**國 立 中 央 大 學**

**工 業 管 理 研 究 所**

**碩 士 論 文**

以基因演算法規劃啤酒釀造

Applying Genetic Algorithm to Schedule Brewery Production

研 究 生：Bui Xuan Toan

指導教授：王啟泰 博士

中 華 民 國 一 ○ 六 年 六 月

Acknowledgement

I would like to say thanks to my parents, they are always great persons to me. I would not event have a chance to study in Taiwan if it is not because of them. My parents’ supports which give me precious time to study and completed this thesis.

In the process of completing this thesis, I have received many good advices from my advisor professor. To me Professor Chi-Tai Wang is the best teacher I have ever had. Whenever I got stuck in a problem arising from the thesis, his advices always give me some clues to create the solution for that problem. Moreover, his professional attitude and great patience have great impacted on me. My advisor professor is just like my role model in life.

Like any other master students in the Institute of Industrial Management, I have to finish the thesis with my own thought and creativity. The process of writing the thesis is a great journey to me, I have a chance to not only review a ton of knowledge but also to challenge and explore myself. On that journey, professor Wang’s advice is like the compass which gives me the right direction to go. It is a great luck to me to have a chance to work with professor Chi-Tai Wang through this thesis.

I also would like to say thank to professor Ying-Jiun Hsieh and professor Tsai, Chi-Yang Who give me many useful advices about how to further improve my thesis. Beside the advices of professors, I also feel very gratitude about the patient and time of Sabeco Corporation employees who had spent their valuable time to help me understand their production system.

Last, but certainly not least, I would like to say thank to all of my lab-mates. They have created a friendly and productive environment which I think is a very important element. I have also received help when it comes to language difficulty. It is a pleasure to work in professor Wang’s lab.

**中文摘要**

排程規劃在製造業有舉足輕重的地位，本研究就是站在這角度上進行研究的延伸。由於排程規劃是指數型成長的複雜性問題，在限制求解時間的條件下，此劃問題可以被歸類為Np-hard的問題。因為上述原因，目前在規劃排程的應用上，不是站在求得最佳解的角度去尋找方向，而是使用啟發式的演算法來解決排程上的研究議題。

目前有許多學者研究如何使用不同地啟發式演算法來解決排程議題，如區域搜尋、禁忌搜尋和基因演算法等方法。根據近幾年的學術研究趨勢，基因演算法被認定為一個有效率的解決排程議題的演算法，也由於基因演算法逐漸地被重視下，其所討論的研究領域與應用類型更為全面。

在啤酒製造工業，作物的發酵過程是整個釀造過程中的關鍵。此過程決定了啤酒的品質、配方、風味以及生產線上的流程規劃。原料發酵的過程佔據了整個生產流程約41天的時間，而每一種不同類型的啤酒也有其發酵需求時間和限制，有效地規劃發酵流程對於啤酒釀造是非常重要的。本研究提出如何使用基因演算法，進行排程規劃求解最適宜的生產排程方式，期望藉由有效地生產計劃與分析工具，利用機台的排程規劃搭配，減少生產排程中，可能發生的瓶頸時間以及準備時間。

**關鍵字：**排程規劃、批量、啤酒工業、兩階段生產法、基因演算法。

Abstract

Scheduling is one of the most important problems in any manufacturing industry. Therefore, the problem has been studied extendedly. Since, the scheduling problem is classified as NP-hard problem, which means the time required for finding the optimal solution of the problem is grown exponentially with the size of the problem. Therefore, it is unrealistic to find optimal solution for the scheduling problem in the scene of the real world industrial case, even with today advanced computer system.

There are many heuristic algorithms have been proposal to solve the scheduling problem. They are beam search, local search technique, tabular search and Genetic Algorithm (GA), to name a few. In recent years, GA has become a noticeable candidate for solving the scheduling problem effectively. The idea of mimicking the evolutionary process is very interesting to researchers. And the recent advanced in heuristic GA has sparked more attention toward new research and application in the field of GA.

In Brewery industry, the fermentation process is the most crucial components of the whole manufacturing process. It will decide the quality, taste of the products as well as the productivity of the production line. Since, the fermentation time can take up to 41 days, and the requirement time is varying a lot between different types of beers, therefore finding a good scheduling solution to dealing with this complexity is crucial for beer manufacturers. This research will propose a GA to solve the scheduling problem in beer production. The proposed methodology will serve as a planning and analysis tool to utilize assets (tanks, filling lines) effectively, reduce congestion and synchronize the production process between the two production stages (liquid preparation and bottling).

**Keywords:** scheduling, lot sizing, brewery industry, two-stage production, GA.

Table of Contents

[Acknowlegement i](#_Toc487235984)

[Abstract ii](#_Toc487235985)

[Table of Contents iii](#_Toc487235986)

[Table of Figures v](#_Toc487235987)

[List of Tables vii](#_Toc487235988)

[CHAPTER 1. Introduction 1](#_Toc487235989)

[1.1 Motivations 1](#_Toc487235990)

[1.2 Objectives 2](#_Toc487235991)

[1.3 Research Framework 3](#_Toc487235992)

[CHAPTER 2. Literature Review 4](#_Toc487235993)

[2.1 Scheduling Problems in Industrial Management 4](#_Toc487235994)

[2.1.1 The Job-Shop Scheduling Problem 5](#_Toc487235995)

[2.1.2 The Flexible Job-Shop Scheduling Problem. 7](#_Toc487235996)

[2.1.3 The Integrated Operation Sequence and Resource Selection Problem: 8](#_Toc487235997)

[2.1.4 The Scheduling Problem in Soft Drink and Brewery Industry. 10](#_Toc487235998)

[2.2 GA Approach 13](#_Toc487235999)

[2.2.1 GA in General 14](#_Toc487236000)

[2.2.2 GA and Network Modeling 16](#_Toc487236001)

[CHAPTER 3. Proposed Methodology 22](#_Toc487236002)

[3.1 Beer Production Process 22](#_Toc487236003)

[3.2 Problem Description and Modeling 23](#_Toc487236004)

[3.3 Data Collection and Input Design 27](#_Toc487236005)

[CHAPTER 4. Solution Approach and Sabeco Case Study 31](#_Toc487236006)

[4.1 Tank Types Assignment Result and Introduction to Synchronization between Tank Types Assignment and Filling Stages 31](#_Toc487236007)

[4.2 Data Preparation for Synchronization between the Two Stages. 34](#_Toc487236008)

[4.3 Synchronization with GA 35](#_Toc487236009)

[4.3.1 Genetic Presentation 36](#_Toc487236010)

[4.3.2 Decode the genes to actual plan 37](#_Toc487236011)

[4.3.3 Compute fitness of individual genes 38](#_Toc487236012)

[4.3.4 Genetic operations 39](#_Toc487236013)

[4.3.5 Populated Genetic Algorithm 40](#_Toc487236014)

[CHAPTER 5. Experiment and Discussion 42](#_Toc487236015)

[5.1 Standard Test for the Proposed GA 42](#_Toc487236016)

[5.1.1 Testing on the Benchmark Problem 43](#_Toc487236017)

[5.1.2 Test on Simple Brewery Production Problem. 45](#_Toc487236018)

[5.2 The proposed Approach with Industrial Production Scale. 47](#_Toc487236019)

[5.2.1 Synchronization between the Two Production Stages 47](#_Toc487236020)

[5.3 Plan Presentation and Validation 49](#_Toc487236021)

[5.4 Relation between population size, number of generations and fitness 52](#_Toc487236022)

[5.5 The Relation between Mutate, Crossover and Evolution Process within the Proposed GA 53](#_Toc487236023)

[CHAPTER 6. Conclusion 54](#_Toc487236024)

[6.1 New Contributions 55](#_Toc487236025)

[6.2 Limitations and Further Research 55](#_Toc487236026)

[References 57](#_Toc487236027)

[Appendices 60](#_Toc487236028)

Table of Figures

[Figure 1: Typical scheduling problems (Zhang et al. 2006) 5](#_Toc487545648)

[Figure 2: Graph of job-shop scheduling problem (Adams et al. 1988) 6](#_Toc487545649)

[Figure 3: Operation components of jobs (Zhang et al. 2006) 8](#_Toc487545650)

[Figure 4: The precedence constraint of operation in each job (Baldo et al. 2014) 9](#_Toc487545651)

[Figure 5: The production process in brewery industry (Baldo et al. 2014) 10](#_Toc487545652)

[Figure 6: Production schedule situation in brewery industry (Baldo et al. 2014) 11](#_Toc487545653)

[Figure 7: GA/math programming approach (Toledo et al. 2014) 13](#_Toc487545654)

[Figure 8: Point to point search versus populated based search (Mitsuo et al. 2008) 15](#_Toc487545655)

[Figure 9: A graph G (Mitsuo et al. 2008) 16](#_Toc487545656)

[Figure 10: Adjacency matrix presentation of the graph G (Mitsuo et al. 2008) 17](#_Toc487545657)

[Figure 11: Variable-length encoding (Mitsuo et al. 2008) 17](#_Toc487545658)

[Figure 12: Fixed-length encoding (Mitsuo et al. 2008) 18](#_Toc487545659)

[Figure 13: Priority-based encoding technique (Mitsuo et al. 2008) 19](#_Toc487545660)

[Figure 14: The general structure of hybrid GA (Lin and Gen 2009) 20](#_Toc487545661)

[Figure 15: Applying local search to GA (Lin and Gen 2009) 21](#_Toc487545662)

[Figure 16: Beer production process (www.shutterstock.com) 23](#_Toc487545663)

[Figure 17: Two-stage production process (Baldo et al. 2014) 24](#_Toc487545664)

[Figure 18: Sabeco product set 28](#_Toc487545665)

[Figure 19: Production plan for testing the proposed algorithm in Gantt Chart format 46](#_Toc487545666)

[Figure 20: The GA learning curve (n = 0.4, population size = 100, generation number = 2000, mutate = 0.3, crossover = 0.7) 49](#_Toc487545667)

[Figure 21: Plan present in Gantt Chart format 51](#_Toc487545668)

[Figure 22: Relation between population size, number of generations and fitness (n = 0.4 mutate = 0.3, crossover = 0.7) 52](#_Toc487545669)

[Figure 23: Relation between crossover, mutation probability and fitness (n = 0.4, population size = 50, generation = 1500) 53](#_Toc487545670)

List of Tables

[Table 1: The jobs, operations, precedence constraints 15](#_Toc485692871)

[Table 2: Sabeco product set. 34](#_Toc485692872)

[Table 3: Data input table 34](#_Toc485692873)

[Table 4: Machines capacity table 35](#_Toc485692874)

[Table 5: Jobs assignment to tank types 38](#_Toc485692875)

[Table 6: job assignment experiment 39](#_Toc485692876)

[Table 7: Job assignment to tank type with n = 0.5 39](#_Toc485692877)

[Table 8: A snapshot of processing time table 41](#_Toc485692878)

[Table 9: A snapshot of genes presentation 42](#_Toc485692879)

[Table 10: Machines selection presentation 42](#_Toc485692880)

[Table 11: The benchmark flexible jobs shop schedule with 8 jobs and 8 machines 49](#_Toc485692881)

[Table 12: Testing Result on flexible jobs shop scheduling problem 8 jobs, 8 machines (10 first experiments only), GA parameters setting: population size = 100, generation = 6000, mutate probability = 0.9, crossover = 0.9 50](#_Toc485692882)

[Table 13: 8 jobs, 8 machines testing summary 50](#_Toc485692883)

[Table 14: Input Data for testing the proposed algorithm 51](#_Toc485692884)

[Table 15: Assignment result for testing the proposed algorithm n = 0.4 52](#_Toc485692885)

[Table 16: Input data, large scale production 53](#_Toc485692886)

[Table 17: Scenario analysis for synchronization (population = 100, generation = 2000, mutate = 0.3, crossover = 0.7) 54](#_Toc485692887)

[Table 18: Plan presentation in job and operation perspective, a snapshot 56](#_Toc485692888)

[Table 19: Plan presentation in machine and time perspective, a snapshot 56](#_Toc485692889)

# Introduction

## Motivations

As the result of booming economy and drinking culture in Vietnam, the beer consumption in Vietnam has increased very fast in the past decade. In 2016 brewery consumption in Vietnam increased 10 percent compared to 2015 and 50 percent compared to 2010 (according to Vietnamese Ministry of Public Health). Now, Vietnam is the largest beer consumption in South East Asia, and the third largest beer consumption in Asia just after China and Japan (according to Vietnamese Ministry of Public Health). In 2016, Vietnamese consumers consumed 3.78 billion litter of beer. Beside huge consumption volume, the brewery market in Vietnam is still predicted growing fast. According to Euromonitor, the legal drinking age population in Vietnam will increase from 68.7 million in 2016 to 72.4 million by 2021, and the Vietnamese government is expecting that the growth rate of the economy is around 6.5 percent in the next 5 years. Therefore, Vietnamese brewery market is very huge and promising. Although, local brewery manufacturing company like Sabeco (the largest beer manufacturing company in Vietnam with 40.59 percent of the market share) produces huge amount of brewery products but the company still does not have an effective computerized scheduling function.

The last time I visited a Sabeco brewery manufacturing plant in my hometown, it has no computerized scheduling function. The plan receives order from the header quarter in Ho Chi Minh city and produce 3 shifts per day if needed. The fact that brewery industry is a huge industry in Vietnam and there still is no computerized scheduling function (at least with the local company like Sabeco). These two facts are the motivation for me to develop a scheduling function for the brewery industry. And the chosen method is GA because GA has become popular in recent years and it is considered as an effective algorithm for optimization problem in general as well as scheduling problem specifically.

## Objectives

The motivation for this research thesis comes from the real industrial beer manufacturing case in Ninh Thuan province in Vietnam. During a summer vacation in 2016, I have had a trip to visit a beer manufacturing plant in my hometown (Ninh Thuan provine, Vietnam). At that time, I realized that the plant manager needs a better production planning tool. Therefore, I have been thinking about combining the duty of writing a research thesis with solving the real world industrial case. At the result of this thought, this research thesis ultimate goal is to build a scheduling function for brewery industry. A long with this major goal, the other objectives of this research thesis are as flowing:

* Give the reader comprehensive review about scheduling literature.
* Conduct a comprehensive review about GA in recent years.
* Give an interesting introduction about GA to the readers.
* Apply GA as a framework of the scheduling tool.

Rising from those objectives, this research thesis needs to answer the flowing questions:

* What are the major scheduling problems in literature discussion as well as the major scheduling problems in brewery industry.
* How to design an effective GA to solve the scheduling problem in the context of brewery industry.

## Research Framework

This thesis research concerning about applying GA to schedule brewery production. The thesis is organized into 5 chapters. Chapter 1 will discuss about the motivation and objectives of the research. In this section, there are some related questions to the research goal which is rising to give the audiences a comprehensive overview of the thesis content.

In chapter 2, some typical scheduling problems in literature are reviewed. These problems build the back ground understanding of scheduling problems. In addition, the literature related to brewery scheduling is discussed. The discussion of brewery scheduling literature includes brewery production processes, the difficulty or the problem needed to be solved by scheduling. Several models and algorithm from literature researchers also are discussed to give an overview of the past contribution for solving the scheduling problem in brewery industry. Finally, GA is be described both in general idea and application for network models. The application of GA in network model problem is discussed in detail since the brewery scheduling problem is modeled as a network problem. In Chapter 3, the brewery production processes is described in detail. The production processes understanding is very helpful in building the mathematical models. Since the goal of this thesis research is creating a scheduling function for a real case, the model need to be closed to the real case as much as possible. In chapter 4, the proposal genetic approach will be described in five components: 1. a genetic representation of potential solution to the problem, 2. a way to create population, 3. an evaluation function rating solutions in term of their fitness, 4. genetic operators that alter the genetic composition of the offspring (mutate, crossover, etc.), 5. parameters that GA uses (population size, probabilities of applying genetic operators, etc.). In chapter 5, severval experiments are conducted to give a better look on the behavior of the proposed GA. The contributions, limitations and further research direction are mention in chapter 6.

