INTELLIGENT PRODUCTION SCHEDULING FOR MOULD MAKING

TANG LAP YING

M.Phil

The Hong Kong Polytechnic University

2015

The Hong Kong Polytechnic University

Department of Industrial and Systems Engineering

Intelligent Production Scheduling for Mould Making

TANG Lap Ying

A thesis submitted in partial fulfillment of the requirements

for the degree of Master of Philosophy

October 2013

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____ (Signed)

_____TANG Lap Ying 鄧立瑩_____(Name of student)

This work is dedicated to my beloved parents,

for their endless support and wholehearted trust.

Abstract

Scheduling is an important decision-making process in manufacturing industries. Scheduling problems are usually modeled and solved in a mathematically feasible way. As a result, the solutions generated from these greatly simplified problems are infeasible for real-life cases. The complexity and instability of production systems are still underestimated in many scheduling techniques in academic literature. Furthermore, most of the scheduling techniques are problem-specific; and the flexible production model in mould shop has been rarely studied. It is important to develop an appropriate scheduling algorithm to meet the industry's need.

Asahi (H.K.) Ltd. is a plastic product and plastic injection mould manufacturer. Their products are diversified, including electronic product dummies and accessories. These products are mainly produced by thermoplastic injection moulding, and specific mould has to be prepared before injection moulding of any plastic part. The tooling department of Asahi (H.K.) Ltd. is responsible for the mould design and manufacture. Due to the uniqueness of its injection moulds, the components in each mould are different. Different operations and processing routes have to be taken. This high flexibility component variety however causes difficulty in today's quick response production planning and scheduling.

In order to efficiently find an optimal schedule for real life mould shop, this research thus proposed to tackle the Asahi's mould shop scheduling problem with new heuristic and meta-heuristic algorithms. They are hybrid Nawaz-Enscore-Ham (NEH), Random Keys Harmony Search (RKHS), and Random Keys Genetic Algorithm (RKGA). For the meta-heuristic algorithms, different parameter values are selected for parameter tuning and choice of parameters is provided.

These constraint-handling-free algorithms are implemented with MATLAB. The random keys representation can avoid the existence of duplicated position value in sequencing after re-sequencing. The computational results demonstrate that RKGA performs the best among the proposed algorithms. In average, the best value RKGA generated is 3.84% better than the best value of the proposed heuristic algorithms. Data adapted from Asahi's production were tested. All the algorithms can finish computation within 10 seconds. It is suggested that the proposed algorithms can be applied in different scheduling problems in future study.

Publications

Tang, L. Y., Yu, K. M., &Wan, Peter (2011). Flexible Mould Shop Scheduling using Harmony Search. *Annual Journal of Institute of Industrial Engineers (Hong Kong), 31*, 413-419.

Tang, A. L. Y., Yu, K. M., & Wan, Peter (2012). Makespan Minimization On Injection Mould Shop Scheduling Using Hybrid Dispatching Rules And Harmony Search. *Proceedings of the IASTED International Conference Artificial Intelligence and Soft Computing*, 264-269.

Acknowledgements

I would like to express my sincere appreciation and gratitude to my chief supervisor, Dr. K. M. Yu, for invaluable advice, continual support and encouragement throughout the entire research period.

Furthermore, I would like to thank for the financial assistance supported by Asahi (H.K.) Ltd. and Department of Industrial and Systems Engineering, the Hong Kong Polytechnic University under Teaching Company Scheme (Project no.: H-ZW0Z).

I would like to offer particular thanks to Mr. Gordon Chan, the Chairman, Miss Clara Chan, the CEO, and Mr. Peter Wan, my industrial supervisor of Asahi (H.K.) Ltd. for giving me an opportunity to participate in this project. I am grateful to my colleagues, Mr. L. K. Yuen, Mr. Stanley Leung and Mr. Simon Chui for providing me assistance throughout the case study in Asahi (H.K.) Ltd.

Finally, I would like to thank my families and friends who have whole heartedly supported my study.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ACO	Ant Colony Optimization
ADSA	Autonomous Decentralized Scheduling Algorithm
ATO	Assemble-To-Order
B&B	Branch And Bound
BOM	Bill-Of-Material
CAPP	Computer-Aided Process Planning
CNC	Computer Numerical Control
CPM	Critical Path Method
EDM	Electric discharge machining
EPS	Entry Point Sequence
ERP	Enterprise Resource Planning
ERT	Earliest Release Time
ETO	Engineer-To-Order
FFSSP	Flexible Flow Shop Scheduling Problem
FIFO	First-In-First-Out
GA	Genetic Algorithm
HS	Harmony Search
I.T.	Information Technology
IDEF	Icam DEFinition for Function Modelling
IPO	Input-Process-Output

ISO	International Organization for Standardization
LM	Lean Manufacturing
LPT	Longest Processing Time
MTO	Make-To-Order
MTS	Make-To-Stock
NEH	Nawaz-Enscore-Ham
NP	Nondeterministic Polynomial
OPP	Order Penetration Point
PERT	Program Evaluation and Review Technique
PFSSP	Permutation Flow Shop Scheduling Problem
PMC	Procurement and Material Control
PT	Processing Time
R&D	Research and Development
RK	Random Keys
RKGA	Random Keys Genetic Algorithm
RKHS	Random Keys Harmony Search
SDSM	Scheduling Decision Support Model
SME	Small and Medium Enterprise
SPT	Shortest Processing Time
TOC	Theory Of Constraints

Chapter 1 Introduction

1.1. Research Company Background

Asahi (H.K.) Limited, established in 1995, is a Hong Kong based plastic injection product manufacturer with main production base located in Dongguan, China. The company has equipped high-end automatic machines and ERP system. It has obtained certification of several international standards.

Asahi specializes in manufacture of high precision moulds for plastic components of a diverse range of product types, such as electronic products' dummies, baby toys and syringes. The comprehensive services provided by Asahi ranging from tool design to mass production. Depending on customers' requests, injection moulds can be solely sold as the end products.

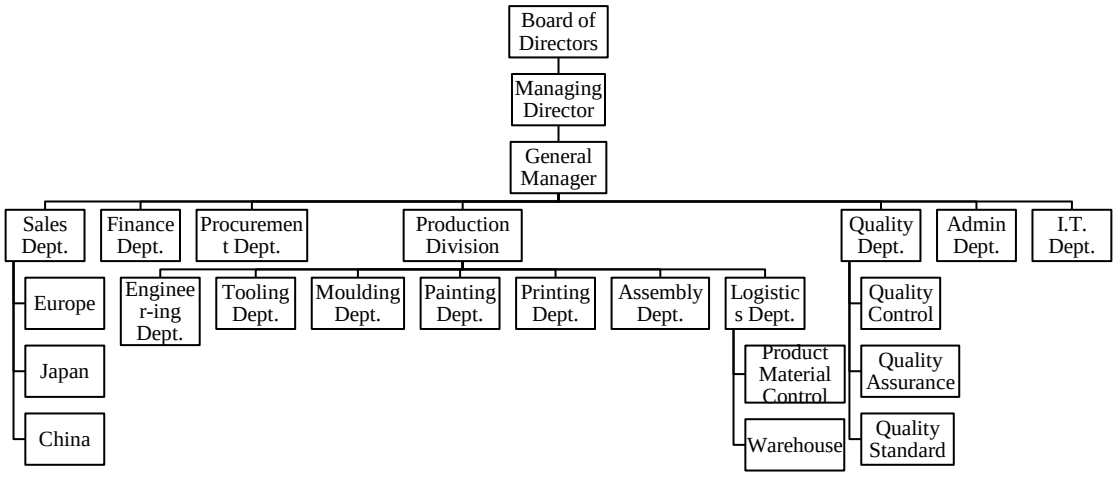

Figure 1.1 Official organization chart of Asahi (H.K.) Limited

The current company's organization is displayed in Figure 1.1.The company aims at becoming a world class enterprise of high precision mould manufacturer

by making a breakthrough in mould making methodology.

1.2. Introduction of Mould Making

Injection moulding is a manufacturing process for mass production of plastic products with same shape and size. This is a critical process in plastic product manufacture. It is performed on an injection moulding machine (Figure 1.2) with a mould clamping on it. The injection process is explained as follows: 1) plastic granules are heated and melted into liquid form; 2) the molten liquid plastic is then injected into the mould until the mould's inner space is filled; 3) after cooling, the plastic is solidified and ejected from the mould. The whole process repeats until the required volume of plastic parts have produced.

