	49%
54	27%	73%	0%	100%	54%	46%
64	0%	100%	53%	47%	0%	100%
74	58%	42%	43%	57%	22%	78%
84	70%	30%	0%	100%	50%	50%

Table 4.6: % of SKU allocated to Line 2 and Line 4

In Table 4.6 one can clearly see that the implementation of SA has resulted in a more even distribution of the volumes of the SKU's than TS.

4.1.3 Scenario 3

In Scenario 3, we extend Scenario 2 by including the planning of the use of Fermentation Vessels (FV) to the problem. Figure 4.5 illustrates the additional connectivity constraints from FV to SV in our conceptual brewery. In Figure 4.5 one can clearly see connectivity constraints between the FV's and the SV's. For example FV4 may only transfer beer to SV3.

We are not looking explicitly at inter tank transfers between SV and FV, but rather at the amount of aging that transpires as a result of a filtration/packaging solution. Hence we take into consideration the amount of time,$((y' - y)$ where y' represents a point in time after y), volumes of beer in FV's age due to insufficient capacity in the SV's. The actual volume aged,$FV_y - SV_y$, is also taken into consideration. The objective function is thus redefined as,

$$\lambda(C_{TOTAL} + A) + (1 - \lambda)\eta_{Ta}. \tag{4.10}$$

where,

$$A = \sum_{y=0}^{z}((FV_y - SV_y) \times (y' - y)) \tag{4.11}$$

and,

- z is the last point in time where the contents of a fermentation vessel is ready to be transferred and an storage and maturation vessel's capacity is available.
- SV_y is the sum of the capacity volume of the storage and maturation vessels available at time y,

- FV_y is the sum of the volume of the contents of a fermentation vessels available at time y.
- y only represents points in time where $SV_y \leq FV_y$ and aging occurs.
- y' is the next point in time after y, $y' > y$.

The FV data added to the model consisted of 72 tanks, Table 4.7 presents 4 of these tanks as an example.

Brand	**Tank Number**	**Volume**	**Cellar**	**Ready (hr)**
6	123	459	3	113
6	124	457	3	113
1	70	979	2	17
1	66	2746	2	65

Table 4.7: FV Data

The initial solution had a total packaging line runtime of 908 hours and a aging component of 6494700 litre hours. After TS was implemented[3] this was reduced to a packaging line runtime of 541 hours and an aging component of 664571 litre hours. The use of SA[4] resulted in a total packaging line runtime of 542 hours and a aging component of 635382 litre hours. SA was implemented with a cooling schedule comprised of,

- An initial value of the control parameter $T = 2134.75$ (using equation (3.6), $m_1 + m_2 = 270$).
- Markov chains with a length of 550,
- A decrement function, $T_{t+1} = \alpha T, \alpha = 0.95$,
- A stopping criterion such that the method ends when $T \leq 0.001$.

Despite the TS solution's packaging line runtime concluding an hour earlier than that of the SA solution, the aging that results from the SA solution is significantly less than that of the TS solution. One must note however that SA performed significantly more function calls than TS.

4.1.4 Scenario 4

In Scenario 4, we extend Scenario 3 by ignoring any restrictions imposed on the number of jobs that any given filter may process.

[3]TS performed 170000 function calls.
[4]SA performed 180000 function calls.

Both SA and TS were used to optimize scenario 4, the allocation of jobs as a percentage of the total number of jobs, as well as the volume as a percentage of the total volume to the filters are described in Table 4.8. SA was implemented on this scenario using a cooling

Filter	**SA**		**TS**	
	% of Jobs	% of Volume	% of Jobs	% of Volume
1	38	38	40	41
2	33	39	31	36
3	29	23	29	23

Table 4.8: Summary of Results

schedule composed of:

- An initial value of the control parameter $T = 1738200$ (using equation (3.6), $m_1 + m_2 = 270$).
- Markov chains with a length of 550,
- A decrement function, $T_{t+1} = \alpha T, \alpha = 0.95$,
- A stopping criterion such that the method ends when $T \leq 0.001$.

SA performed approximately 227700 function calls resulting in a total packaging line runtime of 520 hours and a aging component of 635382 litre hours.

TS performed approximately 2125000 function calls resulting in a packaging line runtime of 718.7283 hours and a aging component of 642071 litre hours.

Even though TS has a better pack time than SA, 518.7288 vs 520, the aging component of SA is smaller i.e. 635382 vs 642071. Hence SA performed better than TS.

Table 4.9 summarizes the packaging line run times obtained for each scenario after the implementation of TS and SA.