# Literature Review

## Scheduling Problems in Industrial Management

Scheduling is one of the most important functions in Industrial Management. There are many different models have been proposed to solve different scheduling problems in different industries. However, it is noticed that there are three major models in the field of scheduling which considered one plant, they are: the job-shop scheduling model, the flexible job-shop scheduling model, the integrated operation sequence & resource selection model. In Industrial Management, Scheduling function should create a good feasible solution whereas consider all the constraints(Mitsuo, 2008 #20;Cheng, 1995 #32):

* Material availability
* Machine and labor capacity
* Customer service level requirements (due dates)
* Inventory safety stock levels
* Cost
* Distribution requirements
* Sequencing for set-up efficiency
* etc.

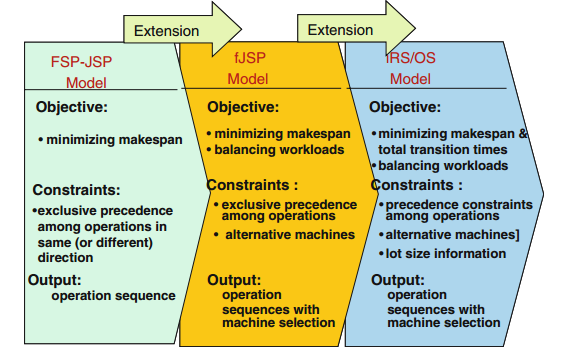


Figure 1: Typical scheduling problems (Zhang et al. 2006)

### The Job-Shop Scheduling Problem

Job-shop scheduling problem can be describe as flowing (Cheng et al. 1996): there are *m* different machines and *n* different jobs to be schedule. Each job is composed of a set of operations and the operation on machines is prespecified. Each operation is characterized by the required machine and the fixed processing time. The assumption for the job shop scheduling problem includes:

* A job does not visit the same machines twice.
* There are no precedence constraints among operations of different jobs.
* Operation can not be interrupted.
* Each machine can process only one job at a time.
* Neither release times nor due dates are specified.

The objective of the job-scheduling model is to determine the operation sequences on the machines in order to minimize the makespan (the completion time of all jobs).

Since the job-shop scheduling problem is classified as NP-hard, many researchers focus on heuristic approach to solve job-shop scheduling problem. There are many heuristic algorithms have been proposal range from simple dispatching rules to complex hybrid search algorithms. Haupt (1989) have done a comprehensive survey on priority rule-base scheduling. Adams et al. (1988) proposed the heuristic algorithm called the shift bottleneck procedure to solve the job-shop scheduling problem. This heuristic algorithm is considered one of the most effective heuristic algorithms to solve the shop-job scheduling problem. In the paper, the authors present the problem by disjunctive graph *G = (N, A, E)*, with a set of nodes *N*, a set of ordinary (conjunctive) arc *A*, and a set of disjunctive arc *E*. The disjunctive graph is illustrated by the figure below:

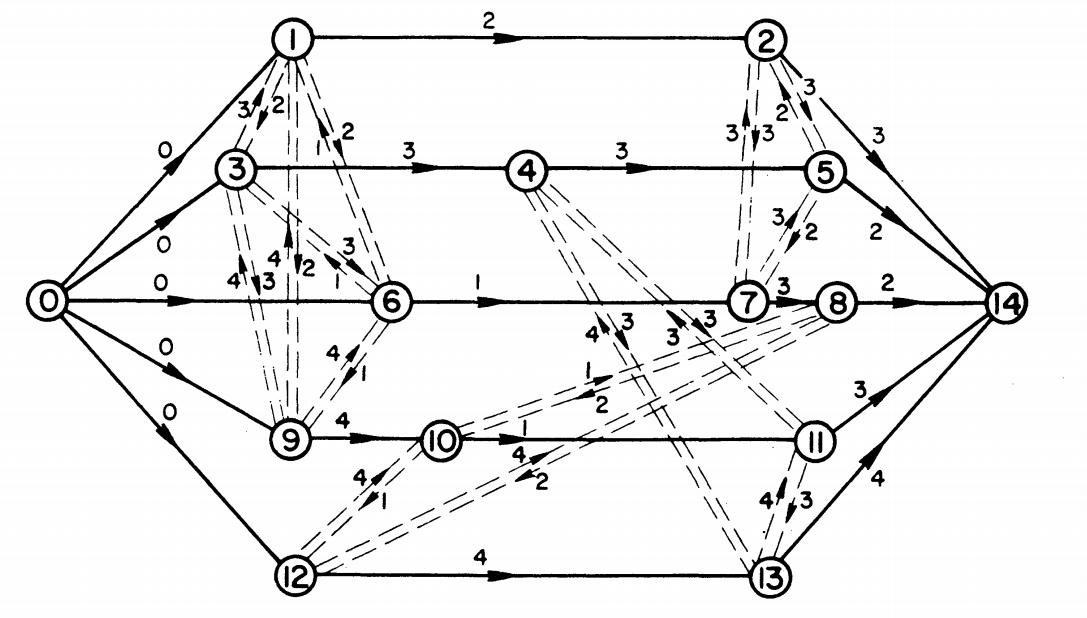


Figure 2: Graph of job-shop scheduling problem (Adams et al. 1988)

Figure 2 represents the job-shop scheduling problem with 15 operations (5 jobs) on 4 machines. The nodes of *G* present operations, the directed arcs present the precedence operations, the pair of disjunctive arcs present possible sequence on the machine. The number on directed arc is the processing time of the operation. A brief description of the shifting bottleneck algorithm (Adams et al. 1988) is as flowing:

* Let be the set of machines already sequenced ( = ∅ at the start).
* Step 1: Identify a bottleneck machine *m* among the machines *k* ∈ *M\* and sequence it optimally. Set ← ∪ {*m*} and go to 2.
* Step 2: Reoptimize the sequence of each critical machine *k* ∈ in turn, while keeping the other sequences fixed; i.e., set : = - {k} and solve P(k, ). Then if = *M*, stop; otherwise go step 1.

The main contribution of the paper is not the idea of solving the sequencing problem one by one, but the way of using the definition of bottleneck machine to decide which machine to be scheduled. This approach based on the idea of giving the priority to the bottleneck machine. In recent year, the GA has become more popular. Many researchers have tried to use GA to solve scheduling problem, such as Cheng et al. (1995), Xia et al. (2016), Kundakci and Kulak (2016). In recent years, many researchers focus on the dynamic aspect of scheduling and using hybrid GA to solve the dynamic scheduling problem. Kundakci and Kulak (2016) proposed a hybrid GA to solve the dynamic job-shop scheduling problem. Xia et al. (2016) proposed an integrated process planning and scheduling model and solved it by using the hybrid GA with variable neighborhood search.

### The Flexible Job-Shop Scheduling Problem.

The flexible job-shop scheduling problem is the extended problem of the job-shop scheduling problem. That is a machine can process more than one type of operation. Meaning that, for given any operation, there must exist at least one machine capable of performing it (Mitsuo et al. 2008). There are two kind of flexible on the machines (Kacem et al. 2002):

* Total flexibility: all operations can be processed on all machines.
* Partial flexibility: some operations can only be processed on a sub set of machines.

Flexible job-shop scheduling problem can be described as flow: there are *n* jobs to be schedule on m machine. Each job *i* contained ordered operations. A machine can only process one operation as a time, and the machine is busy until the operation is complete. The flexible job-shop problem is to assign operations on machines and to schedule operations assigned on each machine, subject to the constraints that:

* The operation sequence of each job is prescribed.
* Each machine can process only one operation as a time.

Since, flexible job-shop scheduling problem is the generalized problem of job-shop scheduling problem, which is even harder to solve. In recent year, the heuristic approach to solve the problem in reasonable time is popular (Gen et al. 2009, Pezzella et al. 2008).

### The Integrated Operation Sequence and Resource Selection Problem:

The integrated operation sequence and resource selection problem is extended from the traditional scheduling problem (the job-shop scheduling problem and the flexible job-shop scheduling problem), which is closer to the real manufacturing environment. For example, job is combined by a set of operations. Some operations have precedence constraint, other do not (Mitsuo et al. 2008):

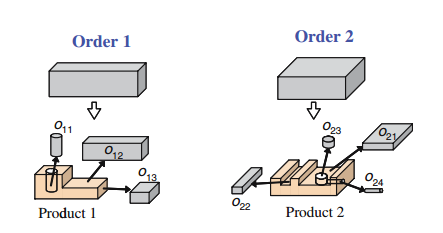


Figure 3: Operation components of jobs (Zhang et al. 2006)

Table 1: The jobs, operations, precedence constraints

|  |  |  |
| --- | --- | --- |
| Job | Operation | Precedence constraint |
| 1 | O11, O12, O13 | (O12, O13) |
| 2 | O21, O22, O23, O24 | (O21, O23), (O23, O24), (O24, O22) |

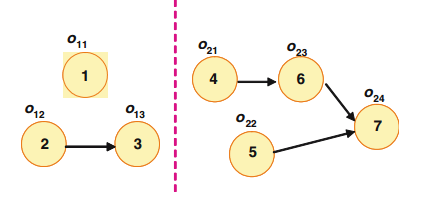


Figure 4: The precedence constraint of operation in each job (Baldo et al. 2014)

The integrated operation sequence and resource selection problem can be describe as flow (Zhang et al. 2006): given a set of K orders with lot size find an operations sequence for each job, a schedule in which jobs pass between machines and a schedule in which operations on the same jobs are processed such that it satisfies the precedence constraints and it is optimal with respect to minimize makespan and balance the workloads.

Since the integrated operation sequence and resource selection problem is resembling with the real world manufacturing problem, it has been received great attention from researchers. However, the problem is a NP-hard problem and it is even more complex than the job-shop scheduling problem and the flexible job-shop scheduling problem. Therefore, to solve the problem, many heuristic algorithms have been proposed and the GA is one of the main branches of the movement. Zhang et al. (2006) proposed two vector-based coding approach for encoding the operation sequence and resource selection in chromosome. They also used multistage operation-based GA combine with left-shift hillclimber to speed up the computation time. Amin-Naseri and Afshari (2012) proposed the similar approach with (Zhang et al. 2006), which is used GA combined with local search technique to speed up the processing time.

### The Scheduling Problem in Soft Drink and Brewery Industry.

The production process of soft drink and brewery industry is characterized by two major production stages: liquid preparation (stage I), bottling (stage II), and the sequence dependence between the two stage, the raw material have to go through the two processing stage consecutively in order to become a complete product (Ferreira et al. 2009), (Baldo et al. 2014).

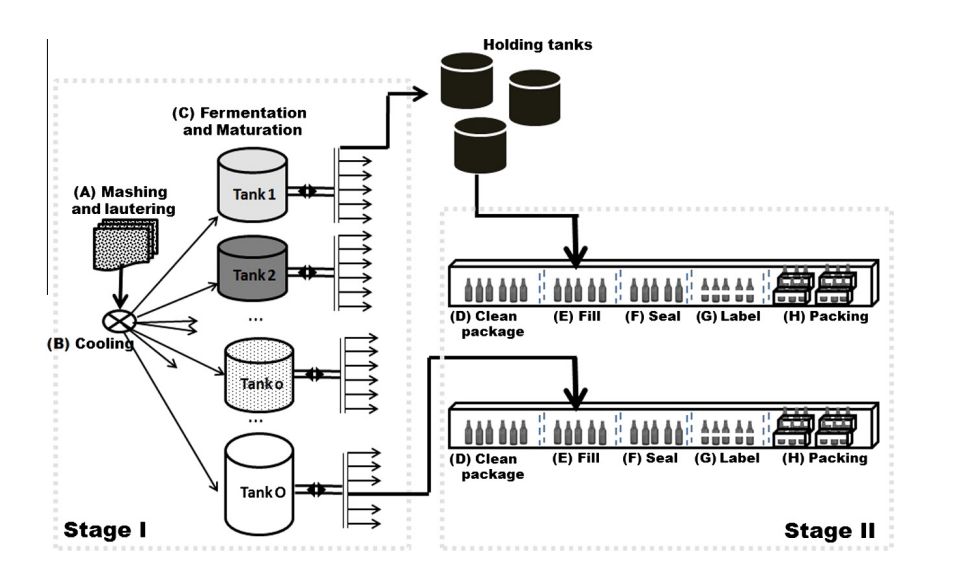


Figure 5: The production process in brewery industry (Baldo et al. 2014)

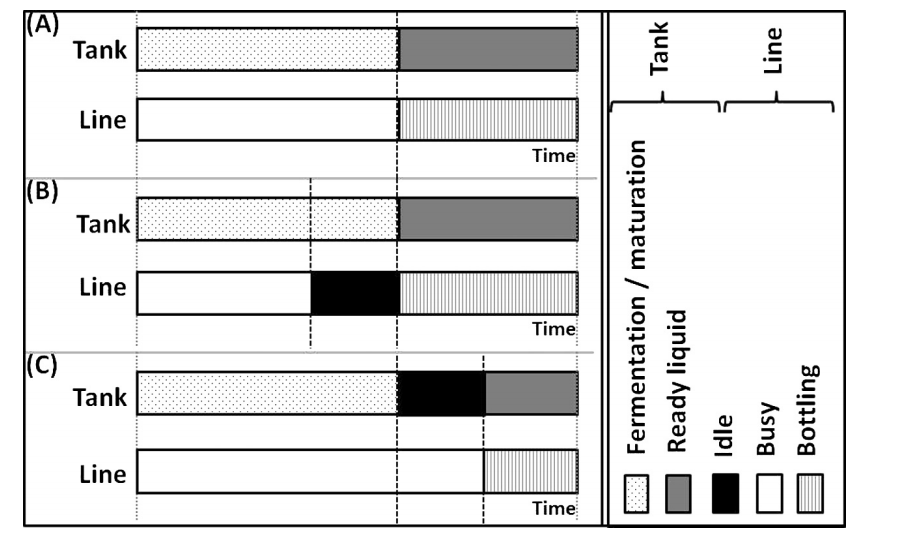


Figure 6: Production schedule situation in brewery industry (Baldo et al. 2014)

One of the important questions in soft drink and brewery industry is how to use the source (tanks and bottling lines) effectively. The figure 6 depicts situations in production process of a typical brewery production line. Situation (A) is the ideal situation, in which the tank and line are synchronized, the ready liquid in the tank go immediately to the filling line. Situation (B), the filling line has to wait for ready liquid. Situation (C), the liquid in the tank is ready but the filling line is busy for other job, then it has to wait. The problem rising from production is how to synchronize production process between the two stages, reduce the bottle neck and increase utilization of the asset.

The scheduling problem in soft drink and brewery industry usually is formulated as a variance of the general lot sizing and scheduling problem (Fleischmann and Meyr 1997). The model schedule a set of products= *1, 2, …, J* over a finite planning horizon containing a set of macro periods *t = 1, 2, …, T*. Each macro-period *t* is divided into a fixed number of non-overlapping micro-periods with variable length, where denotes the set of micro-periods *s* belonging to the macro-period *t* and all micro-periods are ordered in the sequence *s = 1,..., S*. The two layer of time structure is based on the idea that the external dynamics of the system (demands and holding costs, etc.) should be modeled by a fixed discrete time grid. And, the internal dynamics of the system (the changes of system stage) should be independent on external dynamics, and controlled by decisions. Both internal and external dynamics are linked by the assignment of micro-periods to macro-periods. The length of micro period is also decision variable but expressed in the model by the quantity produced within. The number of micro-periods || within macro-period t is fixed in advance to allow MIP-modeling. It sets an upper bound to the number of possible changeovers within macro period t. A lot consists of a sequence of micro-periods assigned to the same item and may continue over different macro-periods.