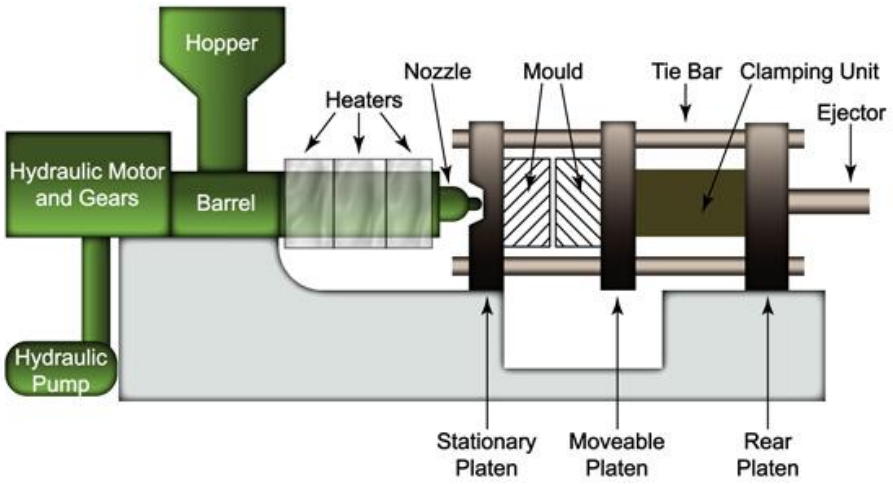

Figure 1.2 An injection moulding machine

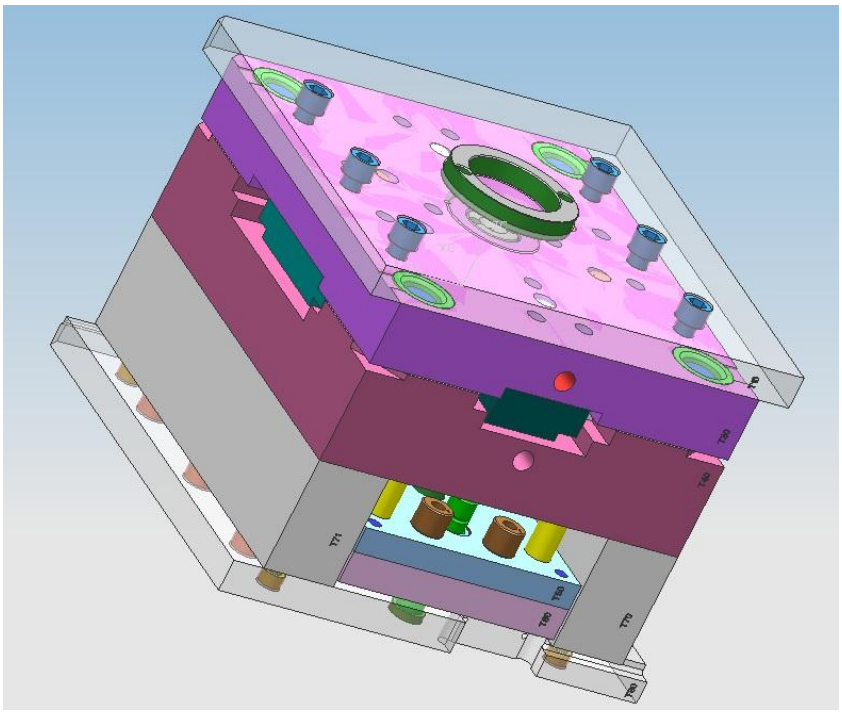

Figure 1.3An engineering drawing of an injection mould

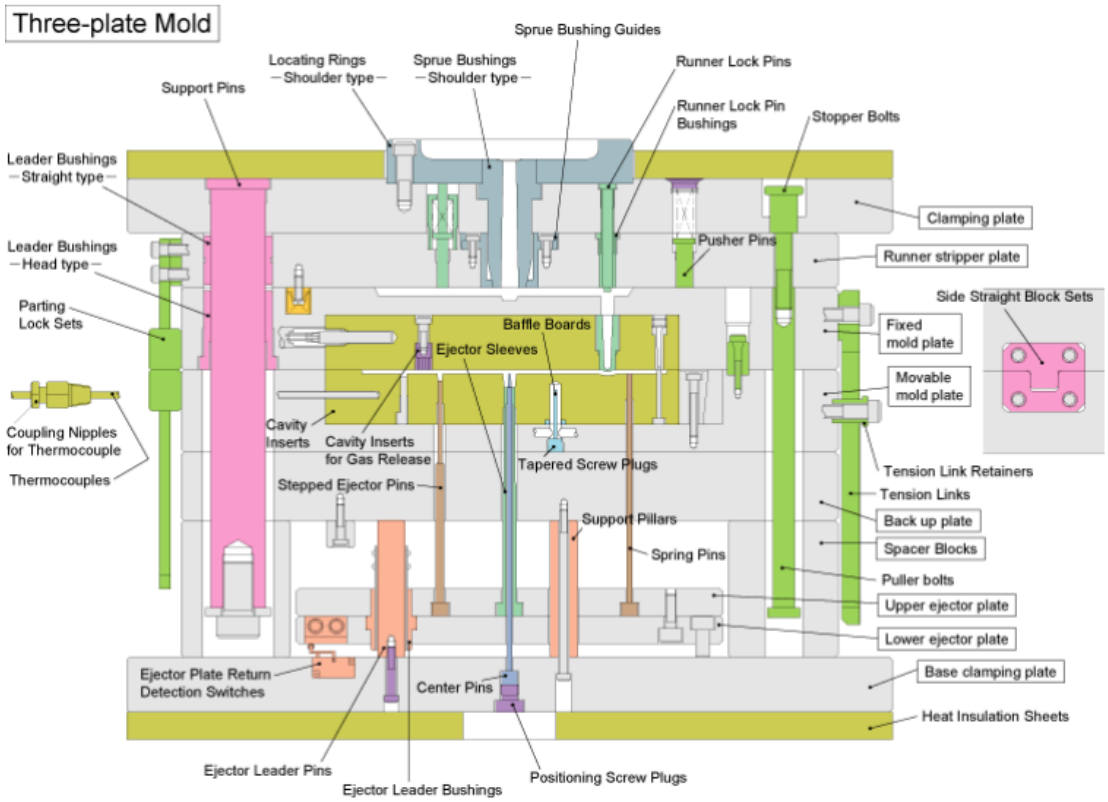

Figure 1.4 A cross-sectional drawing of an injection mould

Injection moulds (Figures 1.3 & 1.4) are fundamental tools in injection moulding. They differ widely in their size and composition. Each mould is tailor-made for producing specific plastic components. It is a very complex device with numerous components. A mould is constituted by hundreds of components, in which half of them have to be further processed in the mould shop while the other half, as standard components, can be bought directly from suppliers without further processing. There would be vast amounts of highly diversified components produced in the mould shop with precedence and route constraints. When all the mould components are prepared, a mould would be assembled accordingly. The general workflow in mould shop is illustrated in Figure 1.5.

The mould components are made by undergoing a series of operations in mould shop. The processing routes vary from component to component. It is determined by the mould components' shapes. Take a mould cavity (Figure 1.6) as an example; it is the mould component to shape the plastic into mobile phone shell. The mould cavity is processed from a steel cube. The processing sequence is described as follows: 1) mill the cube with a margin volume (green highlights); 2) drill the holes (purple highlights); 3) further milling the cube (red highlights); 4) stop up undesirable openings with copper stoppers to complete the cooling channels; 5) grind the cube's plane (green highlights) to meet specification; 6) mill the cube with high precision machine to meet specification (red highlights); 7)

EDM the fine details (blue highlights); 10) polish the cube to meet roughness requirement.