Scenario	TS	SA
1	591	587
2	585	573
3	541	542
4	518	520

Table 4.9: Results Summary of Scenarios 1 to 4

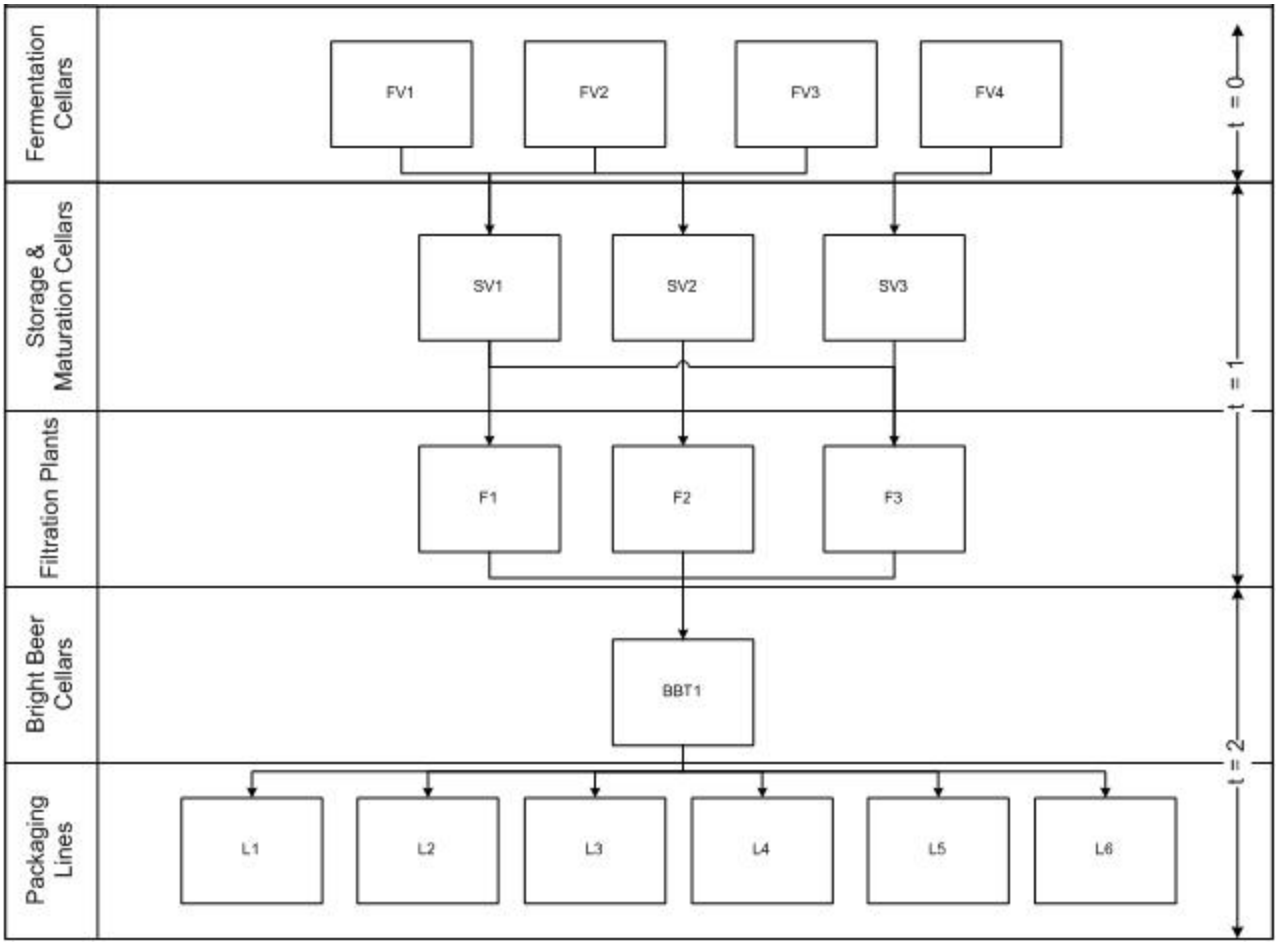


Figure 4.5: Conceptual Layout of a Brewery

Chapter 5

Appendix A

Tenure	D.S.	Obj. Value
5	5	619.0911
5	10	619.0911
5	15	611.3076
5	30	613.3915
5	45	600.7108
10	5	629.7863
10	10	721.1669
10	15	721.1669
10	30	721.1669
10	45	719.6529
20	5	741.1127
20	10	741.1127
20	15	741.1127
20	30	741.1127
20	45	741.1127
30	5	748.9169
30	10	748.024
30	15	751.9289
30	30	748.7021
30	45	751.9289
40	5	746.5802
40	10	751.9289
40	15	748.0732
40	30	751.3171
40	45	751.0732