The general lot sizing and scheduling problem then was adapted by Ferreira et al. (2009) to formulate the two-stages model for soft drink production. Ferreira et al. (2012) formulate the two stage production process with single-stage formulation. Baldo et al. (2014) modeled the brewery production process with two-stage MIP model and solve it with relax-and-fix, relax-and-optimize heuristic. Toledo et al. (2014) used heuristic GA to solve the lot sizing and scheduling problem in soft drink industry. In their approach, GA deals with sequencing decisions for production lots, after that the remaining linear model gives answer for lot sizing decision. The figure 7 depicts the GA and math programming approach for two-stage production process in soft drink industry.

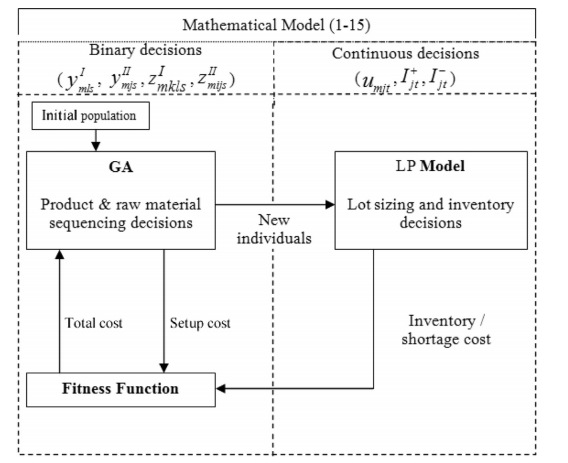


Figure 7: GA/math programming approach (Toledo et al. 2014)

## GA Approach

GA is a class of heuristic optimization algorithm. It is inspired by the natural evolutionary process. Scientists have proved that all living things in the planet are made from small elements call genes. Genes decides all the characteristic of a living organism or individual. Through the pressure of the evolutionary process, the individual with better characteristics will have more change to survive and pass their genes to other generations through breeding. Gene material keeps evolving through natural selection process.

GA is designed to mimic the genetic evolutionary process. The solutions are encoded in genes. Genes keep evolving through operations like crossover (exchange genes material between individuals) and mutation (change genes material in an individual). A selection mechanism is necessary in GA to decide how good an individual is, and how the genes evolution process is conducting.

### GA in General

Computer scientist has been dreaming about one type of algorithm that can solve different type of optimization problem. The GA has been born from that desire and it try to mimic the natural evolutionary process. Indeed, the GA has been applied to many fields such as Finance, Supply Chain Management, and Information Technology. Orito et al. (2009) used GA solve Index Fund Optimization Problem, which is a very useful tool for hedge trading strategy in Financial Market. Leu and Namatame (2008) use GA to solve optimize network design problem.

As mention above, GA can be used as optimization solver for a wide range of problem in many fields. However the principle structure of the algorithm is the same across applications as mention by Mitsuo et al. (2008). In general, a GA has five basic components:

1. A genetic representation of potential solutions to the problem.
2. A way to create a population (an initial set of potential solutions).
3. An evaluation function rating solutions in term of their fitness.
4. Genetic operators that alter the genetic composition of offspring (crossover, mutation, selection, etc.).
5. Parameter values that GA uses (population size, probabilities of applying genetic operators, etc.).

Mitsuo et al. (2008) pointed out the important advantage of GA compare to the conventional optimization search method that is GA is populated base search approach.

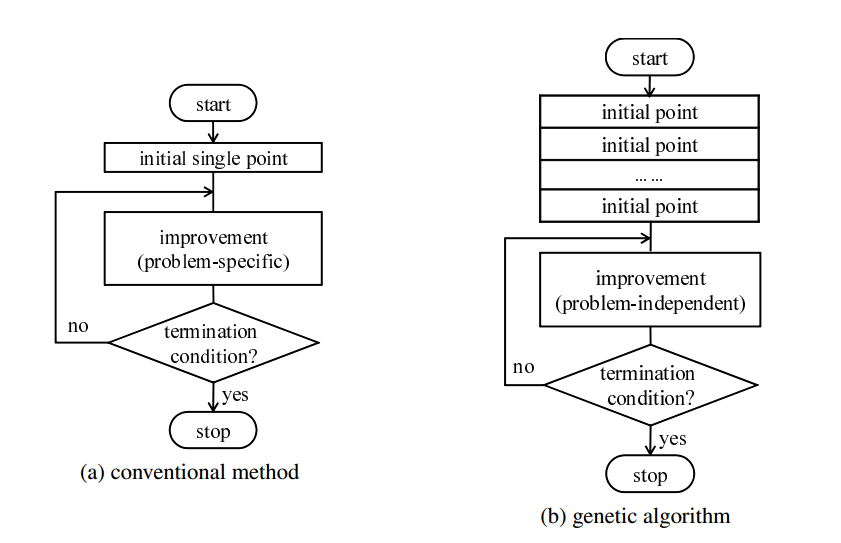


Figure 8: Point to point search versus populated based search (Mitsuo et al. 2008)

The conventional approach is point to point search, can be trap in the local optimum, whereas GA is population to population search method with a degree of randomness. Therefore, the probability of being trap in local optimum of GA is lower than the conventional optimization methods. GA can get good solution in a short period of time since it has both good combination of exploration (search randomly) and exploitation (search in certain direction). At first, the random initialization of the population helps GA explore the search space. As the fitness improving through generations, the crossover, mutation functions help solution move to a certain direction to get the better solution quicker.

### GA and Network Modeling

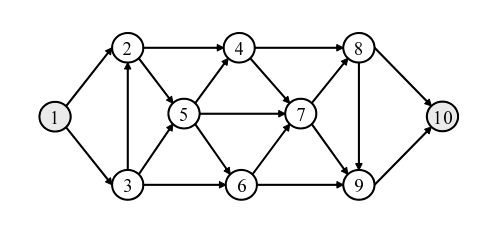


Figure 9: A graph G (Mitsuo et al. 2008)

The directed graph (Figure 9) G can be presented in three different ways: edge lists, adjacency lists, adjacency matrix. In the first way, the graph can be characterized by the number of nodes and a list of arcs. In the figure 9, the graph G can be presented by 10 nodes and a list of arcs *A* = {(1,2), (1,3), (2, 4), (2, 5), (3, 2), (3, 5), (3, 6), (4, 7), (4, 8), (5, 4), (5, 6), (5, 7), (6, 7), (6, 9), (7, 8), (7, 9), (8, 9), (8, 10), (9, 10)}. In the second way, a graph can be described by the number of nodes and n list , …, , …, , where contains all nodes j for which G contains an arc (i, j). For example, the graph G can be presented by 10 nodes and the adjacency lists: = {2,3}, = {4,5}, = {2,5,6}, = {7,8}, = {4,6,7}, = {7,9}, = {8,9}, = {9,10}, = {10} and = ∅. In the third way, a graph of n nodes can be presented by a (n\*n) adjacency matrix *A = (),* where = 1 if and only if (*i, j)* is an arc of *G*, and = 0 otherwise.

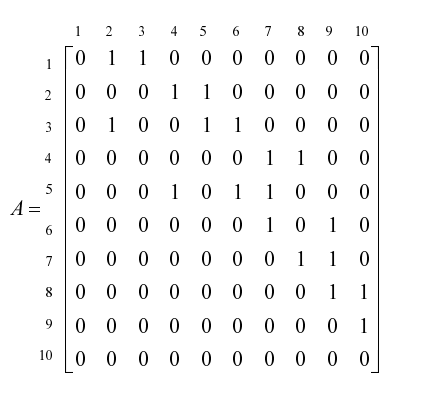


Figure 10: Adjacency matrix presentation of the graph G (Mitsuo et al. 2008)

Genetic presentation is one of the five major components of a GA. Indeed, a good GA has to be both fast and accuracy. The accuracy ability is to present a feasible solution in the chromosome as well as make sure the feasibility of the chromosome after going through all the operations (mutation, crossover, etc.). Feasible solutions can be presented in the chromosome in several ways. Chang Wook and Ramakrishna (2002) use variable-length encoding technique to solve the shortest path routing problem. Inagaki et al. (1999) use fixed-length encoding technique to solve the multiple routing problem.

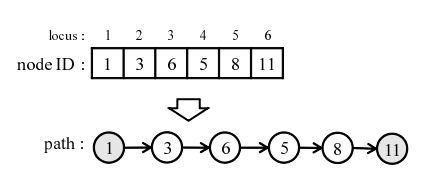


Figure 11: Variable-length encoding (Mitsuo et al. 2008)

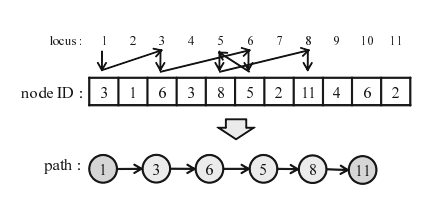


Figure 12: Fixed-length encoding (Mitsuo et al. 2008)

In the variable-length encoding technique, locus (the position of gene in chromosome) sequence presents the sequence of nodes that the path goes through, and the value of gene is the id of the node (figure 11). In fixed-length encoding technique, to present an arc from node *i* to node *j*, put node *j* to the *i-th* locus. This process is reiterated from source node 1 through sink node n. If the route does not path through a node *x*, select one node randomly from the set of nodes which are not connect to node *x* and put it in the *x-th* locus. The disadvantage of variable-length and fixed-length encoding technique (Mitsuo et al. 2008) is that the infeasible chromosome can be created through genetic operation.

In recent year, priority-based encoding technique is become popular. Mitsuo et al. (2008) listed the advantage of priority-based encoding technique as flowing: (1) any permutation of the encoding corresponds to a path (feasibility), (2) most existing genetic operators can be easily applied to the encoding, (3) any path has a corresponding encoding (legality), (4) any point in solution space is accessible for genetic search.

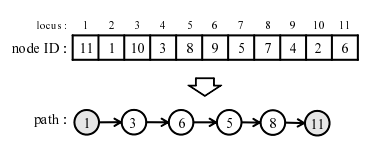


Figure 13: Priority-based encoding technique (Mitsuo et al. 2008)

In the priority-based encoding technique, the locus (position of gene in chromosome) represents the id of node. The alleles (the value of gene) represent the priority level of that node. To illustrate the priority-based encoding technique, let take the example from figure 13. First, we try to find the next node from the source 1. From the adjacency list S1, we know node 3 and node 2 are eligible. However node 3 has higher priority than node 2, therefore node 3 is in the path. The iteration continues with adjacency list S3 until we get to the path. The priority-based encoding technique was used by (Zhang et al. 2006), (Gen et al. 2009).

In recent year, GA has evolved in to hybrid heuristic search method, which is the combination between genetic search and local search. The important question is how to design a good combination between the two search method to get a reliable and fast hybrid GA.

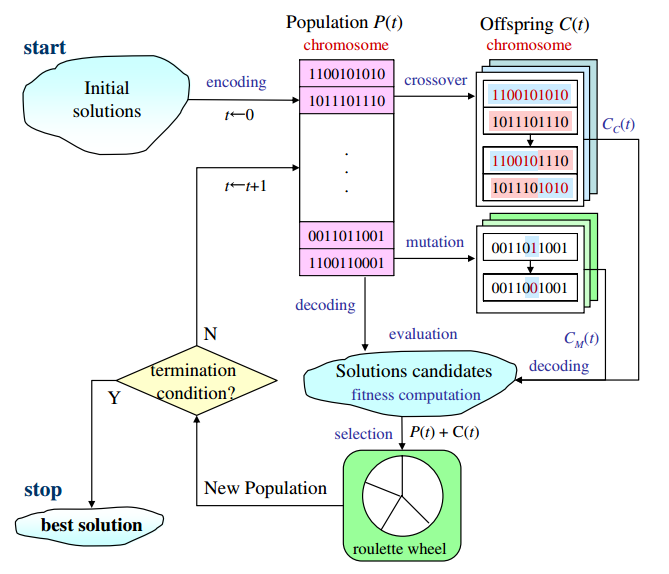


Figure 14: The general structure of hybrid GA (Lin and Gen 2009)

The general structure of hybrid GA is depicted by figure 14. The traditional local optimization search (hill-climbing, etc.) is applied to help offspring get better genes and then be put back to the population. By adding a local search algorithm to each iteration of GA, we have better algorithm. Since, the algorithm has two good characteristics from both GA and local search algorithm. The randomness from GA will help the hybrid GA avoid being trapped in local optimization. The directed search from local search algorithm will help the hybrid GA move toward the local optimum faster than the GA alone.

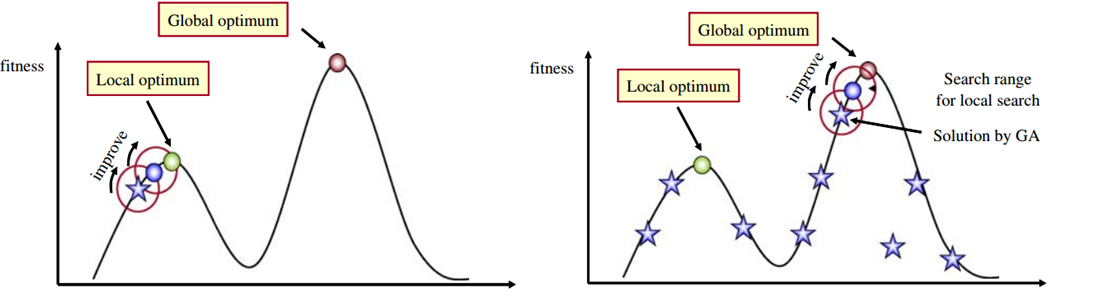


Figure 15: Applying local search to GA (Lin and Gen 2009)

Lin and Gen (2009) proposed a very interesting approach called auto-tuning strategy. GA was invented based on the evolutionary idea in which the algorithm adapts itself to the type of problems. It is desired to have a GA that can adapt to different type of problems and auto-tuning strategy is one of the answers. The auto-tuning strategy (Lin and Gen 2009) help the GA balancing between exploration (random search) and exploitation (directed search) by control the probability of genetic operators. Generally speaking, if the GA is biased toward exploitation then the probability of random search is low and the probability of local search is high. On the other hand, if the GA is biased toward exploitation then the probability of random search is high and the probability of local search is low. We can control the balance between exploitation and exploration by controlling the probability of GA operators like cross-over (neural), immigration (exploration bias), mutation (exploitation bias).

# Proposed Methodology

## Beer Production Process

The main ingredients for making beers include: barley, hops (responsible for the bitterness of beer), water and yeast (responsible for transforming sugar to alcohol and C02). The manufacturing process is varying between different manufacturers, however the core production process is the same for all brewery industry. There are five major steps for beers making. The first step is malting. The grains (mainly barley) are harvested and processed through a process of heating, drying out and cracking. The main goal of malting is to isolate the enzymes needed for brewing. The second step is mashing in which grains are steeped in hot (but not boiling) water for an hour. Then, water is drained out from the mash. This activates enzymes in the grains that cause it to break down and release sugars. The result of mashing is sticky water which contains sugar extracted from the mash, called *wort*. The thirst step is boiling, in which wort is boiled for about an hour while ops and other spices are added several times. The fourth step is fermentation, after the boiling step, the wort is cooled, strained and filter. It is then put in a fermenting vessel and yeast is added to it. At this point, the brewing is completed and the fermentation begins. The beer is stored for a couple of weeks at room temperature (in the case of ales) or many weeks at cold temperatures (in the case of lagers) while the yeast transform the sugar in the wort. The final step is bottling. The ready liquid will be poured to the bottle in a filling line.



Figure 16: Beer production process (www.shutterstock.com)

## Problem Description and Modeling

In brewery production process, there are two key types of machines. They are tanks and filling lines, these types of machines are also the bottleneck of production processes. Tanks are used to hold the liquid and the holding time can take from one or two weeks to one or two months depending on the type of beers. Filling lines are used to fill the bottles with liquid. Filling lines are the most important machines in brewery manufacturing plant. Due to the long fermentation in tanks and the expensive filling lines, the task of synchronizing operation between the two types of machines is very important in brewery production.