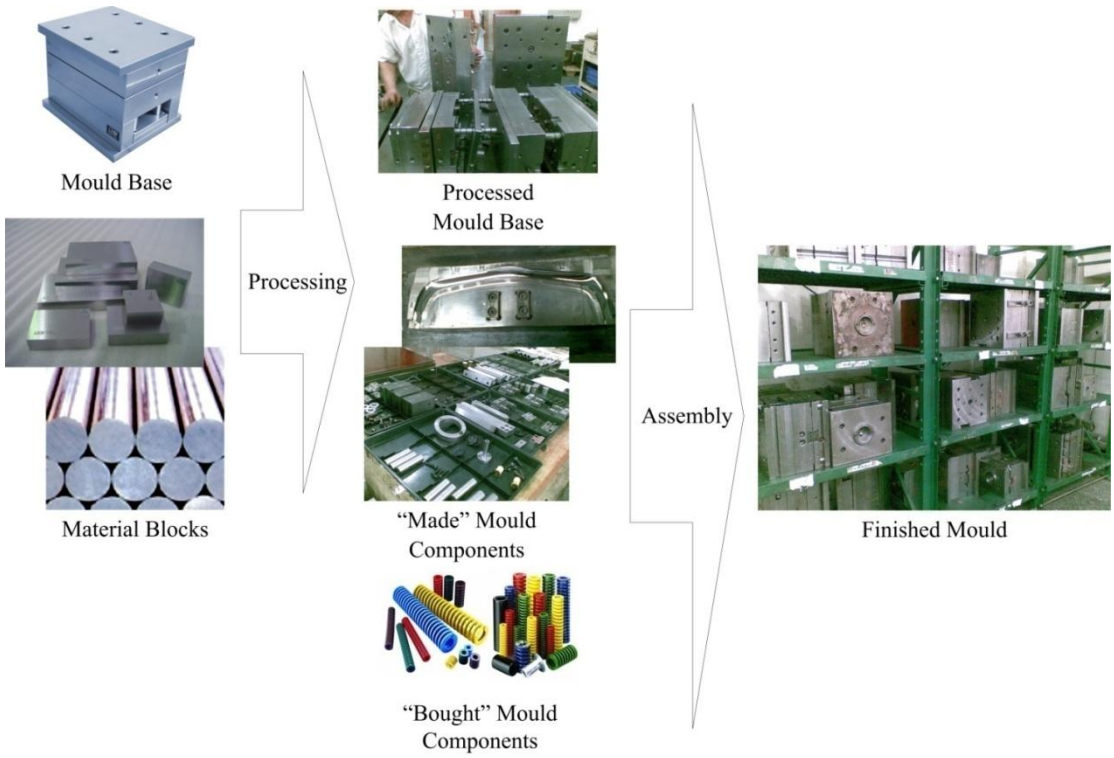

Figure 1.5The general workflow in mould shop

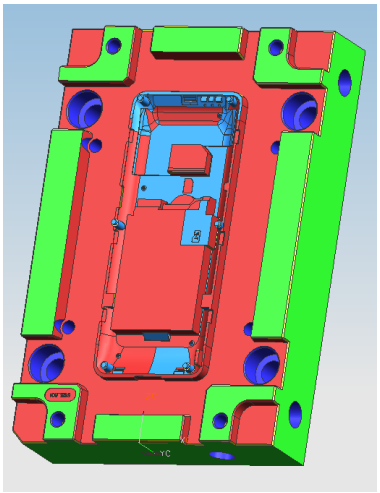

Figure 1.6A mould cavity for shaping mobile phone shell

1.3. Research Company Operations

Asahi mainly acts as a contract manufacturer for the renowned brands. Mostly, customers approach Asahi with the product prototypes. A production project can be started only after a customer's order is confirmed. As a result, the company adopts the pull-type engineer-to-order production strategy. Production is planned on project-based with limited lead time. There are several projects running simultaneously. Any unexpected incident can alter the whole plan. Figure 1.7shows the product breakdown structure of a ready-to-delivery plastic product. Injection moulds are fundamental tools in plastic product manufacturing. Due to the tight schedule requested by customers, Asahi must have better control in production planning. Quick response to unexpectedness is the key to remain competitive in the industry.

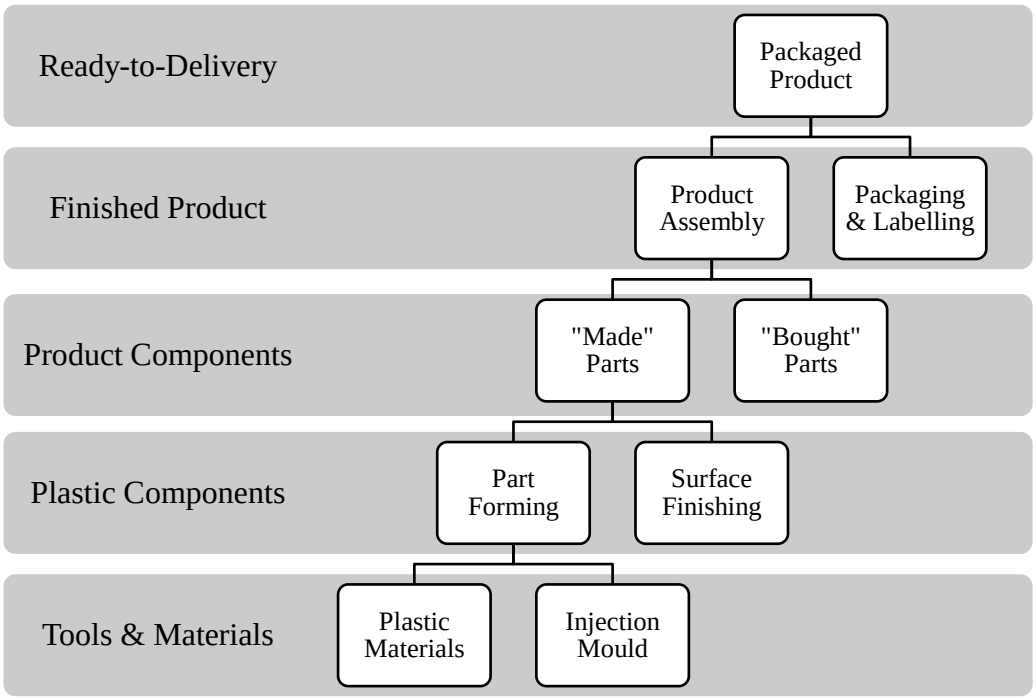

Figure 1.7 Product breakdown structure of a ready-to-delivery product

The relationships between the parties involved in Asahi's business operations are shown in Figure 1.8. Sales department is the front-line department facing customers. It is responsible for fostering the communications between Asahi's internal departments and customers. Production division involves departments in charge of producing the finished goods. Procurement department maintains and manages a supplier pool, which can provide materials or services to the company if necessary. Finance department deals with financial matters. Quality department assesses the quality of all the products and operations in the company. Admin department is responsible for staff recruitment, staff welfare and facility maintenance. I.T department handles all the network and system problems.

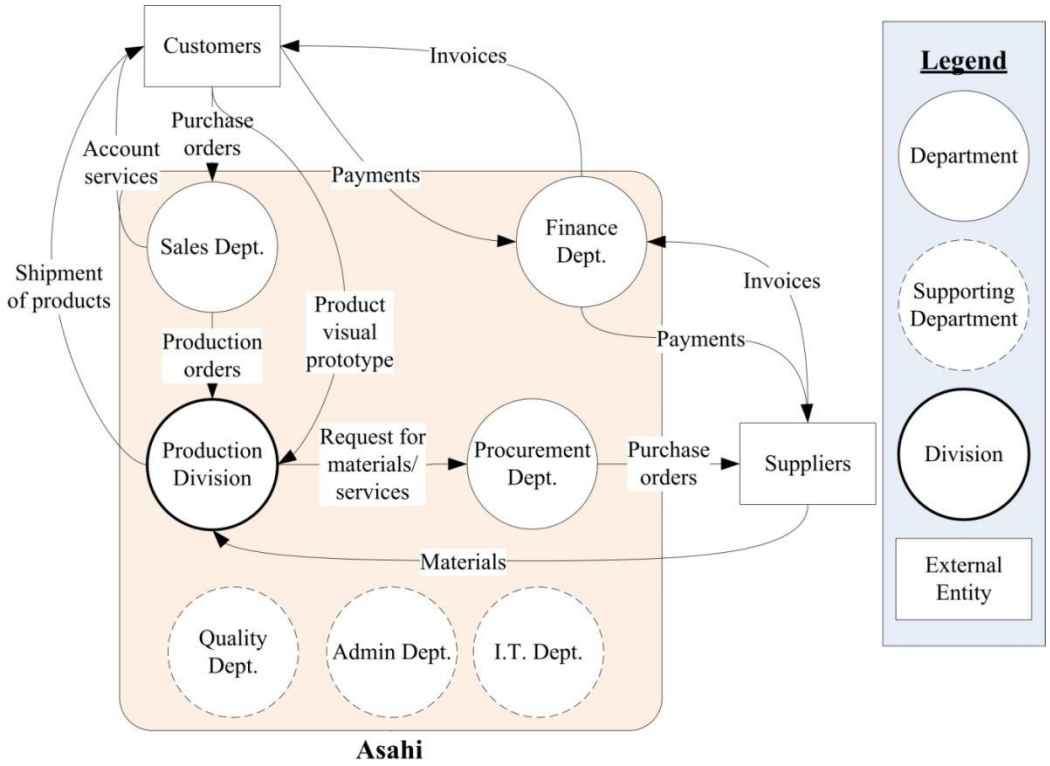

Figure 1.8Relationship map of the parties involved in Asahi's business operations

1.4. Project Development

The general project development workflow starts with sales department (Figure 1.9).Once a sales representative receives a customer's request, the company's capability is evaluated internally to ensure that the request can be fulfilled. Based on the company's current situation, a proposal, including price and payment method, is suggested to the customer. Negotiation is normally required before coming to an agreement.

After reaching an agreement, the new project can be launched. The project starts with product engineering development. In this process, all the engineering designs and specifications are confirmed. All the follow-up production processes have to abide by these specifications. Table 1.1 relates each main production stage to the department(s) which is/are responsible for the corresponding stage. Table 1.2 specifies the production departments' responsibilities.

The tools used for mass producing plastic parts are injection moulds. Tooling department is responsible for making moulds which give shape to plastic parts. When all the tools and materials are ready, a small amount of product samples are produced in production pilot run. Moulding, painting and printing are the processes to produce plastic parts. Moulding is a shape forming process while painting and printing are the surface finishing processes. Once all the "made" parts

and "bought" parts are prepared, they can be assembled into the finished products. After packaging and labeling, these product samples are delivered to the customers for quality checking. Figures 1.9 and 1.10 describe the activities involved from Product Engineering Development to Pilot Production Run using IDEF0 approach. The product samples are the proof of the company's production capability. If the samples are satisfactory, it signifies the accessibility of the production processes.