Table 5.1: Scenario 1:Tabu Search Results - 100 Iterations

Tenure	**D.S.**	**Obj. Value**
5	5	610.5905
5	10	615.1478
5	15	602.4470
5	30	605.7431
5	45	619.0911
10	5	624.9347
10	10	651.2482
10	15	612.4082
10	30	612.1908
10	45	627.1935
20	5	741.1127
20	10	741.1127
20	15	741.1127
20	30	740.8136
20	45	741.1127
30	5	748.0732
30	10	751.9289
30	15	747.9086
30	30	751.9289
30	45	750.4289
40	5	748.3495
40	10	751.9289
40	15	751.9289
40	30	746.2875
40	45	748.3495

Table 5.2: Scenario 1:Tabu Search Results - 150 Iterations

Tenure	**D.S.**	**Obj. Value**
5	5	614.9624
5	10	611.1269
5	15	591.0355
5	30	612.6734
5	45	610.5031
10	5	618.8648
10	10	602.5910
10	15	631.5604
10	30	640.0287
10	45	613.1576
20	5	741.1127
20	10	741.1127
20	15	740.4327
20	30	741.1127
20	45	741.1127
30	5	751.9289
30	10	748.3495
30	15	748.0732
30	30	751.9289
30	45	751.9289
40	5	751.9289
40	10	751.9289
40	15	750.5732
40	30	749.0732
40	45	751.7905

Table 5.3: Scenario 1:Tabu Search Results - 200 Iterations

Bibliography

[1] Hans-Otto Gnther H. and van Beek P. (Editors), Advanced Planning and Scheduling Solutions in Process Industry, Springer, ISBN-10: 3540002227, 2003.

[2] Pringle A., 'Industrial Engineer with 6 years experience in the brewing industry', personal communication, 2006.

[3] http://www.lifl.fr/t̃albi/challenge/docs/SCHEDULING-GRID-COMPUTING-TESTCASE.pdf, An Introduction of the Flowshop Scheduling Problem, 2007.

[4] Jungwattanakit, J, Reodecha, M, Chaovalitwongse, P. and Werner, F., 'Algorithms for flexible flow shop problems with unrelated paralelle machines, setup times, and dual criteria', Internal Report, Department of Industrial Engineering, Chulaongkorn University, Thailand, 2006.

[5] Pardalos P.M. and Resende G.C.M.,'Handbook of Applied Optimization', Oxford University Press, 2002.

[6] Hansman,K. and Hoeck,M, 'Production Control of a Flexible Manufacturing System in a Job Shop Enviroment', Internal Report, Department of Industrial Engineering, University of Hamburg, Germany, 2006.

[7] Jones DF, Mirrazavi SK and Tamiz M, 'Multi Objective meta-heuristics: An overview of the current state-of-art', European Journal of Operations Research, 137(1):1-9, 2002.

[8] Glover F. and Laguna M.,'Tabu Search', Kluwer Academic Publishers, 1997.

[9] Nowicki E., Smutnicki C., 'Flexible flow shop scheduling', Proc. of International Conference on Industrial Engineering and Production Management, Book II, Lyon, France, 85-94, 1997.

[10] Cheloueh, R and Siarry, P, 'Tabu Search Applied to Global Optimization', European Journal of Operations Research, 123(2):256-270, 2000.

[11] Glover F., 'Tabu Search: A Tutorial', Interfaces, 20(4):74-94,1990.

[12] Reeves, C., 'Modern Heuristic Techniques for Combinatorial Problems' MaGraw-Hill Book Company Europe, 1995.

[13] Metropolis N.; Rosenbluth A.W, Rosenbluth M.N, Teller A.H. and Teller E, 'Equation of state calculation by fast computing machines', Journal of Chemical Physics, 21,1087-1091, 1953.

[14] Kirkpatrick S, Gellat C.D. and Vecchi M.P., 'Optimization by simulated annealing', Science, 220,671-680,1983.

[15] Cerny V., 'A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm', Journal of Optimization Theory and Applications, 45:41-55, 1985.

[16] Dekkers A. and Aarts E.,'Global Optimization Simulated Annealing', Mathematical Programming, 50:367-393, 1991.

[17] Aarts E. and Korst J., Simulated Annealing and Boltzmann Machine, John Wiley & Sons, 1989.