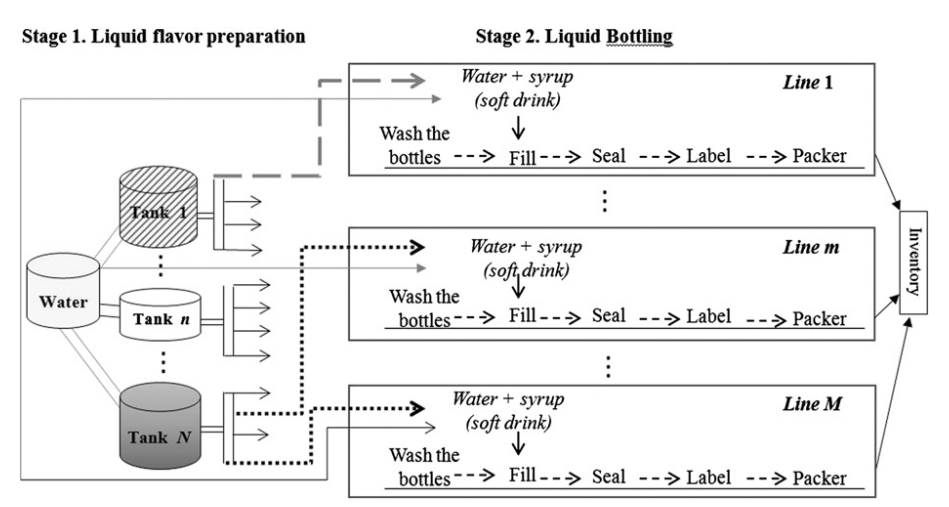


Figure 17: Two-stage production process (Baldo et al. 2014)

The figure 17 shows that the brewery production process is characterized by two major operations. They are fermentation and bottle filling. To dealing with the scheduling problem in brewery industry, this thesis proposes an approach which is the combination of math programming and GA for flexible job shop scheduling problem. The proposed approach is come from the idea of formulating the brewery production as a flexible jobs shop scheduling problem while take into account of the characteristics of the brewery production process such that:

* The fermentation time of a product is fixed and unrelated with the volume of production, while the bottle filling time is related with the production volume and filling machines capacity in a linear fashion. Therefore, the operation time of fermentation and filling should be computed differently.
* The sequence constraint of operations in the same job is reserved. The filling operation can only be processed when the fermentation operation is finished.
* A machine can only process an operation as a time. A machine can only process a set of operations base on their function. For example, the bottle filling line can only fill bottle products, the can filling line can only fill can products.
* A job will enter the machine immediately when the previous job is finished.

Firstly, the tank type assignment problem is solved by a math model such that it minimizes the fermentation time. The math model is described as flowing:

**Set:**

p ∈ P with P is a set of products.

t ∈ T with T is a set of tank types.

**Decision variable and parameter:**

: The number of time tank type t was used for producing product p.

: Fermentation time of product p.

: Demand for product p.

: Tank type t capacity.

**Objective function:**

Min (1)

**Constraint:**

≥ ∀ p ∈ P (2)

∀ ∈ P (3)

≥ 0, ∈ Z (4)

The objective function (1) is to minimize the fermentation time of a given quantity demand. Constraint (2) is to make sure that the demand always be fulfilled. Constraint (3) is to make sure that tank types assignment is use equally. It is depend on the specific tanks availability on the actual plant, planner can add constant coefficients into the equations (3) to use more of a curtain tank type or less of other tank types. Equation (4) is the valid condition of decision variables. Note that in this model, tank types are considered but not specific tanks, because the second GA algorithm will help to choose specific machines in a flexible job shop scheduling problem.

The second problem is the synchronization between tanks and filling machines. To solve this problem a GA algorithm is proposed to dealing with flexible job shop scheduling problem. The flexible job shop scheduling problem is described by a math formulation as flowing (Mitsuo et al. 2008):

**Set**:

i ∈ I with I is a set of jobs.

j ∈ J with J is a set of machines.

k ∈ K with K is a set of operations.

**Parameter:**

: total number of operations in job i.

: the i-th job.

: the k-th operation of job i,

: the j-th machine.

: processing time of operation on machine j.

U: a set of machines with the size m.

: a set of available machines for the operation .

: workload (total processing time) of machine j.

**Decision variables:**

= 1 if machine j is selected for operation , 0 otherwise.

= completion time of the operation .

**Objective function:**

Min = (5)

Min = {} (6)

Min = (7)

**Constraints:**

- - ≥ 0 k = 2, …, K; ∀ i, j (8)

= 1 ∀ k, i (9)

∈ 0, 1 ∀ j, k, i (10)

≥ 0 (11)

The objective function (5) is to minimize the makespan of all the jobs. The combination of objective function (6) and (7) consider the total processing time and dispatching the operations averagely over machines. The objective function (6) and (7) help to drive the solution toward balance operation assignment between machines. Constraint (8) states that the successive operation has to be started after the completion of its precedent operation of the same job, which represents the operation precedence constraints. Constraint (9) states that one machine has to be selection for each operation. Constraint (10) and constraint (11) are the valid condition for decision variable.

## Data Collection and Input Design

The data for this thesis experiment is collection from Saigon Alcohol Beer and Beverages Corporation (Sabeco). The data is collected through face to face interview and internet searching. The main source of information is come from the Sabeco official website ([www.sabeco.com.vn](http://www.sabeco.com.vn))

Table 2: Sabeco product set.

|  |  |  |  |
| --- | --- | --- | --- |
| **Product Code** | **Product Name** | **Beer Type** | **Volume/bottle or can** |
| **B1** | 333 premium (bottle) | premium lager | 330ml |
| **B2** | 333 (can) | lager | 330ml |
| **B3** | saigon lager (bottle) | lager | 450ml |
| **B4** | saigon lager (can) | lager | 330ml |
| **B5** | saigon export (bottle) | lager | 355ml |
| **B6** | saigon special (bottle) | lager | 330ml |
| **B7** | saigon special (can) | lager | 330ml |



Figure 18: Sabeco product set

Table 3: Data input table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Product Code** | **Product Name** | **Volume/bottle** | **Demand** | **Beer Type** | **Fermentation Time (day)** |
| **B1** | 333 premium (bottle) | 330ml | production demand | premium lager | 28 |
| **B2** | 333 (can) | 330ml | production demand | lager | 21 |
| **B3** | saigon lager (bottle) | 450ml | production demand | lager | 21 |
| **B4** | saigon lager (can) | 330ml | production demand | lager | 21 |
| **B5** | saigon export (bottle) | 355ml | production demand | lager | 28 |
| **B6** | saigon special (bottle) | 330ml | production demand | lager | 28 |
| **B7** | saigon special (can) | 330ml | production demand | lager | 28 |

Table 4: Machines capacity table

|  |  |  |
| --- | --- | --- |
| **Machine** | **Capacity** | **unit** |
| **TA1** | 500 | BBL |
| **TA2** | 500 | BBL |
| **TA3** | 500 | BBL |
| **TA4** | 500 | BBL |
| **TA5** | 500 | BBL |
| **TB1** | 1000 | BBL |
| **TB2** | 1000 | BBL |
| **TB3** | 1000 | BBL |
| **TB4** | 1000 | BBL |
| **TB5** | 1000 | BBL |
| **TB6** | 1000 | BBL |
| **TB7** | 1000 | BBL |
| **TB8** | 1000 | BBL |
| **FC1** | 30000 | can/hour |
| **FB1** | 35000 | bottle/hour |

Sabeco is the largest brewery producer in Vietnam. In 2013 Sabeco produced 1,330 billion litter of beer. The company has around 24 plants in Vietnam with the capacity of 1.8 billion litters (According to Sabeco website: <http://sabeco.com.vn>). For a large brewery producer like Sabeco, large tanks and high capacity filling line are used in production process. Tanks with the capacity of 1000 BBL (1 BBL (barrel) = 1.17 hectoliters (hl), and 1 hectoliter = 100 liters (l), 1 litter = 1000 milliliters (ml), ml is the volume measure for beer bottle or can in Vietnam) and 500 BBL, bottle filling line and can filling line with the capacity around 30.000 bottle or can/hour (According to Sabeco website: <http://sabeco.com.vn>).

The program in this thesis is designed so that the users can easily get their schedule result given the demand quantity that they want to produce. Products and machines are coding so that it is easy to make reference and compute the result in computer system.

The coding for products and machines are explained as flowing:

* B: product (beer) range from 1 to 7.
* TA: tank type A which has the capacity of 500 BBL.
* TB: tank type B which has the capacity of 1000 BBL.
* FC: can filling line which has the capacity of 30.000 cans/hour.
* FB: bottle filling line which has the capacity of 35.000 bottles/hour

Note: all the information about the product set and machine type is taken from Sabeco website <http://sabeco.com.vn>, exact machine is based on the assumption of author. The program is design with flexibility in mind. The user can change, contract or expand machines setting and the program should work fine.

# Solution Approach and Sabeco Case Study

To solve the scheduling problem in brewery industry, this thesis proposes an approach of two stage computation. First step, given the demand of different products the program will assign jobs to tank types. The tank types are considered instead of specific tanks because of flexible machines chosen purpose for the later GA algorithm for flexible job shop scheduling problem. Second step is trying to synchronize the production process between fermentation state and filling bottle stage. This is where GA is deployed to help the production process more smoothing.

## Tank Types Assignment Result and Introduction to Synchronization between Tank Types Assignment and Filling Stages

Basically, Brewery production process includes two major operations. First, liquid is fermented in large tanks. After go through the fermentation time, the mature liquid is filled in bottle or can on the bottle filling line or the can filling line. The first stage of computation will automatically assign job to tank types in order to minimize the fermentation time. Base on the job assignment result, jobs and operations definition need to be defined in order to prepare the input for the synchronize production stage. Since the synchronization between fermentation stage and filling stage is modeled as flexible jobs shop scheduling problem, and the input of the flexible jobs shop scheduling problem is the operation time of each operation on each machine, therefore the operation time of each operation on each machine need to be calculated. However, there is a fundamentally difference between the operation time of filling operations and fermentation operations. In filling operations, the time to filling a certain quantity of liquid is linear with the capacity of filling machine. Filling time equals to demand divided by the capacity of filling line. On the other hand, the fermentation time of a certain product is fixed and the tank capacity is not affect the fermentation time. Because of the intrinsic character of the brewery production process, it is needed for the first stage computation to solve the assignment of tank types problem, get the fermentation time of the fermentation operations and then compute the filling time for filling operations to prepare the input for synchronization stage. Before go the synchronization stage, it is important to discuss the result of the jobs assignment to tank types problem.

Table 5: Jobs assignment to tank types

|  |  |  |
| --- | --- | --- |
| **Tank Type** | **Product** | **Time** |
| **TA** | B2 | 2 |
| **TA** | B3 | 8 |
| **TA** | B4 | 6 |
| **TA** | B5 | 1 |
| **TA** | B7 | 2 |
| **TB** | B2 | 2 |
| **TB** | B3 | 8 |
| **TB** | B4 | 6 |
| **TB** | B5 | 1 |
| **TB** | B7 | 2 |

The jobs assignment result is presented in the table above. For example, product B2 is produced in 2 tank type A (TA) and 2 tank type B (TB), and so on. This is just one assignment synchronization stage.

For the purpose of finding the best combination between types of tanks (in this case, there are two types of tanks, TA and TB) the constraint (3) is rewritten with an experiment constant n as flowing:

≥ 0 ∀ ∈ P (12)

The equation (3) is introduced to balance the assignment between tank types, because tank type B (TB) capacity is larger than tank type A (TA) and the minimization model will tend to assign jobs to tank type B. The equation (12) is introduced to help find the assignment curve to support the planner choose what plan should be suitable.

Table 6: job assignment experiment

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **N** | **0.1** | **0.2** | **0.3** | **0.4** | **0.5** | **0.6** | **0.7** | **0.8** | **0.9** | **1** |
| **Objective** | 714 | 714 | 756 | 756 | 777 | 798 | 798 | 819 | 819 | 819 |

Table 7: Job assignment to tank type with n = 0.5

|  |  |  |
| --- | --- | --- |
| **Tank Type** | **Product** | **Time** |
| **TA** | B2 | 2 |
| **TA** | B3 | 5 |
| **TA** | B4 | 4 |
| **TA** | B5 | 2 |
| **TA** | B7 | 2 |
| **TB** | B2 | 2 |
| **TB** | B3 | 9 |
| **TB** | B4 | 7 |
| **TB** | B5 | 0 |
| **TB** | B7 | 2 |

From the table (6) it is clear that, the assignment model gets better objective value if n is smaller. Smaller value of n means that the assignment model assigns more jobs to tank type B which has larger capacity. However, the production process in brewery production is two stages production (fermentation and filling stages). The jobs assignment in tank will affect the filling stage. If we try to optimize each production stage separately then the whole production may not be optimized at all.

To dealing with two stages production process in brewery industry, the thesis proposed two stage of computation. The first job assignment to tanks model is served as a job assignment tool, the second GA flexible jobs shop scheduling problem will handle the operation sequence decision and machine selection decision therefore synchronize production between the two stages. Furthermore, it is necessary to try different configuration of tank assignment to find a good solution which consider the whole production process.

## Data Preparation for Synchronization between the Two Stages.

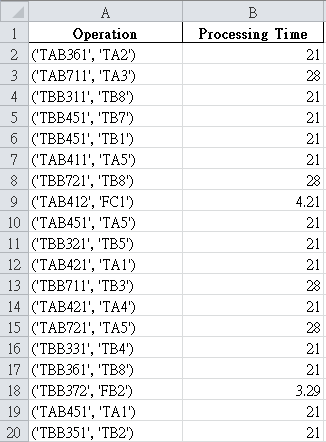
After the jobs assignment to tanks is finished. The data need to be processed more in order to be the input for GA flexible jobs shop scheduling problem. The preparation step is conducted as following:

If tank type A (TA) is assigned to use n time to produce SaiGon Larger (can) or product B4. Then the program will define the jobs as TAB41, TAB41, …, TAB4n. Since each job has only two operation (fermentation and filling). Each job TAB4n will be divided into two smaller operation TAB4n1 for fermentation and TAB4n2 for filling the liquid to bottle or can. Since the coding of job and operation have the information about tank types and products, we can use the information to get the fermentation time and compute the filling time of each operation.

Pseudocode for data preparation:

* Input: the tank types assignment for production, machines configuration.
* Output: processing time table. The processing time table is present as a dictionary with keys are operations and machines, values are the time to finish the operation given the machine.
* Step 1: given the tank types assignment create job list coding.
* Step 2: given the job list coding create the operation list coding.
* Step 3: given the operation list coding get the machine set that can process each operation (present like a dictionary with called operation machine dictionary).
* Step 4: given the operation list coding, operation machine dictionary, machine capacity get fermentation time or compute filling time for the operation.

Table 8: A snapshot of processing time table



Processing time table is the input for the synchronization stage. It is presented in form of a look up table. They keys are operation coding and machine coding while value is the processing time of the given operation on the given machine.

## Synchronization with GA

The synchronization between stages is model as a flexible jobs shop scheduling problem. Each job in brewery industry is characterized by two operations, they are fermentation operation and filling operation. The fermentation processing time is characterized by the type of product, while the filling processing time can compute from product volume, machine capacity. The scheduling problem in brewery industry is solved by using GA methodology. The proposed GA will deal with operations sequence decision and machine selection decision in order to find a good scheduling solution. The proposed GA not only try to minimize makespan but also try to balance the workload between machines.

### Genetic Presentation

Each individual genes (or solution encoding) is present by two vectors. Vector 1 presents operation sequences decision. Vector 2 presents machine selection decision.