Project Workflow	**Responsible Department**
Contract Establishment	Sales
Product Engineering Development	Engineering
Mould Development	Tooling
Production Pilot Run	Moulding, Painting, Printing, Assembly
Mass Production	Moulding, Painting, Printing, Assembly
Delivery	Logistics

Table 1.1Responsible departments for each project stage

Production Departments	Responsibilities
Engineering	Design products with engineering specifications
Tooling	Design and manufacture injection moulds for plastic parts
Injection Moulding	Produce plastic parts using injection moulding
Painting	Paint the plastic parts
Printing	Print patterns on plastic parts' surfaces
Assembly	Assembly the finished parts into a finished product
Logistics	Receive incoming materials and deliver ready-to-delivery products

Table 1.2Production departments' responsibilities

Following the approval of product samples, the product is eligible for mass production. The processes in mass production are generally as same as in the pilot production. The major difference is the number of products to be produced. After the production and delivery of required number of products, the order is considered to be fulfilled.

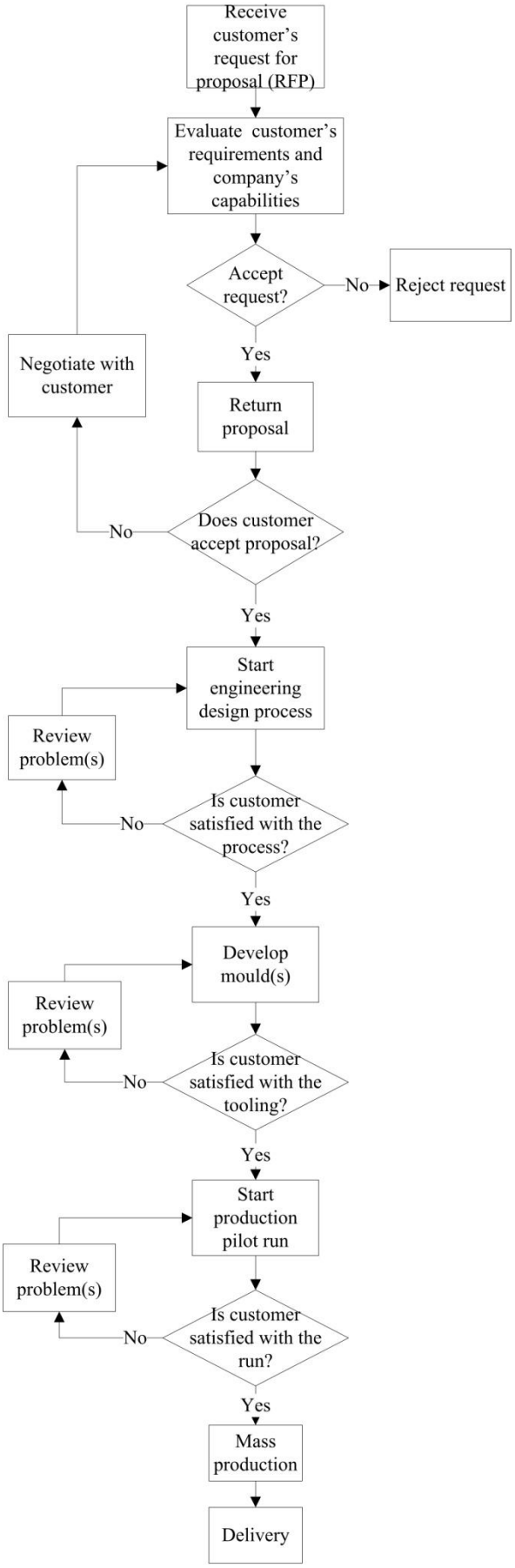

Figure 1.9Project development workflow

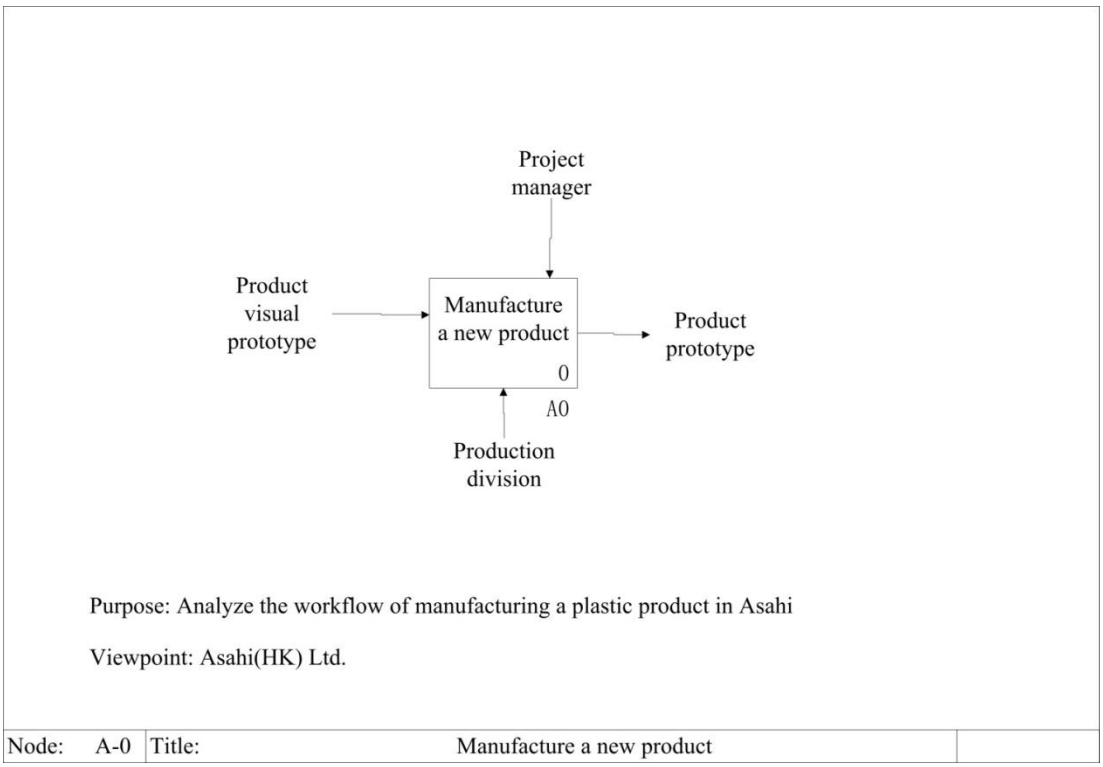

Figure 1.10Workflow of manufacturing a new product

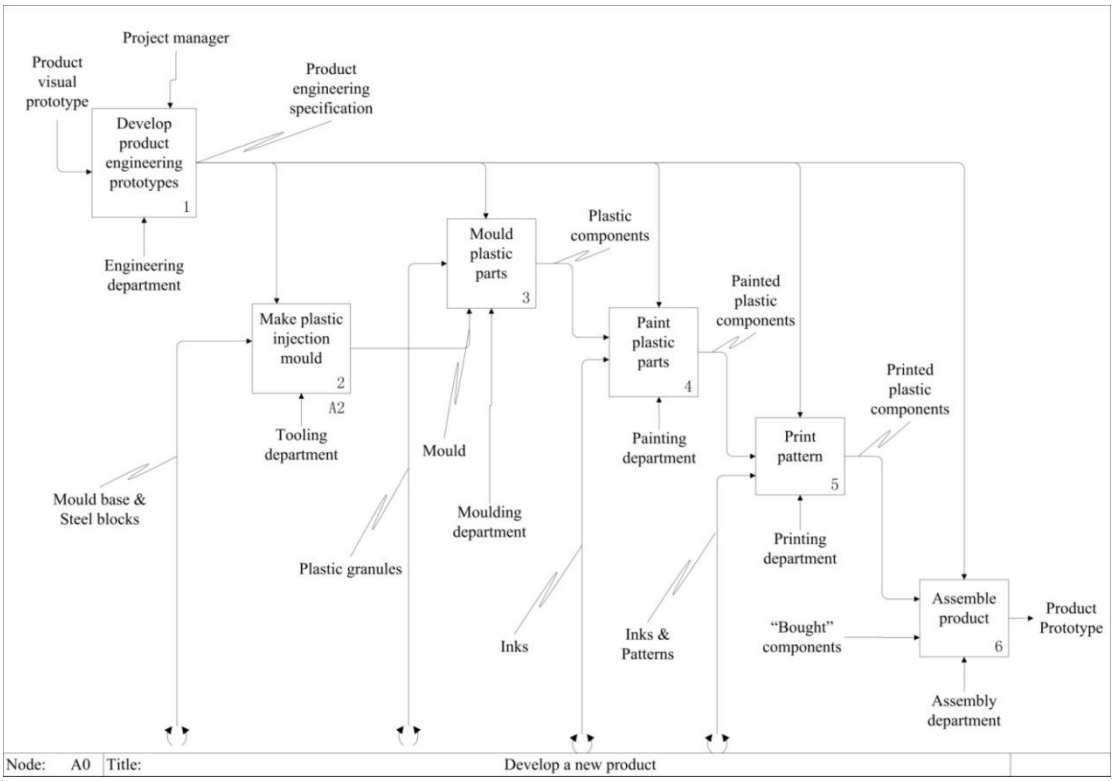

Figure 1.11Detailed workflow of manufacturing a new product

1.5. Moulding Tool Development

Tooling Department is the one responsible for mould making in Asahi. Tooling Department starts a new project after receiving the notification from engineering department. The finished moulds are delivered to moulding department for trial moulding. At present, 100 sets of moulds are made in average every month. The average mould making duration is around 15 days. Project tardiness often occurs. As mould making is a critical process under overall plastic product manufacturing, delays in delivering moulds can lead to delay in overall project completion. Failure to meet the delivery date is an act of breaching the contract, which can result in loss of business relationship and credibility.

A context diagram of relationships between tooling department and related internal departments is shown in Figure 1.12. Tooling department starts making new moulds for a new product after receiving notification from Engineering department. The workflow in this department is illustrated in Figure 1.13. Figure 1.14 is the decomposition of the functional component "Make plastic injection mould" of IDEF0 model (Figure 1.11). It shows the activities involved in Tooling department. A context diagram of relationships between tooling department and related internal departments is shown in Figure 1.12. Tooling department starts making new moulds for a new product after receiving notification from Engineering

These activities are explained below.

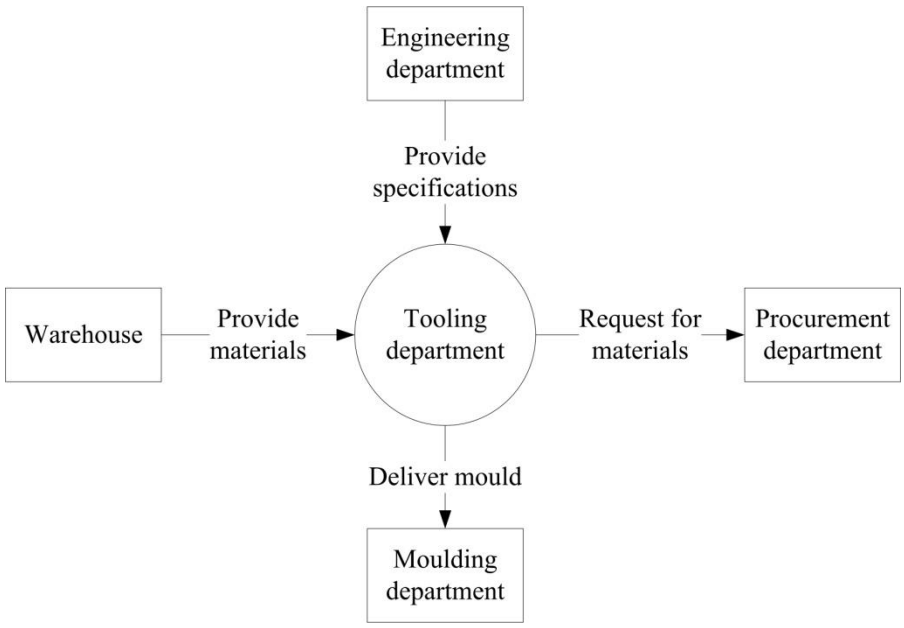

Figure 1.12Relationships between tooling department and related internal departments

1.5.1. Design Mould

After receiving product engineering design, mould designers start designing the moulds for each plastic part. A mould originates with a standard mould base, which is composed of several mould plates. More components are usually added on in order to form the complicated structure of the plastic part. The mould design confines the quality of finished plastic products. This involves the consideration of injection moulding mechanism. The designers have to determine the type of mould used, design the layout of mould components, establish the Bill-Of-Material (BOM), and decide the manufacturing process for each mould component.