Table 9: A sequence decision genes for simple case



The idea for this genes presentation was proposed by Gao et al. (2007). Each operations sequence will be presented by a list of job. For example, the first job coding ‘JBB61’ will present the operation coding ‘JBB611’, the first job coding ‘TBB31’ will present the operation coding ‘TBB311’, the second job coding ‘JBB61’ will present the operation coding ‘JBB612’ and so on. The sequence decision is read from left to right. Machines selection decision is presented by a list of machines and each machine is mapping with a given list of operations. The table below will present a machines selection decision genes for simple case.

Table 10: A machine selection genes for simple case



The coding of the operation give us the information about the operation itself, therefore we can use it reference to a set of machine that can process the operation. For example ‘TAB311’ is fermentation operation of product B3 on tank type A, and ‘TAB212’ is the filling operation of product B3.

### Decode the genes to actual plan

From the genes, the actual plaining is obtained through a decoding mechanism. The decoding mechanism is described as a pseudocode as flowing:

* Input: genes presentation, operations list, processing time table.
* Output: the plan like a dictionary with key is (‘operation’, ‘machine’) and value is starting time and completion time.

Step 1: from the genes presentation get a set of machines and a set of jobs.

Step 2: from a set of machines and a set of job create a dictionary of machines time and jobs time to keep track of machine time and jobs time. Machines time is to keep track of the current processing status in machine aspect. Job time is to keep track of the current processing status in job aspect.

Step 3: For each operation on the sequence decision:

Step 3.1 get the operation time of the current operation on the chosen machine from the processing time table.

Step 3.2: update the machines time of the chosen machine by operation time, update the jobs time of the current operation by the operation time.

Step 3.3: get the completion time of the current operation by choosing the maximum of machines time of the chosen machine and the jobs time of the current job.

Step 3.4: compute the starting time of the current operation by the completion time of the current operation – processing time of the current operation.

Step 3.5: update both machines time of the chosen machine and the jobs time of the current job by the completion time of the current operation.

Step 3.6: get the starting time and completion time of the plan (‘current operation’, ‘chosen machine’) by the current starting time and completion time respectively.

Step 4: return the plan

The plan is built on the assumption that the next operation will enter the machine immediately when the chosen machine is free.

### Compute fitness of individual genes

A schedule plan can be decoded from genes of an individual which is defined by genes and fitness. From the plan of a specific genes encoding, it is easy to get the completion time of all the jobs, the maximum workload of a machine, and the total workload of all machines. The fitness of individual genes is design with 3 attributes:

* The last time to finish all the jobs.
* The maximum workload of a machine.
* The total workload of all machines.

The fitness of an individual with the above three attributes is design with hierarchical structure to compare the fitness between individuals. The first priority is the last time to finish all the jobs. The second priority is the maximum workload of a machine. The third priority is the total workload of all machines. The smaller value of each fitness attribute the better the individual is.

The hierarchical attributes structure is design not only to help the GA find the shorted time to complete all the jobs but also help it to archive balance workload between machines. In reality, it is desirable to finish all the jobs as soon as possible, however the balance workload between machines should be considered.

To summary, each individual is design with genes and fitness. Genes carry the information about sequence decision and machine selection decision. From genes of an individual a plan can be derived. Fitness of an individual is design with the hierarchical structure. The structure is helpful to recognize better genes which present shorter time to finish all the jobs and balance workload between machines.

### Genetic operations

In GA literatures, there are two basic genetic operations. They are mutation and crossover. These genetic operations are inspired by the evolutionary process in the nature world. Mutation is the process of changing the material of genes in an individual. Crossover is the process of exchanging the genes material between two individuals to create new individuals. Through the evolution process and the pressure of natural selection, the better adapted individuals have more change to survive.

The mutation mechanism affects both sequence decision genes and machines selection genes of an individual. Mutation mechanism on sequence decision genes is processed by randomly selection two genes on the sequence decision genes then swap them. Since each gene is a job coding, and the decoding mechanism will decode genes to actual operation plan by reading through the ordered sequence of job on the sequence decision genes. Therefore, this mutation process will change the genes material but will not create any unvalid genes. On the other hand, mutation on machines selection genes is processed by randomly select an operation in the operations set, and choose the machine that can process that operation. By doing so, the mutation process on machine selection genes always create valid genes.

In Contrast with mutation, crossover is the process of exchange genes material between two individuals. Crossover operation on sequence decision genes is described by pseudocode flowing:

Step 1: get the sequence decision genes on the two parent individuals.

Step 2: cut the two sequence decision genes at the middle point.

Step 3: combine the head of the first genes with the tail of the second genes, and the head of the second genes with the tail of the first genes to create two new individuals.

Step 4: find missing or exceeding genes on the two new genes and fix them (since the operations set is known it is easy to find the missing and exceeding operation and then fix them).

Step 5: return the two new valid genes.

Crossover operation on machines selection genes is simpler than on sequence decision genes. Since the machine selection genes is reference to a fixed set of operations, therefore by cutting the two parent genes at the middle point and combining them in two new genes, the process will always create valid genes.

### Populated Genetic Algorithm

Each individual is design with genes and fitness value. By using hierarchical fitness structure, it is easy to compare the fitness value between individuals. Furthermore, individuals are put into an evolutionary environment which is defined by the proposed populated GA algorithm. The proposed populated GA has some parameters as flowing:

* Population size: the number of individuals in a generation. Each generation is an evolution loop of the proposed GA.
* Number of generation: the number of evolution loops. It is also the exist condition for the proposed populated GA.
* Crossover probability: the probability to apply crossover operation.
* Mutation probability: the probability to apply mutation operation.

The evolutionary process in the proposed populated GA algorithm is described as flowing pseudocode:

* Input: population size, mutate and crossover mechanism, the probability of mutation and crossover, number of generation.
* Output: the best individual.

Step 1: initialize the population randomly with given population size.

Step2: create an empty list to keep track of the best individuals through generations

Step 3: for n in the number of generation:

Step 3.1: randomly select individuals to apply mutation and crossover mechanism with the given probability.

Step 3.2: compare between the new individuals and their parents, choose the better individuals.

Step 3.3: put the better individuals back the population.

Step 3.4: find the best individual in the current generation.

Step 3.5 if the current best individual is better than the last best individuals, append to the list of best individuals.

Step 4: return the best individual from the best individuals list.

# Experiment and Discussion

In this section, the experiments are conducted to test on the validation of the proposed GA and the proposed approach for scheduling problem in brewery industry. First, the proposed GA algorithm is test on a bench mark flexible jobs shop scheduling problem to test its’ effectiveness. The second test case is conducted in a relative small real case. The purpose of this test is to validate the schedule result from the proposed approach. Finally, the proposed approach is tested in an industrial scale. Further experiments are conduct to explore the relation between different parameters in the propose approach. The result from these experiments can be used to recommend the parameter setting for the user of the proposed GA.

## Standard Test for the Proposed GA

In other to test the general performance of the proposed GA, the normal approach is to test with the benchmark problem. In this case the testing problem is flexible jobs shop scheduling with 8 jobs and 8 machines. The benchmark problem is proposed by Kacem et al. (2002) and Xia and Wu (2005). The other ways to testing the performance of the proposed GA is testing on a small problem that human can easily test the result.

### Testing on the Benchmark Problem

Table 11: The benchmark flexible jobs shop schedule with 8 jobs and 8 machines

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Jobs** | **Operations** | **M1** | **M2** | **M3** | **M4** | **M5** | **M6** | **M7** | **M8** |
| **J1** | **O11** | 5 | 3 | 5 | 3 | 3 |  | 10 | 9 |
|  | **O12** | 10 |  | 5 | 8 | 3 | 9 | 9 | 6 |
|  | **O13** |  | 10 |  | 5 | 6 | 2 | 4 | 5 |
| **J2** | **O21** | 5 | 7 | 3 | 9 | 8 |  | 9 |  |
|  | **O22** |  | 8 | 5 | 2 | 6 | 7 | 10 | 9 |
|  | **O23** |  | 10 |  | 5 | 6 | 4 | 1 | 7 |
|  | **O24** | 10 | 8 | 9 | 6 | 4 | 7 |  |  |
| **J3** | **O31** | 10 |  |  | 7 | 6 | 5 | 2 | 4 |
|  | **O32** |  | 10 | 6 | 4 | 8 | 9 | 10 |  |
|  | **O33** | 1 | 4 | 5 | 6 |  | 10 |  | 7 |
| **J4** | **O41** | 3 | 1 | 6 | 5 | 9 | 7 | 8 | 4 |
|  | **O42** | 12 | 11 | 7 | 8 | 10 | 5 | 6 | 9 |
|  | **O43** | 4 | 6 | 2 | 10 | 3 | 9 | 5 | 7 |
| **J5** | **O51** | 3 | 6 | 7 | 8 | 9 |  | 10 |  |
|  | **O52** | 10 |  | 7 | 4 | 9 | 8 | 6 |  |
|  | **O53** |  | 9 | 8 | 7 | 4 | 2 | 7 |  |
|  | **O54** | 11 | 9 |  | 6 | 7 | 5 | 3 | 6 |
| **J6** | **O61** | 6 | 7 | 1 | 4 | 6 | 9 |  | 10 |
|  | **O62** | 11 |  | 9 | 9 | 9 | 7 | 6 | 4 |
|  | **O63** | 10 | 5 | 9 | 10 | 11 |  | 10 |  |
| **J7** | **O71** | 5 | 4 | 2 | 6 | 7 |  | 10 |  |
|  | **O72** |  | 9 |  | 9 | 11 | 9 | 10 | 5 |
|  | **O73** |  | 8 | 9 | 3 | 8 | 6 |  | 10 |
| **J8** | **O81** | 2 | 8 | 5 | 9 |  | 4 |  | 10 |
|  | **O82** | 7 | 4 | 7 | 8 | 9 |  | 10 |  |
|  | **O83** | 9 | 9 |  | 8 | 5 | 6 | 7 | 1 |
|  | **O84** | 9 |  | 3 | 7 | 1 | 5 | 8 |  |

Table 11 shows a benchmark 8 jobs, 8 machines flexible job shop scheduling problem (Kacem et al. 2002). Each operation can be processed by a set of machines, if a machine can not process a given operation then that cell has no entry data.

Table 12: Testing Result on flexible jobs shop scheduling problem 8 jobs, 8 machines (10 first experiments only), GA parameters setting: population size = 100, generation = 6000, mutate probability = 0.9, crossover = 0.9

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Time** | **MaxWorkload** | **TotalWorkLoad** |
| **0** | 17 | 14 | 85 |
| **1** | 17 | 13 | 77 |
| **2** | 16 | 13 | 80 |
| **3** | 15 | 15 | 81 |
| **4** | 17 | 13 | 81 |
| **5** | 19 | 15 | 93 |
| **6** | 16 | 13 | 74 |
| **7** | 16 | 15 | 86 |
| **8** | 15 | 12 | 78 |
| **9** | 15 | 12 | 83 |
| **10** | 15 | 14 | 75 |

Table 13: 8 jobs, 8 machines testing summary

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Time** | **MaxWorkload** | **TotalWorkLoad** |
| **count** | 100 | 100 | 100 |
| **mean** | 16.01 | 13.53 | 82.32 |
| **std** | 1.366962 | 1.193035345 | 5.950774502 |
| **min** | 14 | 11 | 73 |
| **25%** | 15 | 13 | 78 |
| **50%** | 16 | 13 | 81 |
| **75%** | 17 | 14 | 86 |
| **max** | 20 | 17 | 103 |

The proposed GA is tested with the benchmark 8 jobs, 8 machine problem in 100 experiments. In each experiment, the result is measured in 3 measurements which include ‘Time’ (the time to finish all jobs), ‘MaxWorkload’ (the maximum processing time on a machine) and ‘TotalWorkload’ (the total processing time of all machines). The testing result is summarized in table 13. The mean of ‘Time’ measurement is equal to the best solution obtained by Kacem et al. (2002) and Xia and Wu (2005). The best solution obtained by the proposed GA is 14, equal to the best solution obtained by Gao et al. (2007). The proposed GA algorithm hits the best result in 9 times out of 100 experiments.

### Test on Simple Brewery Production Problem.

In this section, the correctness of the proposed approach is testing with a small real case problem. The real case testing problem is relative small, but the data input is based on the real data collect from Sabeco website. The goal of this test case is to verify the correctness of the proposed algorithm.

Table 14: Input Data for testing the proposed algorithm

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Product Code** | **Product Name** | **Volume/bottle** | **Demand (ml)** | **Beer Type** | **Fermentation Time (date)** |
| **B1** | 333 Premium (bottle) | 330ml | - | premium lager | 28 |
| **B2** | 333 (can) | 330ml | - | lager | 21 |
| **B3** | Saigon Lager (bottle) | 450ml | 175,500,000 | lager | 21 |
| **B4** | Saigon Lager (can) | 330ml | 58,500,000 | lager | 21 |
| **B5** | Saigon Export (bottle) | 355ml | - | lager | 28 |
| **B6** | Saigon Special (bottle) | 330ml | 175,500,000 | lager | 28 |
| **B7** | Saigon Special (can) | 330ml | - | lager | 28 |

The table 14 describes the input data for the testing problem, machines configuration and machines capacity is the same as the data configuration stage. The Demand is set to be small in order to create a relative simple plan in which it is easy to verify the validation of the result schedule plan.

Table 15: Assignment result for testing the proposed algorithm n = 0.4

|  |  |  |
| --- | --- | --- |
| **Tank Type** | **Product** | **Assignment** |
| TA | B3 | 1 |
| TA | B4 | 1 |
| TA | B6 | 1 |
| TB | B3 | 1 |
| TB | B4 | 0 |
| TB | B6 | 1 |

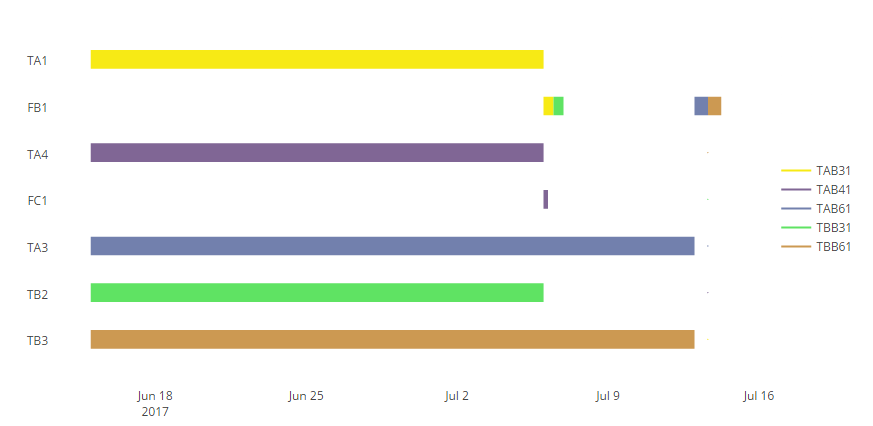


Figure 19: Production plan for testing the proposed algorithm in Gantt Chart format

Table 15 shows tank types assignment result for the testing problem. The actual scheduled plan from the proposed algorithm is displayed in figure 23. In figure 19, vertical axis shows chosen machine, horizontal axis shows the time line. Jobs are display on the right side of figure 23. Operations is displayed by color bars, operations are in the same job will have the same color. Figure 19 shows a perfectly valid schedule plan. The sequence constraint of operations in the same job is reserved. In this case filling stage can only take place when fermentation stage is finished. A machine can only process one operation as a time, and a machine can only process a set of operations base on their function. In this case, the bottle filling line can only fill bottle products, the can filling line can only fill can products. For example, in the job ‘TAB31’ (product B3 is fermented in tank type TA), the fermentation process of product B3 (Saigon Lager - bottle) will take place in tank ‘TA1’ after that the filling line ‘FB1’ (the bottle filling line) will process the filling operation of job ‘TAB31’. Figure 23 shows that it is safe to conclude that the proposed algorithm work well and create good and valid schedule solution.