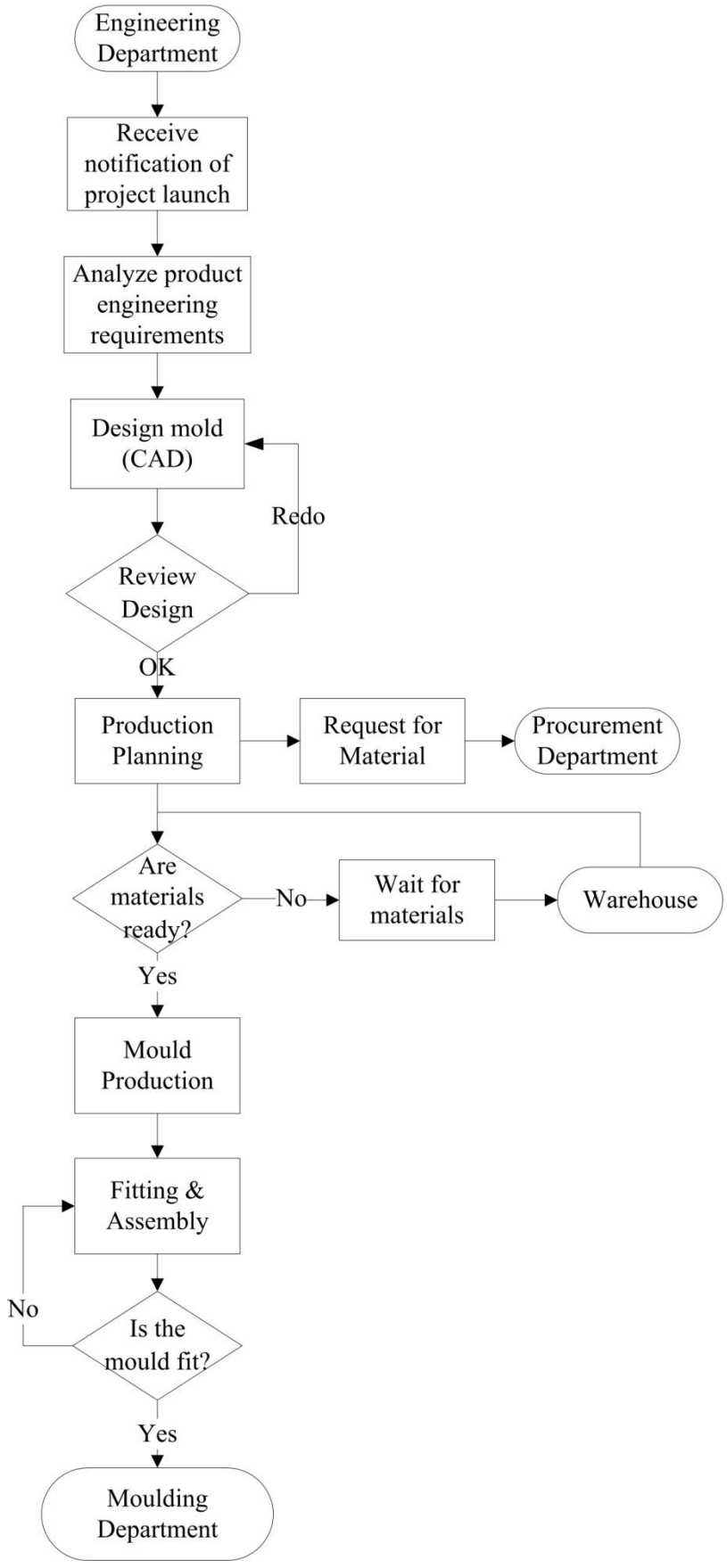

Figure 1.13Tooling department's workflow

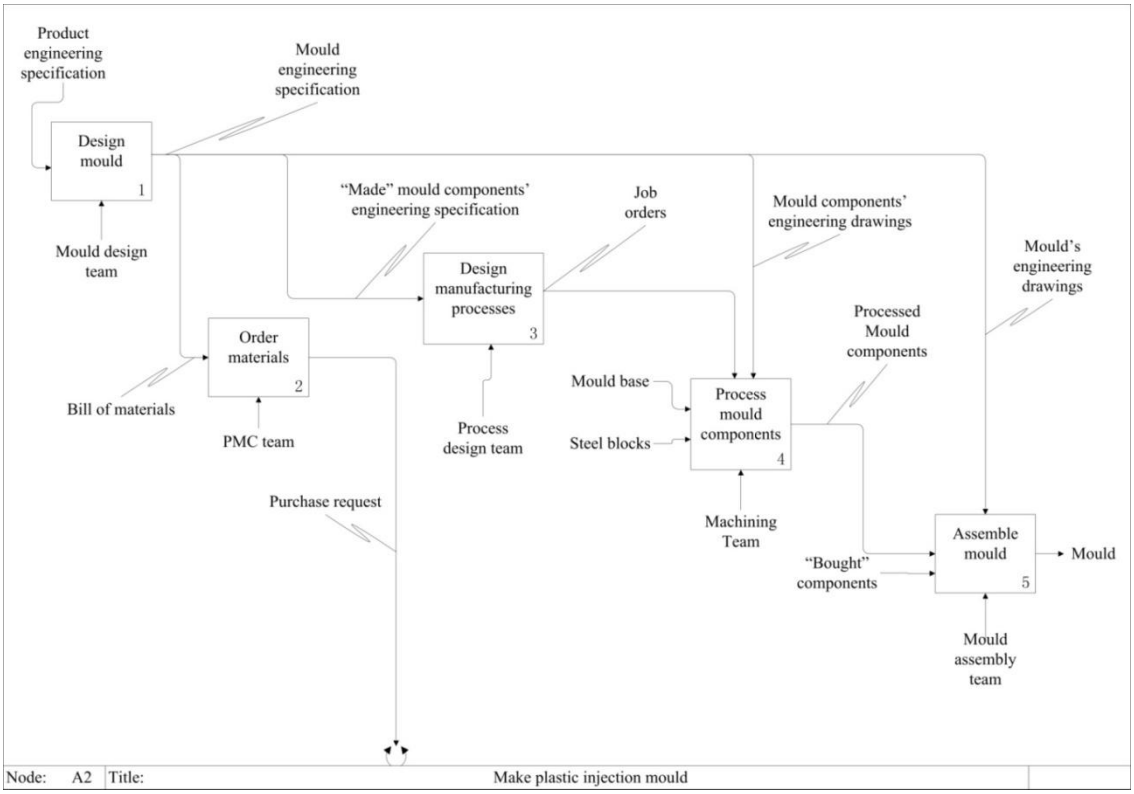

Figure 1.14Workflow of making a plastic injection mould

1.5.2. Order Materials

The mould components can be categorized into two types: standard and tailor-made. Standard components are the ones which are commonly used in different moulds. They can usually be assembled into the mould without further processing in the mould shop. These components include screws, springs, ejector pins, locating rings and bushings. Safety stocks of these components are maintained to deal with the uncertainties in demand. Once the BOM is issued, the Procurement and Material Control (PMC) Team will check the stock levels of required standard components in case any of them are out of stock.

1.5.3. Design Manufacturing Processes

Tailor-made components are the ones which vary according to the mould design. These components include cores, cavities, slides and lifters. They are made of steel blocks. After further processing in the mould shop, these steel blocks become mould components.

1.5.4. Process Mould Components

The operations in the mould shop can be categorized into manual and computer numerical control (CNC) operations. Input-Process-Output (IPO) models of these two types of operations are shown in Figures1.15 and 1.16respectively. Comparison between manual machining and CNC machining operations is given in Table 1.3.

All the operations in mould shop are for machining purpose except the one for coordinate measuring. Due to outstanding performance in obtaining dimension accuracy, CNC machining operations are usually used for finish machining. In contrast, manual machining operations are responsible for rough machining due to their relatively low operating costs. The Manual machining operations demand sophisticated operation skills of operators while CNC machining operations demand sophisticated programming skills of CNC programmer. Table 1.4 describes the operations involved in mould shop. Among all the operations, only

sinker-EDM requires tailor-made tools, which are tool-electrodes. They are made in the mould shop as well. The making of such tools complicates the whole mould making process. Further explanation will be provided in later section.

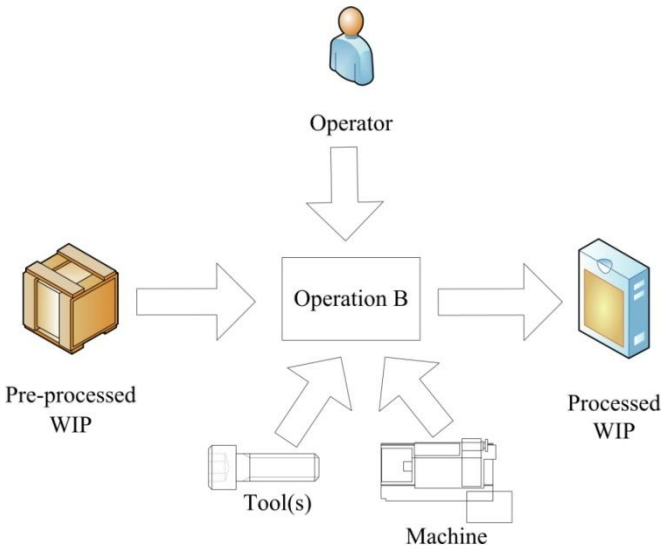

Figure 1.15 IPO model of manual operations

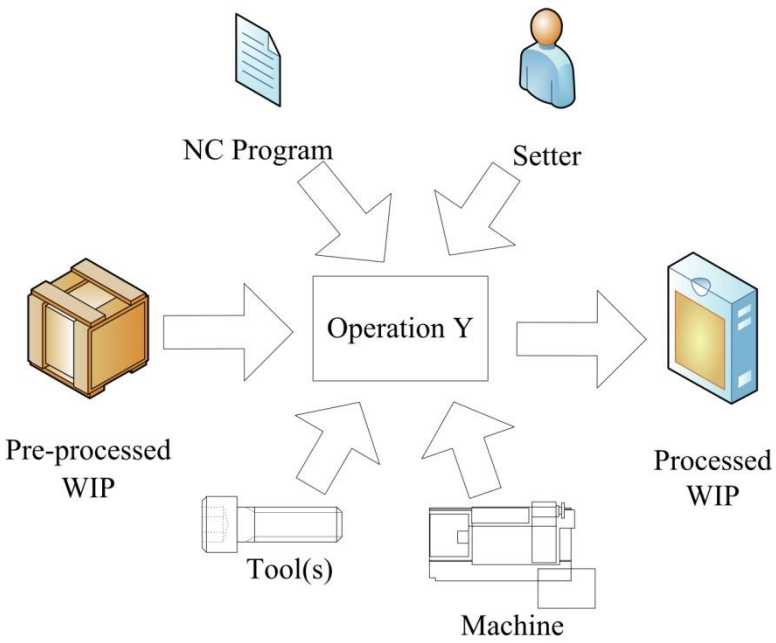

Figure 1.16IPO model of CNC operations

1.5.5. Assemble Mould

When all the components have been made or bought, they can be assembled into a mould. As a mould is a high precision device, small inaccurate alignment of its components can result in moulding defects. If there is any problem in fitting, reworking must be carried out. After assembly, the mould can be taken to trial injection in moulding department.

	Manual operations	**CNC operations**
Required labour	Operators	NC programmers and machine setters
Critical skills	Operating skills of operators	Programming skills of NC programmers
Operating costs	Lower	Higher
Dimension accuracy	Lower	Higher
Feed rate	Lower	Higher

Table 1.3Comparison between manual machining and CNC machining operations

Operation Types	**Operations**	**Applications**	**Tools' Characteristics**
Manual	Manual Milling	Rough machining workpieces, creating runners	Reusable, standard
	Deep hole drilling	Creating cooling channels	Reusable, standard
	Turning	Machining cylindrical workpieces, e.g. pins	Reusable, standard
	Grinding	Straighten the planes and sharpen the angles	Reusable, standard
	Polishing	Enhancing surface roughness	Consumable, standard
Computer Numerical Control (CNC)	CNC Milling	Finish machining workpieces	Reusable, standard
	Coordinate measuring	Verifying geometry	Reusable, standard
	Sinker-EDM	Further finish machining workpieces, creating complex features which milling cannot achieve	Consumable, *tailor-made*
	Wire-EDM	Cutting through workpieces, e.g. creating pins' slots	Consumable, standard