## The proposed Approach with Industrial Production Scale.

In this section, the proposed approach is tested with large demand which is close to the problem face by the brewery company like Sabeco.

Table 16: Input data, large scale production

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Product Code** | **Product Name** | **Volume/bottle** | **Demand (ml)** | **Beer Type** | **Fermentation Time (day)** |
| **B1** | 333 premium (bottle) | 330ml | - | premium lager | 28 |
| **B2** | 333 (can) | 330ml | 30,000,000,000 | lager | 21 |
| **B3** | saigon lager (bottle) | 450ml | 125,000,000,000 | lager | 21 |
| **B4** | saigon lager (can) | 330ml | 100,000,000,000 | lager | 21 |
| **B5** | saigon export (bottle) | 355ml | 10,000,000,000 | lager | 28 |
| **B6** | saigon special (bottle) | 330ml | 175,500,000 | lager | 28 |
| **B7** | saigon special (can) | 330ml | 30,000,000,000 | lager | 28 |

### Synchronization between the Two Production Stages

In the attempt to optimize the whole production process, this thesis tries different tanks types assignment configuration combining with GA for flexible jobs shop scheduling problem. A scenario analysis is conduct as flowing:

Table 17: Scenario analysis for synchronization (population = 100, generation = 2000, mutate = 0.3, crossover = 0.7)

|  |  |
| --- | --- |
| **n** | **Finish** |
| 0.1 | 86.38 |
| 0.2 | 79.45 |
| 0.3 | 82.14 |
| 0.4 | 80.22 |
| 0.5 | 80.63 |
| 0.6 | 93.27 |
| 0.7 | 95.21 |
| 0.8 | 106.25 |
| 0.9 | 105.31 |
| 1 | 105.33 |

Table 17 shows that, tank types assignment problem with n = 0.2 will work best with GA for flexible jobs shop scheduling problem. The total time to finish all the demand is 79.45 days with the configuration n = 0.2. It is noticeable that the tank types assignment definitely affects the whole scheduling process. The production process becomes more effective when the tasks are assigned equally between different types of resource. In other words, this kind of scenario analysis helps planner find the configuration in which use the available resource effectively.

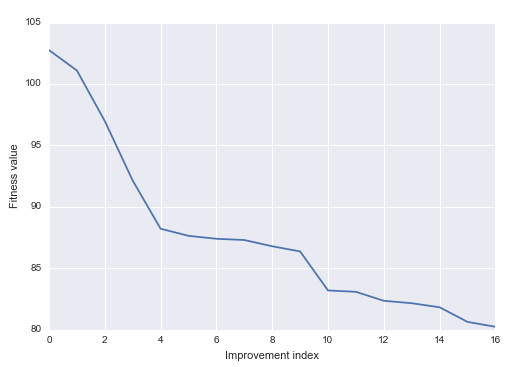


Figure 20: The GA learning curve (n = 0.4, population size = 100, generation number = 2000, mutate = 0.3, crossover = 0.7)

In figure 20, the GA learning curve displays the learning process of the GA algorithm. At first, the total time need to finish all demand is more than 100 days. Through the evolution process, the result scheduling plan become better and better. Through 2000 generations, the total time to process to all demand is 80.22 days.

## Plan Presentation and Validation

As industrial production scale, the schedule plan from the proposed algorithm becomes relatively complex. It is necessary to find different way to present the schedule plan which help the user easy to use and validate the schedule plan.

Table 18: Plan presentation in job and operation perspective, a snapshot

|  |  |  |  |
| --- | --- | --- | --- |
| **Operation** | **Machine** | **Start** | **Finish** |
| **TAB211** | **TA4** | 0 | 21 |
| **TAB212** | **FC1** | 21 | 22.25 |
| **TAB221** | **TA5** | 21 | 42 |
| **TAB222** | **FC1** | 72.5 | 73.75 |
| **TAB311** | **TA5** | 42 | 63 |
| **TAB312** | **FB1** | 65.03 | 68.32 |
| **TAB321** | **TA2** | 0 | 21 |
| **TAB322** | **FB1** | 48.58 | 51.87 |
| **TAB331** | **TA1** | 21 | 42 |
| **TAB332** | **FB1** | 42 | 45.29 |
| **TAB341** | **TA2** | 21 | 42 |
| **TAB342** | **FB1** | 58.45 | 61.74 |

Table 19: Plan presentation in machine and time perspective, a snapshot

|  |  |  |  |
| --- | --- | --- | --- |
| **Operation** | **Machine** | **Start** | **Finish** |
| **TBB362** | **FB1** | 21 | 24.29 |
| **TBB352** | **FB1** | 24.29 | 27.58 |
| **TAB332** | **FB1** | 42 | 45.29 |
| **TBB392** | **FB1** | 45.29 | 48.58 |
| **TAB322** | **FB1** | 48.58 | 51.87 |
| **TBB312** | **FB1** | 51.87 | 55.16 |
| **TBB372** | **FB1** | 55.16 | 58.45 |
| **TAB342** | **FB1** | 58.45 | 61.74 |
| **TBB382** | **FB1** | 61.74 | 65.03 |
| **TAB312** | **FB1** | 65.03 | 68.32 |
| **TBB332** | **FB1** | 68.32 | 71.61 |
| **TBB322** | **FB1** | 71.61 | 74.9 |
| **TBB342** | **FB1** | 74.9 | 78.19 |
| **TAB512** | **FB1** | 78.19 | 78.52 |

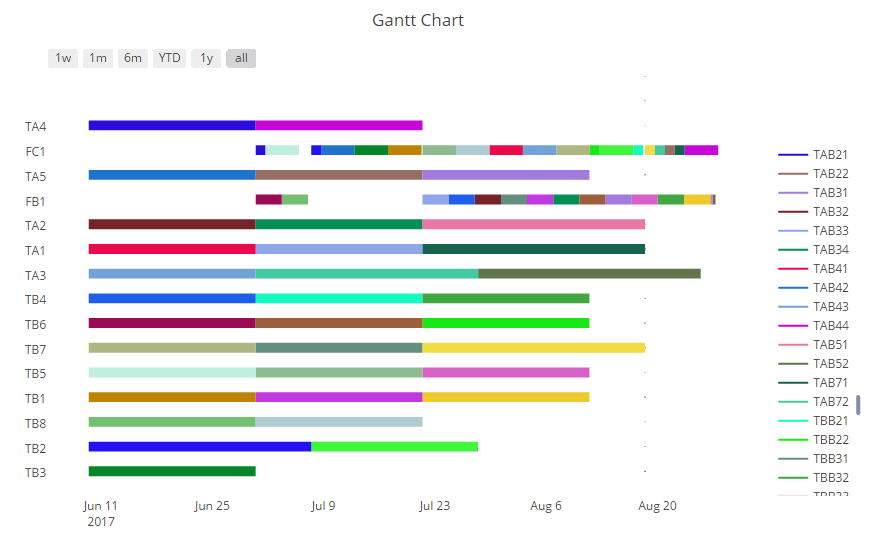


Figure 21: Plan present in Gantt Chart format

By present the result plan in different perspective, it is easier for testing the validation of the plan. In Table 18, the resulting plan is sorted in term of jobs and operations. It helps us check the sequencing constraint of operation in the same job, that is the flowing operations have to be later the precedence operations of the same job. In this case the filling operations have to be later than the fermentation operations. In Table 19, the result plant is sorted in term of machines and times. It helps us the check the validation of job processing, that is a machine can only process one job as a time (no overlapping time between operations in one machine). In figure 21, the whole resulting plant is presented in Gantt Chart format, which helps us visualize the hold plan. The color label is index by jobs at the right side of the Gantt Chart, operations of a same job will display with the same color.

## Relation between population size, number of generations and fitness

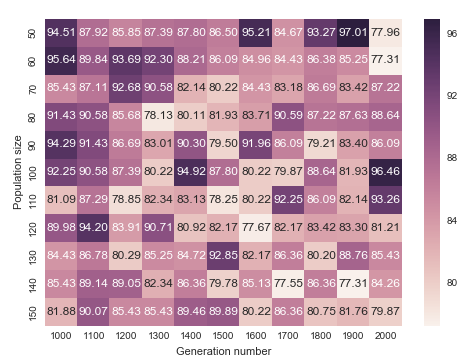


Figure 22: Relation between population size, number of generations and fitness (n = 0.4 mutate = 0.3, crossover = 0.7)

As expected, the general trend is that as the population size and the number of generation increase the schedule plan getting better. In this experiment, the recommended configuration is population size ≥ 120, number of generation ≥ 1600. It is noticeable that the configuration of population size = 50, number of generation = 2000 and the configuration of population size = 60, number of generation = 2000 can also give good results.

## The Relation between Mutate, Crossover and Evolution Process within the Proposed GA

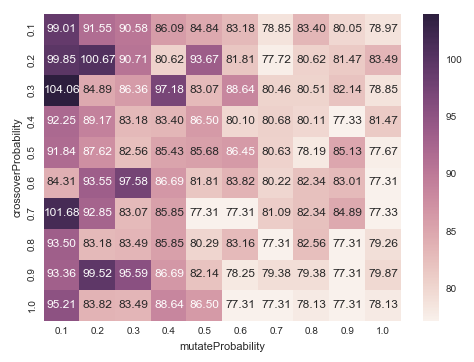


Figure 23: Relation between crossover, mutation probability and fitness (n = 0.4, population size = 50, generation = 1500)

The figure 23 presents the relation between crossover probability, mutation probability and fitness within the proposed GA algorithm. It is noticeable that there is no clear pattern about the effect of crossover probability to the performance of the algorithm. However, the overall fitness becomes better with the configuration of crossover probability equal or larger than 0.4. There is a clear pattern about the effect of mutate probability to the fitness value. The proposed GA will work better with if the mutate probability is set equal or larger than 0.7. Since the right corner of the figure shows good fitness values, the overall recommended configuration both parameters is crossover probability equal or larger than 0.8, mutation probability equal or larger than 0.7.

# Conclusion

This thesis is evolving from the fact that brewery industry is a huge industry in Vietnam and Vietnamese brewery manufacturers still do not a scheduling function to help schedule their production process. One example for this situation is Sabeco Corporation. Even though Sabeco is the largest beer producer in Vietnam, their scheduling activity is still done by experience and do not have any support system or tool. The goal of this thesis is try to build a scheduling function for a Vietnamese brewery company like Sabeco. The propose approach have to take into account the characteristic of Vietnamese brewery company like Sabeco as well as the characteristic of the brewery production process. The assumption for the proposed approach is listed as flowing:

* The fermentation time of a product is fixed and unrelated with the volume of production, while the bottle filling time is related with the production volume and filling machines capacity in a linear fashion. Therefore, the operation time of fermentation and filling should be compute differently.
* The sequence constraint of operations in the same job is reserved. The filling operation can only process when the fermentation operation is finished.
* A machine can only process an operation as a time. A machine can only process a set of operations base on their function. For example, the bottle filling line can only fill bottle products, the can filling line can only fill can products.
* A job will enter the machine immediately when the previous job is finished.

## New Contributions

In this thesis, the scheduling problem is formulated as flexible jobs shop scheduling problem. In order to take into account the characteristic of the brewery production process a simple math programming model is introduced as a job assignment to tanks. In order to deploy the proposed GA for synchronization between fermentation and filling stages, the result from assignment model need to be processed in order to form processing time table (Table 8). The processing time table is the key input for the proposed GA. Through the evolution process of the proposed GA, the schedule plan become better and better.

Moreover, three different presentations of the schedule plan are introduced to validate the schedule result. These different presentations of the schedule plan can help the user understand the complex schedule result easily, quickly and in different perspective. The schedule plan will become easier to use with these supporting report. Several experiments are conduct to investigate the relation between population size and number of generations, the relation between crossover probability and mutation probability. These experiments explore different parameters setting and help to find a good configuration within the proposed algorithm.

## Limitations and Further Research

The approach proposes by this thesis has two major limitations. Firstly, the tanks assignment and synchronization between fermentation and filling stages are done separately. In order to find a good combination between tanks assignment and synchronization, it is necessary to run the scenario analysis to find a good configuration. In other words, the scenario analysis is needed to find an appropriate value of n in the equation (12). Secondly, the proposed approach is based on flexible jobs shop scheduling problem, therefore the processing time of a given operation on a chosen machine is fixed. However, in the real industrial production case, the liquid in fermentation stage can state in the tank longer than their fermentation time waiting for filling in bottles.

For further research, these two drawback need to be addressed. In other to address these two problems, several questions are needed to be answered. Firstly, How to model the fermentation waiting time as real as possible. The fermentation time can be flexible since the liquid can stay in tanks more than the requirement fermentation time. Secondly, the proposed approach can be further improved by finding a way to optimize the two production stages simultaneously.

References

1. Adams J, Balas E, Zawack D. (1988). The Shifting Bottleneck Procedure for Job Shop Scheduling. Management Science. Vol 34(3), pp. 391-401.
2. Amin-Naseri MR, Afshari AJ. (2012). A hybrid genetic algorithm for integrated process planning and scheduling problem with precedence constraints. The International Journal of Advanced Manufacturing Technology. Vol 59(1), pp. 273-287.
3. Baldo TA, Santos MO, Almada-Lobo B, Morabito R. (2014). An optimization approach for the lot sizing and scheduling problem in the brewery industry. Computers & Industrial Engineering. Vol 72, pp. 58-71.
4. Chang Wook A, Ramakrishna RS. (2002). A genetic algorithm for shortest path routing problem and the sizing of populations. IEEE Transactions on Evolutionary Computation. Vol 6(6), pp. 566-579.
5. Cheng R, Gen M, Sasaki M. (1995). Film-copy deliverer problem using genetic algorithms. Computers & Industrial Engineering. Vol 29(1), pp. 549-553.
6. Cheng R, Gen M, Tsujimura Y. (1996). A tutorial survey of job-shop scheduling problems using genetic algorithms—I. representation. Computers & Industrial Engineering. Vol 30(4), pp. 983-997.
7. Ferreira D, Clark AR, Almada-Lobo B, Morabito R. (2012). Single-stage formulations for synchronised two-stage lot sizing and scheduling in soft drink production. International Journal of Production Economics. Vol 136(2), pp. 255-265.
8. Ferreira D, Morabito R, Rangel S. (2009). Solution approaches for the soft drink integrated production lot sizing and scheduling problem. European Journal of Operational Research. Vol 196(2), pp. 697-706.
9. Fleischmann B, Meyr H. (1997). The general lotsizing and scheduling problem. Operations-Research-Spektrum. Vol 19(1), pp. 11-21.
10. Gao J, Gen M, Sun L, Zhao X. (2007). A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems. Computers & Industrial Engineering. Vol 53(1), pp. 149-162.
11. Gen M, Gao J, Lin L. Multistage-Based Genetic Algorithm for Flexible Job-Shop Scheduling Problem. In: Gen M, Green D, Katai O, McKay B, Namatame A, Sarker RA, et al., (Eds.) Intelligent and Evolutionary Systems. Berlin, Heidelberg: Springer Berlin Heidelberg; (2009), pp. 183-196.
12. Haupt R. (1989). A survey of priority rule-based scheduling. Operations-Research-Spektrum. Vol 11(1), pp. 3-16.
13. Inagaki J, Haseyama M, Kitajima H. (1999). A genetic algorithm for determining multiple routes and its applications. Circuits and Systems, 1999 ISCAS '99 Proceedings of the 1999 IEEE International Symposium on. Vol 6, pp. 137-140.
14. Kacem I, Hammadi S, Borne P. (2002). Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews). Vol 32(1), pp. 1-13.
15. Kundakci N, Kulak O. (2016). Hybrid genetic algorithms for minimizing makespan in dynamic job shop scheduling problem. Computers & Industrial Engineering. Vol 96, pp. 31-51.
16. Leu G, Namatame A. Evolving Failure Resilience in Scale-free Networks. Intelligent and Evolutionary Systems (2008), pp. 49-59.
17. Lin L, Gen M. (2009). Auto-tuning strategy for evolutionary algorithms: balancing between exploration and exploitation. Soft Computing. Vol 13(2), pp. 157-168.
18. Mitsuo G, Runwei C, Lin L. Network Models and Optimization: Multiobjective Genetic Algorithm Approach. 1st ed. London: Springer-Verlag London; (2008).
19. Orito Y, Jun T, Takeda M, Yamamoto H. Index Fund Optimization Using Genetic Algorithm and Scatter Diagram Based on Coefficients of Determination. Intelligent and Evolutionary Systems (2009), pp. 1-11.
20. Pezzella F, Morganti G, Ciaschetti G. (2008). A genetic algorithm for the Flexible Job-shop Scheduling Problem. Computers & Operations Research. Vol 35(10), pp. 3202-3212.
21. Toledo CFM, de Oliveira L, Pereira RD, Franca PM, Morabito R. (2014). A genetic algorithm/mathematical programming approach to solve a two-level soft drink production problem. Computers & Operations Research. Vol 48, pp. 40-52.
22. Xia H, Li X, Gao L. (2016). A hybrid genetic algorithm with variable neighborhood search for dynamic integrated process planning and scheduling. Computers & Industrial Engineering. Vol 102, pp. 99-112.
23. Xia W, Wu Z. (2005). An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. Computers & Industrial Engineering. Vol 48(2), pp. 409-425.
24. Zhang H, Gen M, Seo Y. (2006). An effective coding approach for multiobjective integrated resource selection and operation sequences problem. Journal of Intelligent Manufacturing. Vol 17(4), pp. 385-397.

Appendices

A Sequence Decision Genes for Industrial Case











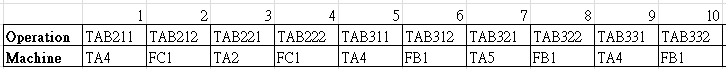


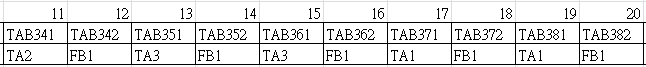


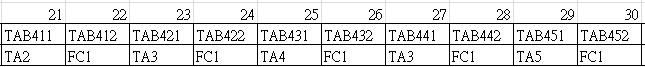


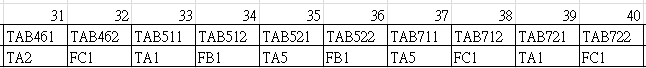
Note: the integer index numbers from 1 to 74 is just for the purpose of presentation. It helps reader easier to keep track of the sequence decision genes. The sequence decision genes is presented as a list of job coding and is read from left to right.

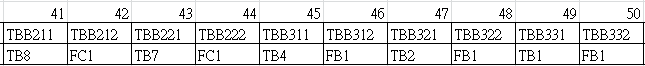
Machine Selection Genes Presentation

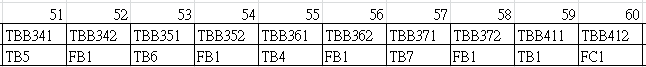


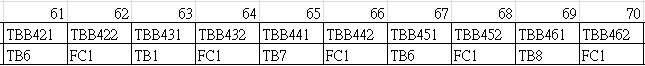


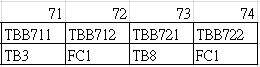












Note: the integer index numbers are just for the presentation purpose. It helps the reader easier to keep track of the machine selection genes. Machine selection genes are mapped to a fixed list of operations. Each operation coding has the information about the operation itself, therefore we reference to a set of machines that can process of that operation.

Processing Time Table for Simple Case

|  |  |
| --- | --- |
| **Operation** | **Processing Time** |
| ('TAB611', 'TA4') | 28 |
| ('TAB411', 'TA3') | 21 |
| ('TBB611', 'TB3') | 28 |
| ('TBB311', 'TB5') | 21 |
| ('TAB411', 'TA2') | 21 |
| ('TBB312', 'FB1') | 0.46 |
| ('TAB312', 'FB1') | 0.46 |
| ('TAB311', 'TA1') | 21 |
| ('TBB311', 'TB2') | 21 |
| ('TAB411', 'TA1') | 21 |
| ('TAB412', 'FC1') | 0.21 |
| ('TBB611', 'TB8') | 28 |
| ('TBB611', 'TB7') | 28 |
| ('TAB311', 'TA2') | 21 |
| ('TAB611', 'TA5') | 28 |
| ('TAB411', 'TA5') | 21 |
| ('TAB311', 'TA3') | 21 |
| ('TBB612', 'FB1') | 0.62 |
| ('TBB611', 'TB1') | 28 |
| ('TAB611', 'TA1') | 28 |
| ('TBB611', 'TB5') | 28 |
| ('TBB311', 'TB1') | 21 |
| ('TAB611', 'TA3') | 28 |
| ('TAB311', 'TA4') | 21 |
| ('TBB611', 'TB4') | 28 |
| ('TAB411', 'TA4') | 21 |
| ('TAB611', 'TA2') | 28 |
| ('TBB311', 'TB7') | 21 |
| ('TBB311', 'TB8') | 21 |
| ('TBB311', 'TB6') | 21 |
| ('TAB612', 'FB1') | 0.62 |
| ('TBB311', 'TB3') | 21 |
| ('TBB611', 'TB2') | 28 |
| ('TBB611', 'TB6') | 28 |
| ('TAB311', 'TA5') | 21 |
| ('TBB311', 'TB4') | 21 |

Processing Time Table for Industrial Scale Case

|  |  |
| --- | --- |
| **Operation** | **Processing Time** |
| ('TBB471', 'TB6') | 21 |
| ('TBB361', 'TB3') | 21 |
| ('TAB351', 'TA5') | 21 |
| ('TBB431', 'TB3') | 21 |
| ('TBB411', 'TB3') | 21 |
| ('TBB381', 'TB5') | 21 |
| ('TAB221', 'TA5') | 21 |
| ('TAB341', 'TA1') | 21 |
| ('TBB471', 'TB4') | 21 |
| ('TAB311', 'TA3') | 21 |
| ('TAB311', 'TA5') | 21 |
| ('TBB431', 'TB6') | 21 |
| ('TBB471', 'TB3') | 21 |
| ('TBB372', 'FB1') | 3.29 |
| ('TBB361', 'TB2') | 21 |
| ('TBB351', 'TB4') | 21 |
| ('TBB411', 'TB8') | 21 |
| ('TBB361', 'TB6') | 21 |
| ('TBB351', 'TB5') | 21 |
| ('TBB391', 'TB5') | 21 |
| ('TAB332', 'FB1') | 3.29 |
| ('TBB451', 'TB4') | 21 |
| ('TBB341', 'TB6') | 21 |
| ('TBB221', 'TB5') | 21 |
| ('TBB472', 'FC1') | 4.21 |
| ('TBB361', 'TB7') | 21 |
| ('TBB341', 'TB1') | 21 |
| ('TAB431', 'TA4') | 21 |
| ('TBB361', 'TB8') | 21 |
| ('TBB471', 'TB1') | 21 |
| ('TBB441', 'TB6') | 21 |
| ('TBB321', 'TB7') | 21 |
| ('TBB721', 'TB4') | 28 |
| ('TBB381', 'TB1') | 21 |
| ('TBB441', 'TB1') | 21 |
| ('TBB451', 'TB5') | 21 |
| ('TAB432', 'FC1') | 4.21 |
| ('TAB522', 'FB1') | 0.33 |
| ('TBB362', 'FB1') | 3.29 |
| ('TBB371', 'TB8') | 21 |
| ('TBB221', 'TB4') | 21 |
| ('TBB212', 'FC1') | 1.25 |
| ('TBB341', 'TB7') | 21 |
| ('TBB211', 'TB1') | 21 |
| ('TAB711', 'TA2') | 28 |
| ('TAB341', 'TA3') | 21 |
| ('TAB721', 'TA5') | 28 |
| ('TAB521', 'TA3') | 28 |
| ('TBB461', 'TB1') | 21 |
| ('TAB221', 'TA4') | 21 |
| ('TAB521', 'TA5') | 28 |
| ('TBB461', 'TB6') | 21 |
| ('TAB721', 'TA3') | 28 |
| ('TBB451', 'TB3') | 21 |
| ('TBB331', 'TB7') | 21 |
| ('TBB312', 'FB1') | 3.29 |
| ('TAB711', 'TA3') | 28 |
| ('TAB212', 'FC1') | 1.25 |
| ('TBB311', 'TB4') | 21 |
| ('TBB441', 'TB5') | 21 |
| ('TBB371', 'TB6') | 21 |
| ('TAB422', 'FC1') | 4.21 |
| ('TAB711', 'TA1') | 28 |
| ('TBB411', 'TB7') | 21 |
| ('TAB342', 'FB1') | 3.29 |
| ('TAB322', 'FB1') | 3.29 |
| ('TAB341', 'TA5') | 21 |
| ('TBB311', 'TB7') | 21 |
| ('TBB311', 'TB6') | 21 |
| ('TBB211', 'TB5') | 21 |
| ('TBB371', 'TB4') | 21 |
| ('TAB211', 'TA3') | 21 |
| ('TBB341', 'TB5') | 21 |
| ('TAB431', 'TA5') | 21 |
| ('TBB391', 'TB1') | 21 |
| ('TBB421', 'TB5') | 21 |
| ('TAB421', 'TA2') | 21 |
| ('TBB441', 'TB4') | 21 |
| ('TBB341', 'TB8') | 21 |
| ('TBB711', 'TB6') | 28 |
| ('TBB381', 'TB4') | 21 |
| ('TBB442', 'FC1') | 4.21 |
| ('TBB211', 'TB2') | 21 |
| ('TBB381', 'TB6') | 21 |
| ('TBB331', 'TB5') | 21 |
| ('TBB351', 'TB2') | 21 |
| ('TBB421', 'TB2') | 21 |
| ('TBB341', 'TB4') | 21 |
| ('TBB351', 'TB3') | 21 |
| ('TAB721', 'TA4') | 28 |
| ('TAB211', 'TA5') | 21 |
| ('TAB711', 'TA4') | 28 |
| ('TBB721', 'TB6') | 28 |
| ('TAB331', 'TA3') | 21 |
| ('TAB331', 'TA4') | 21 |
| ('TAB341', 'TA2') | 21 |
| ('TBB351', 'TB7') | 21 |
| ('TBB221', 'TB3') | 21 |
| ('TAB441', 'TA5') | 21 |
| ('TAB511', 'TA1') | 28 |
| ('TBB411', 'TB1') | 21 |
| ('TBB391', 'TB4') | 21 |
| ('TBB411', 'TB4') | 21 |
| ('TBB431', 'TB2') | 21 |
| ('TAB211', 'TA1') | 21 |
| ('TAB351', 'TA2') | 21 |
| ('TBB471', 'TB5') | 21 |
| ('TBB711', 'TB2') | 28 |
| ('TBB321', 'TB5') | 21 |
| ('TAB711', 'TA5') | 28 |
| ('TBB311', 'TB3') | 21 |
| ('TBB351', 'TB6') | 21 |
| ('TAB512', 'FB1') | 0.33 |
| ('TAB511', 'TA4') | 28 |
| ('TBB382', 'FB1') | 3.29 |
| ('TBB431', 'TB5') | 21 |
| ('TBB221', 'TB8') | 21 |
| ('TBB321', 'TB1') | 21 |
| ('TBB462', 'FC1') | 4.21 |
| ('TAB521', 'TA1') | 28 |
| ('TBB421', 'TB3') | 21 |
| ('TBB391', 'TB7') | 21 |
| ('TAB431', 'TA1') | 21 |
| ('TBB392', 'FB1') | 3.29 |
| ('TBB322', 'FB1') | 3.29 |
| ('TAB411', 'TA2') | 21 |
| ('TBB391', 'TB6') | 21 |
| ('TAB351', 'TA1') | 21 |
| ('TAB211', 'TA2') | 21 |
| ('TBB361', 'TB5') | 21 |
| ('TBB722', 'FC1') | 1.25 |
| ('TBB321', 'TB3') | 21 |
| ('TBB451', 'TB8') | 21 |
| ('TBB451', 'TB2') | 21 |
| ('TAB431', 'TA2') | 21 |
| ('TBB311', 'TB8') | 21 |
| ('TBB431', 'TB1') | 21 |
| ('TAB311', 'TA2') | 21 |
| ('TAB441', 'TA3') | 21 |
| ('TBB451', 'TB7') | 21 |
| ('TBB461', 'TB2') | 21 |
| ('TAB321', 'TA2') | 21 |
| ('TAB411', 'TA5') | 21 |
| ('TBB331', 'TB1') | 21 |
| ('TBB411', 'TB6') | 21 |
| ('TBB321', 'TB6') | 21 |
| ('TBB452', 'FC1') | 4.21 |
| ('TBB412', 'FC1') | 4.21 |
| ('TBB422', 'FC1') | 4.21 |
| ('TBB222', 'FC1') | 1.25 |
| ('TAB411', 'TA4') | 21 |
| ('TBB371', 'TB1') | 21 |
| ('TBB371', 'TB2') | 21 |
| ('TBB381', 'TB3') | 21 |
| ('TBB351', 'TB8') | 21 |
| ('TBB711', 'TB1') | 28 |
| ('TAB722', 'FC1') | 1.25 |
| ('TBB321', 'TB2') | 21 |
| ('TBB421', 'TB1') | 21 |
| ('TAB421', 'TA4') | 21 |
| ('TAB221', 'TA2') | 21 |
| ('TBB391', 'TB2') | 21 |
| ('TAB321', 'TA3') | 21 |
| ('TBB421', 'TB6') | 21 |
| ('TBB342', 'FB1') | 3.29 |
| ('TAB511', 'TA2') | 28 |
| ('TAB221', 'TA1') | 21 |
| ('TAB511', 'TA3') | 28 |
| ('TBB361', 'TB4') | 21 |
| ('TBB711', 'TB3') | 28 |
| ('TBB351', 'TB1') | 21 |
| ('TBB211', 'TB6') | 21 |
| ('TAB331', 'TA1') | 21 |
| ('TBB331', 'TB3') | 21 |
| ('TBB432', 'FC1') | 4.21 |
| ('TAB331', 'TA5') | 21 |
| ('TBB711', 'TB8') | 28 |
| ('TBB461', 'TB4') | 21 |
| ('TBB711', 'TB5') | 28 |
| ('TBB211', 'TB3') | 21 |
| ('TBB221', 'TB2') | 21 |
| ('TBB461', 'TB3') | 21 |
| ('TBB721', 'TB5') | 28 |
| ('TBB332', 'FB1') | 3.29 |
| ('TBB471', 'TB7') | 21 |
| ('TBB311', 'TB2') | 21 |
| ('TAB421', 'TA3') | 21 |
| ('TBB721', 'TB2') | 28 |
| ('TBB221', 'TB6') | 21 |
| ('TBB431', 'TB8') | 21 |
| ('TAB431', 'TA3') | 21 |
| ('TAB351', 'TA4') | 21 |
| ('TAB352', 'FB1') | 3.29 |
| ('TBB461', 'TB7') | 21 |
| ('TBB431', 'TB7') | 21 |
| ('TBB331', 'TB8') | 21 |
| ('TAB511', 'TA5') | 28 |
| ('TAB312', 'FB1') | 3.29 |
| ('TBB341', 'TB2') | 21 |
| ('TAB721', 'TA1') | 28 |
| ('TBB471', 'TB8') | 21 |
| ('TAB712', 'FC1') | 1.25 |
| ('TAB221', 'TA3') | 21 |
| ('TBB381', 'TB2') | 21 |
| ('TBB421', 'TB8') | 21 |
| ('TAB341', 'TA4') | 21 |
| ('TBB441', 'TB2') | 21 |
| ('TBB712', 'FC1') | 1.25 |
| ('TAB311', 'TA4') | 21 |
| ('TBB721', 'TB7') | 28 |
| ('TAB321', 'TA5') | 21 |
| ('TAB311', 'TA1') | 21 |
| ('TBB361', 'TB1') | 21 |
| ('TBB211', 'TB4') | 21 |
| ('TBB371', 'TB5') | 21 |
| ('TBB221', 'TB1') | 21 |
| ('TBB461', 'TB8') | 21 |
| ('TBB321', 'TB8') | 21 |
| ('TAB442', 'FC1') | 4.21 |
| ('TAB521', 'TA2') | 28 |
| ('TBB391', 'TB3') | 21 |
| ('TBB421', 'TB4') | 21 |
| ('TBB311', 'TB5') | 21 |
| ('TBB391', 'TB8') | 21 |
| ('TBB721', 'TB1') | 28 |
| ('TBB371', 'TB3') | 21 |
| ('TBB461', 'TB5') | 21 |
| ('TBB711', 'TB4') | 28 |
| ('TAB351', 'TA3') | 21 |
| ('TAB321', 'TA1') | 21 |
| ('TBB411', 'TB2') | 21 |
| ('TBB451', 'TB6') | 21 |
| ('TBB331', 'TB4') | 21 |
| ('TAB721', 'TA2') | 28 |
| ('TBB451', 'TB1') | 21 |
| ('TBB431', 'TB4') | 21 |
| ('TAB521', 'TA4') | 28 |
| ('TAB421', 'TA5') | 21 |
| ('TAB441', 'TA4') | 21 |
| ('TBB721', 'TB8') | 28 |
| ('TBB711', 'TB7') | 28 |
| ('TAB441', 'TA1') | 21 |
| ('TBB352', 'FB1') | 3.29 |
| ('TBB331', 'TB2') | 21 |
| ('TAB421', 'TA1') | 21 |
| ('TAB412', 'FC1') | 4.21 |
| ('TBB441', 'TB3') | 21 |
| ('TBB371', 'TB7') | 21 |
| ('TBB331', 'TB6') | 21 |
| ('TBB441', 'TB7') | 21 |
| ('TBB381', 'TB8') | 21 |
| ('TAB222', 'FC1') | 1.25 |
| ('TBB381', 'TB7') | 21 |
| ('TAB331', 'TA2') | 21 |
| ('TBB411', 'TB5') | 21 |
| ('TAB321', 'TA4') | 21 |
| ('TBB211', 'TB7') | 21 |
| ('TBB721', 'TB3') | 28 |
| ('TBB311', 'TB1') | 21 |
| ('TBB321', 'TB4') | 21 |
| ('TBB441', 'TB8') | 21 |
| ('TBB341', 'TB3') | 21 |
| ('TBB211', 'TB8') | 21 |
| ('TBB471', 'TB2') | 21 |
| ('TAB441', 'TA2') | 21 |
| ('TAB411', 'TA3') | 21 |
| ('TAB211', 'TA4') | 21 |
| ('TBB421', 'TB7') | 21 |
| ('TBB221', 'TB7') | 21 |
| ('TAB411', 'TA1') | 21 |