Table 1.4Operations involved in mould shop

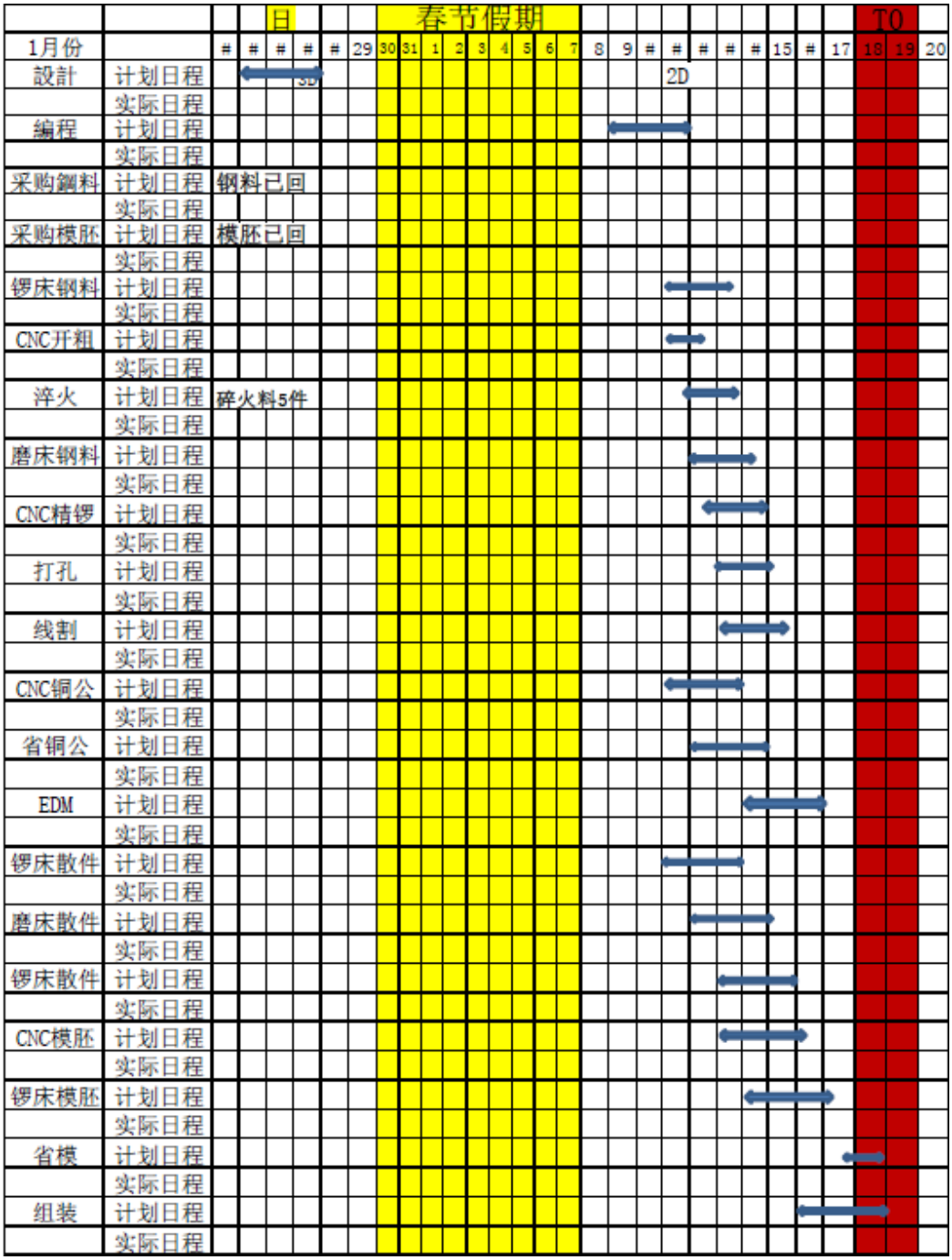

Figure 1.17 Mould production schedule created in Excel by production planner manually

1.6. Research Company Problems

The problems in production planning faced by Tooling Department are

elaborated below:

1.6.1. Inadequacy of present production planning approach

The project schedule is discussed with customers before the launch of any project. Tooling Department is responsible for giving an estimated mould making time for reference. However, production plan (Figure 1.17) of each project is constructed on an ad hoc basis. The engineer-to-order characteristic of mould making industry causes the unavailability for predicting future workload. The wide variation in production routes complicates the planning process. The total processing times can only be estimated roughly before the completion of detailed mould design. When multiple projects are running simultaneously, it is highly possible that the projects' schedules conflict with one another's.

Currently, there are two solutions to deal with the overload problems: first, outsource some of the jobs; second, request for postpone of delivery date. The latter solution, which can lead to dissatisfaction of customers, is less preferable. Outsourcing is usually decided at the last minute which decreases the flexibility of the project schedule.

1.6.2. Frequent happenings of unpredictable incidents

Unpredictable incidents often alter projects' progress and induce rescheduling

of the tasks. These incidents include delays in order releases, releases of urgent orders, cancellations of orders and machine breakdowns.

Delays in order releases happen when the precedent operations cannot finish on time or the required resources are not available. Urgent orders are usually the reworking operations. A workpiece is reworked when it cannot pass the quality check or its specification is amended. These urgent orders add extra workloads to the mould shop. Cancellations of orders occur when project managers ask for a pause. Machine breakdowns can be caused by operation errors or malfunctions of machines. It takes a period of time to fix the problems so the mould shop capacity drops until the machines are repaired. Frequent happenings of unpredictable incidents bring chaos to the mould shop.

1.7. Research Aims and Objectives

This research aims to identify the adequacy of present production planning approach in Asahi, propose a more efficient way to construct the production plans, and improve the adaptability of the mould shop to any unexpected disruptions.

The objective of this project is to investigate the best scheduling algorithm for Asahi (H.K.) Limited which can generate feasible schedules quickly to production planner for decision making.

1.8. Significance of Research

Majority of research in mould making are mainly on the technical side rather than on the management side; whereas research in production planning is seldom specifically to mould shop. This project attempts to study the mould shop planning in view of the new flexible product development endeavor nowadays where decision making is delayed as far as possible and needs to be revised as quickly as possible.

In addition, the mould makers in Southern China and Hong Kong now face a fierce global competition. To maintain competitiveness, the company needs to continuously improve their know-how and better use their tools which are also available to their competitors. With increasing labour costs and shortage of experienced labour, Asahi would like to implement automation in tooling department so that dependency on unstable supply of labour can be lowered and quality control on mould making can be improved. It is a timely study on the new operation challenges faced by the industry.

1.9. Assumptions and Limitations

This research project is company-based. The case study of mould making is based on current situation in Asahi (H.K.) Ltd. It is believed that the methodology is applicable to general mould shops. Any sensitive data which is considered as the

confidential commercial information would not be disclosed. Only the research approach and methodology with non-sensitive data can be published.

Since there are data integrity issues, direct adoption of company data for testing is unfeasible. Data is remade to simulate the case.

1.10. Thesis Outline

This thesis is divided into the following eight chapters:

Chapter 1 introduces the company background and existing problems and explains the significance and objectives of the study.

Chapter 2 reviews the literature irrelevant problems, including the production problems in mould making. The existing solutions and methodologies for solving the problems are studied.

Chapter 3 gives a detailed problem description and formulation. Working principles of proposed algorithms are introduced.

Chapter 4 describes the implementation of the proposed algorithms, the ways to evaluate the effectiveness of suggested methodologies.

Chapter 5 presents the computational results of the methodology out from

MATLAB computation.

Chapter 6 discusses the limitations of the proposed solutions and the difficulties encountered during the study.

Chapter 7draws conclusions of the work undertaken and suggests the future work can be carried out following this study.

Chapter 2 Literature Review

2.1. Overview of Mould Making Industry Trend

Over the last decade, the competition among mould makers is highly fierceunder one global market.Many mould makers has adpoted different stratigies in order to remain competitive. They are aiming at making moulds with better quality, shorter delivery time and lower cost.

Fallböhmer, et al. [1996] surveyed the die and mould making industry in Germany, Japan and United States dated in 1996. At that time, majority of the mould makers produced less than 500 moulds annually. The lead times for production were mostly ranged between five to twenty weeks. Comparing to the past mould shops, the present Asahi's mould shop, which produces nearly 2400 moulds annually with 15-day lead time, has much higher annual production rate and volume. Although the general processing steps (Figure 2.1) of mould making have not changed much, the difference in production performance is probably resulted from the advanced machine performance and relevant digital technologies.

A survey conductedby Loendorfin [2008] shows that American mould makers have applied various methods and techniques to streamline the operations, improve quality and increase efficiency in order to stay competitive in global

competition. Among the numerous actions taken by the mould makers, purchasing new technologies and equipment along with improving workflow and scheduling has brought the greatest positive impact. Since Asahi has already acquired advanced machinery, the next step Asahi should do is to improve its workflow and scheduling.

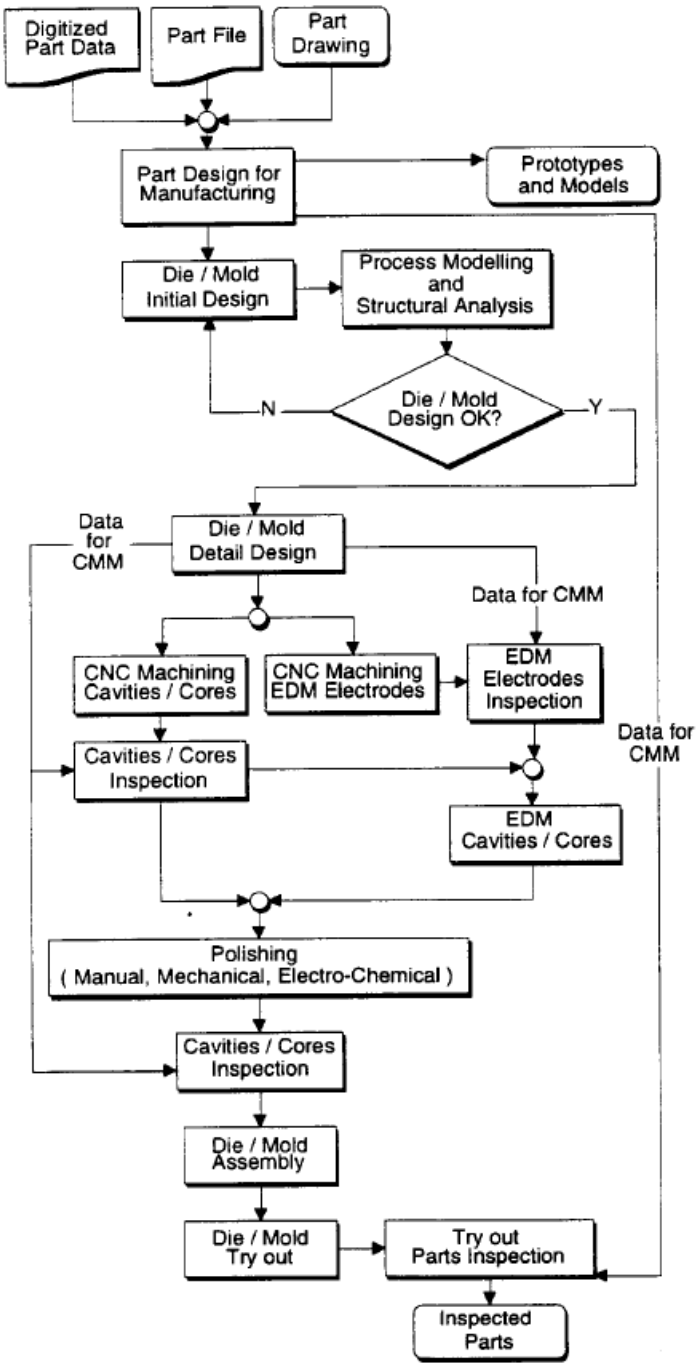

Figure 2.1 Information flow and processing steps in die/mould manufacturing [Fallböhmer, et al., 1996]

Henriques, et al. [2007] identified the challenges and opportunities faced by mould makers when digital technologies have been increasingly important in mould making industry (Figure 2.2). The paper suggests that mould makers should

extend its involvement both upstream and downstream in mould life cycle (Figure 2.3) and foresees that future mould makers will be internet-based virtual companies with remoted process planning and monitoring (Figure 2.4).