Schedule Plan for Industrial Scale with Jobs and Operation Aspect

|  |  |  |  |
| --- | --- | --- | --- |
| **Operation** | **Machine** | **Start** | **Finish** |
| TAB211 | TA4 | 0 | 21 |
| TAB212 | FC1 | 21 | 22.25 |
| TAB221 | TA5 | 21 | 42 |
| TAB222 | FC1 | 72.5 | 73.75 |
| TAB311 | TA5 | 42 | 63 |
| TAB312 | FB1 | 65.03 | 68.32 |
| TAB321 | TA2 | 0 | 21 |
| TAB322 | FB1 | 48.58 | 51.87 |
| TAB331 | TA1 | 21 | 42 |
| TAB332 | FB1 | 42 | 45.29 |
| TAB341 | TA2 | 21 | 42 |
| TAB342 | FB1 | 58.45 | 61.74 |
| TAB411 | TA1 | 0 | 21 |
| TAB412 | FC1 | 50.42 | 54.63 |
| TAB421 | TA5 | 0 | 21 |
| TAB422 | FC1 | 29.25 | 33.46 |
| TAB431 | TA3 | 0 | 21 |
| TAB432 | FC1 | 54.63 | 58.84 |
| TAB441 | TA4 | 21 | 42 |
| TAB442 | FC1 | 75 | 79.21 |
| TAB511 | TA2 | 42 | 70 |
| TAB512 | FB1 | 78.19 | 78.52 |
| TAB521 | TA3 | 49 | 77 |
| TAB522 | FB1 | 78.52 | 78.85 |
| TAB711 | TA1 | 42 | 70 |
| TAB712 | FC1 | 73.75 | 75 |
| TAB721 | TA3 | 21 | 49 |
| TAB722 | FC1 | 71.25 | 72.5 |
| TBB211 | TB4 | 21 | 42 |
| TBB212 | FC1 | 68.51 | 69.76 |
| TBB221 | TB6 | 42 | 63 |
| TBB222 | FC1 | 63.05 | 64.3 |
| TBB311 | TB7 | 21 | 42 |
| TBB312 | FB1 | 51.87 | 55.16 |
| TBB321 | TB4 | 42 | 63 |
| TBB322 | FB1 | 71.61 | 74.9 |
| TBB331 | TB5 | 42 | 63 |
| TBB332 | FB1 | 68.32 | 71.61 |
| TBB341 | TB1 | 42 | 63 |
| TBB342 | FB1 | 74.9 | 78.19 |
| TBB351 | TB8 | 0 | 21 |
| TBB352 | FB1 | 24.29 | 27.58 |
| TBB361 | TB6 | 0 | 21 |
| TBB362 | FB1 | 21 | 24.29 |
| TBB371 | TB1 | 21 | 42 |
| TBB372 | FB1 | 55.16 | 58.45 |
| TBB381 | TB6 | 21 | 42 |
| TBB382 | FB1 | 61.74 | 65.03 |
| TBB391 | TB4 | 0 | 21 |
| TBB392 | FB1 | 45.29 | 48.58 |
| TBB411 | TB7 | 0 | 21 |
| TBB412 | FC1 | 58.84 | 63.05 |
| TBB421 | TB2 | 28 | 49 |
| TBB422 | FC1 | 64.3 | 68.51 |
| TBB431 | TB8 | 21 | 42 |
| TBB432 | FC1 | 46.21 | 50.42 |
| TBB441 | TB1 | 0 | 21 |
| TBB442 | FC1 | 37.67 | 41.88 |
| TBB451 | TB5 | 21 | 42 |
| TBB452 | FC1 | 42 | 46.21 |
| TBB461 | TB5 | 0 | 21 |
| TBB462 | FC1 | 22.25 | 26.46 |
| TBB471 | TB3 | 0 | 21 |
| TBB472 | FC1 | 33.46 | 37.67 |
| TBB711 | TB2 | 0 | 28 |
| TBB712 | FC1 | 28 | 29.25 |
| TBB721 | TB7 | 42 | 70 |
| TBB722 | FC1 | 70 | 71.25 |

Schedule Plan for Industrial Scale with Machine and Time Aspect

|  |  |  |  |
| --- | --- | --- | --- |
| **Operation** | **Machine** | **Start** | **Finish** |
| TBB362 | FB1 | 21 | 24.29 |
| TBB352 | FB1 | 24.29 | 27.58 |
| TAB332 | FB1 | 42 | 45.29 |
| TBB392 | FB1 | 45.29 | 48.58 |
| TAB322 | FB1 | 48.58 | 51.87 |
| TBB312 | FB1 | 51.87 | 55.16 |
| TBB372 | FB1 | 55.16 | 58.45 |
| TAB342 | FB1 | 58.45 | 61.74 |
| TBB382 | FB1 | 61.74 | 65.03 |
| TAB312 | FB1 | 65.03 | 68.32 |
| TBB332 | FB1 | 68.32 | 71.61 |
| TBB322 | FB1 | 71.61 | 74.9 |
| TBB342 | FB1 | 74.9 | 78.19 |
| TAB512 | FB1 | 78.19 | 78.52 |
| TAB522 | FB1 | 78.52 | 78.85 |
| TAB212 | FC1 | 21 | 22.25 |
| TBB462 | FC1 | 22.25 | 26.46 |
| TBB712 | FC1 | 28 | 29.25 |
| TAB422 | FC1 | 29.25 | 33.46 |
| TBB472 | FC1 | 33.46 | 37.67 |
| TBB442 | FC1 | 37.67 | 41.88 |
| TBB452 | FC1 | 42 | 46.21 |
| TBB432 | FC1 | 46.21 | 50.42 |
| TAB412 | FC1 | 50.42 | 54.63 |
| TAB432 | FC1 | 54.63 | 58.84 |
| TBB412 | FC1 | 58.84 | 63.05 |
| TBB222 | FC1 | 63.05 | 64.3 |
| TBB422 | FC1 | 64.3 | 68.51 |
| TBB212 | FC1 | 68.51 | 69.76 |
| TBB722 | FC1 | 70 | 71.25 |
| TAB722 | FC1 | 71.25 | 72.5 |
| TAB222 | FC1 | 72.5 | 73.75 |
| TAB712 | FC1 | 73.75 | 75 |
| TAB442 | FC1 | 75 | 79.21 |
| TAB411 | TA1 | 0 | 21 |
| TAB331 | TA1 | 21 | 42 |
| TAB711 | TA1 | 42 | 70 |
| TAB321 | TA2 | 0 | 21 |
| TAB341 | TA2 | 21 | 42 |
| TAB511 | TA2 | 42 | 70 |
| TAB431 | TA3 | 0 | 21 |
| TAB721 | TA3 | 21 | 49 |
| TAB521 | TA3 | 49 | 77 |
| TAB211 | TA4 | 0 | 21 |
| TAB441 | TA4 | 21 | 42 |
| TAB421 | TA5 | 0 | 21 |
| TAB221 | TA5 | 21 | 42 |
| TAB311 | TA5 | 42 | 63 |
| TBB441 | TB1 | 0 | 21 |
| TBB371 | TB1 | 21 | 42 |
| TBB341 | TB1 | 42 | 63 |
| TBB711 | TB2 | 0 | 28 |
| TBB421 | TB2 | 28 | 49 |
| TBB471 | TB3 | 0 | 21 |
| TBB391 | TB4 | 0 | 21 |
| TBB211 | TB4 | 21 | 42 |
| TBB321 | TB4 | 42 | 63 |
| TBB461 | TB5 | 0 | 21 |
| TBB451 | TB5 | 21 | 42 |
| TBB331 | TB5 | 42 | 63 |
| TBB361 | TB6 | 0 | 21 |
| TBB381 | TB6 | 21 | 42 |
| TBB221 | TB6 | 42 | 63 |
| TBB411 | TB7 | 0 | 21 |
| TBB311 | TB7 | 21 | 42 |
| TBB721 | TB7 | 42 | 70 |
| TBB351 | TB8 | 0 | 21 |
| TBB431 | TB8 | 21 | 42 |

Experiment Result on the Benchmark Flexible Jos Shop Scheduling Problem 8 Jos and 8 Machines.

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Time** | **MaxWorkload** | **TotalWorkLoad** |
| 0 | 17 | 14 | 85 |
| 1 | 17 | 13 | 77 |
| 2 | 16 | 13 | 80 |
| 3 | 15 | 15 | 81 |
| 4 | 17 | 13 | 81 |
| 5 | 19 | 15 | 93 |
| 6 | 16 | 13 | 74 |
| 7 | 16 | 15 | 86 |
| 8 | 15 | 12 | 78 |
| 9 | 15 | 12 | 83 |
| 10 | 15 | 14 | 75 |
| 11 | 17 | 13 | 79 |
| 12 | 16 | 13 | 81 |
| 13 | 16 | 13 | 78 |
| 14 | 17 | 15 | 87 |
| 15 | 16 | 16 | 80 |
| 16 | 15 | 13 | 84 |
| 17 | 16 | 14 | 84 |
| 18 | 15 | 13 | 80 |
| 19 | 15 | 13 | 80 |
| 20 | 18 | 17 | 99 |
| 21 | 16 | 11 | 78 |
| 22 | 16 | 13 | 83 |
| 23 | 16 | 13 | 79 |
| 24 | 16 | 16 | 87 |
| 25 | 15 | 14 | 79 |
| 26 | 16 | 16 | 84 |
| 27 | 16 | 14 | 81 |
| 28 | 15 | 12 | 78 |
| 29 | 15 | 13 | 91 |
| 30 | 16 | 13 | 88 |
| 31 | 14 | 14 | 80 |
| 32 | 14 | 14 | 80 |
| 33 | 14 | 14 | 80 |
| 34 | 16 | 12 | 75 |
| 35 | 19 | 14 | 93 |
| 36 | 18 | 14 | 90 |
| 37 | 15 | 14 | 80 |
| 38 | 16 | 13 | 82 |
| 39 | 16 | 14 | 75 |
| 40 | 15 | 12 | 76 |
| 41 | 15 | 12 | 75 |
| 42 | 19 | 13 | 89 |
| 43 | 17 | 13 | 79 |
| 44 | 17 | 14 | 91 |
| 45 | 14 | 12 | 77 |
| 46 | 15 | 14 | 75 |
| 47 | 15 | 14 | 83 |
| 48 | 15 | 15 | 77 |
| 49 | 14 | 12 | 77 |
| 50 | 15 | 12 | 77 |
| 51 | 16 | 13 | 73 |
| 52 | 14 | 14 | 86 |
| 53 | 15 | 13 | 83 |
| 54 | 14 | 12 | 77 |
| 55 | 16 | 16 | 86 |
| 56 | 16 | 14 | 78 |
| 57 | 17 | 15 | 84 |
| 58 | 15 | 12 | 75 |
| 59 | 18 | 13 | 95 |
| 60 | 15 | 12 | 78 |
| 61 | 18 | 14 | 87 |
| 62 | 17 | 15 | 81 |
| 63 | 17 | 16 | 93 |
| 64 | 18 | 13 | 92 |
| 65 | 16 | 13 | 84 |
| 66 | 16 | 13 | 79 |
| 67 | 17 | 13 | 80 |
| 68 | 16 | 13 | 87 |
| 69 | 15 | 12 | 79 |
| 70 | 14 | 13 | 79 |
| 71 | 15 | 12 | 77 |
| 72 | 16 | 15 | 82 |
| 73 | 16 | 13 | 87 |
| 74 | 20 | 15 | 91 |
| 75 | 16 | 14 | 79 |
| 76 | 15 | 14 | 88 |
| 77 | 20 | 15 | 92 |
| 78 | 15 | 15 | 81 |
| 79 | 17 | 14 | 83 |
| 80 | 16 | 13 | 88 |
| 81 | 16 | 13 | 77 |
| 82 | 17 | 13 | 77 |
| 83 | 15 | 12 | 83 |
| 84 | 16 | 12 | 75 |
| 85 | 15 | 15 | 81 |
| 86 | 19 | 16 | 96 |
| 87 | 15 | 12 | 77 |
| 88 | 15 | 13 | 76 |
| 89 | 15 | 14 | 77 |
| 90 | 16 | 13 | 77 |
| 91 | 19 | 14 | 78 |
| 92 | 18 | 14 | 84 |
| 93 | 18 | 15 | 103 |
| 94 | 16 | 13 | 83 |
| 95 | 14 | 14 | 82 |
| 96 | 16 | 13 | 89 |
| 97 | 16 | 13 | 82 |
| 98 | 15 | 13 | 88 |
| 99 | 15 | 12 | 79 |