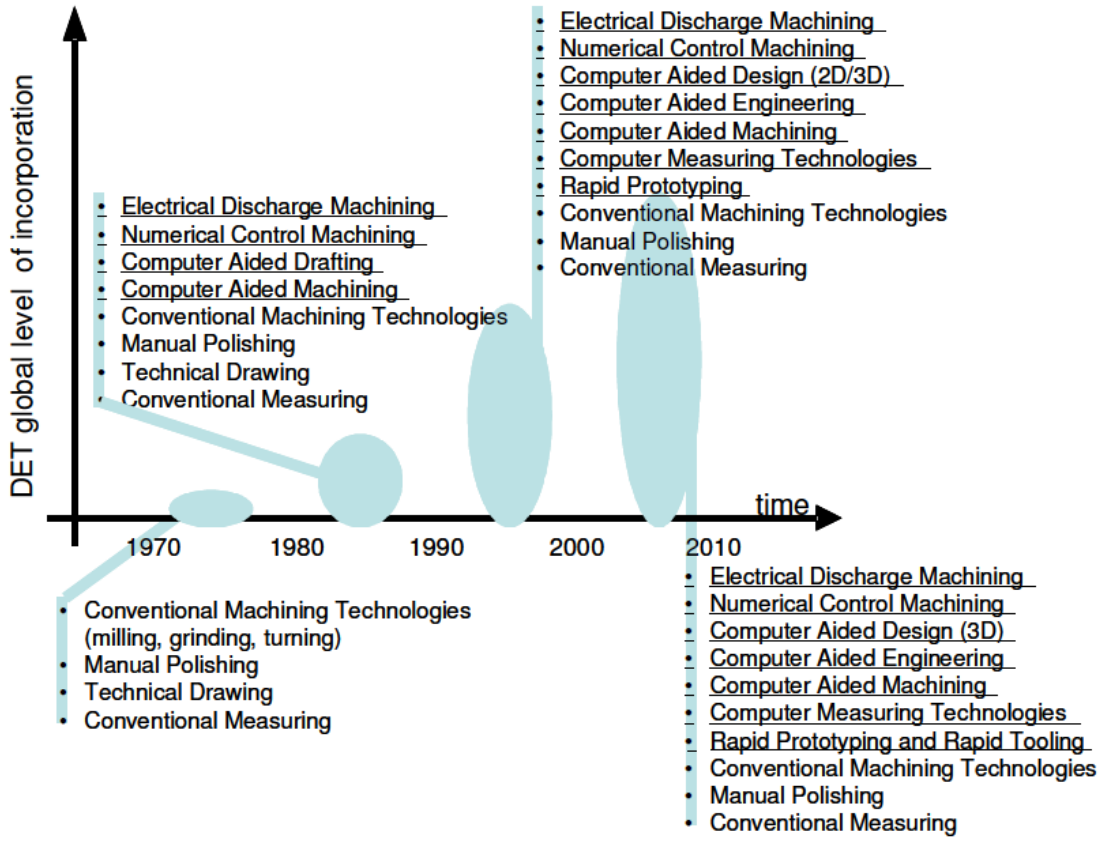

Figure 2.2Time evolution of most relevant moulds manufacturing technologies

(The technologies based on digital technologies are underlined)

[Henriques, et al., 2007]

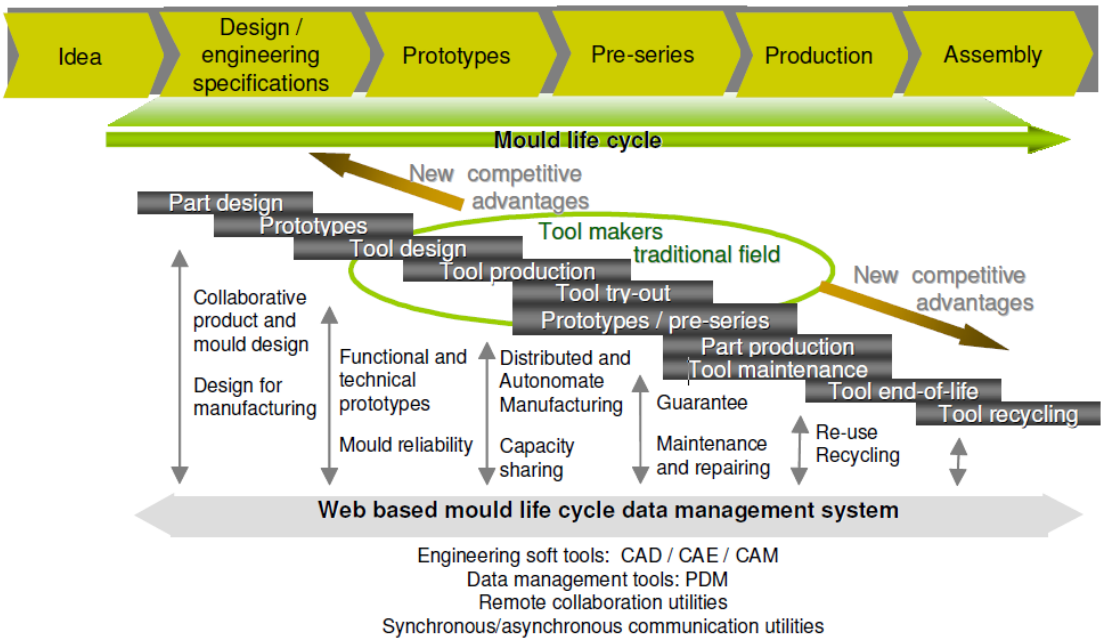

Figure 2.3Moulds life cycle management based on digital technologies [Henriques, et al., 2007]

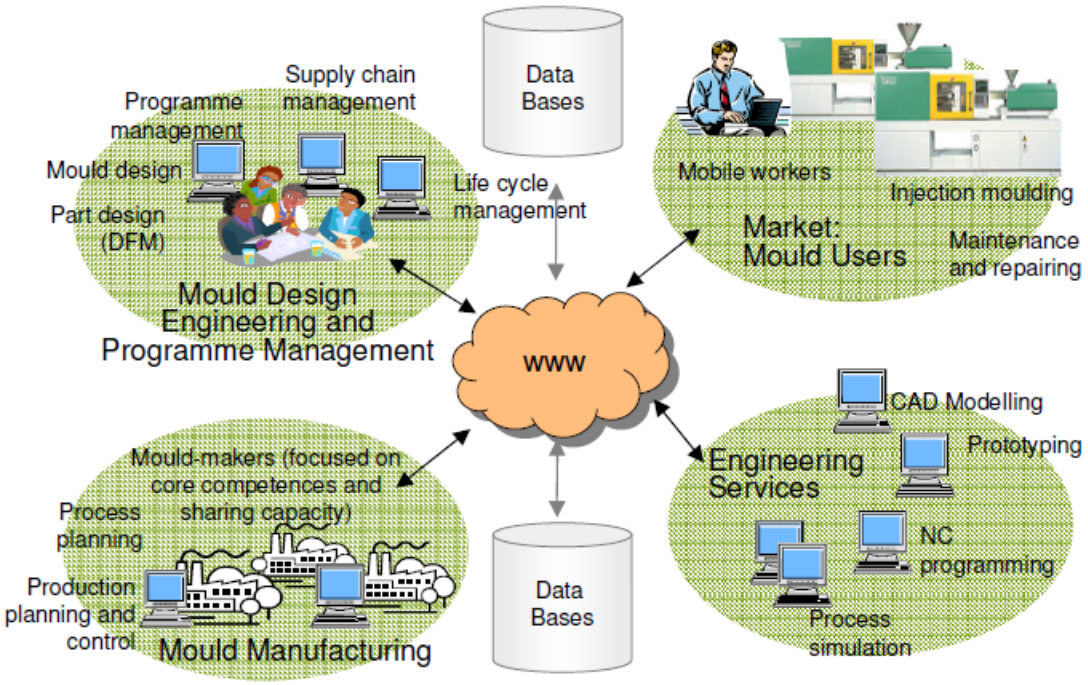

Figure 2.4Futrue digital technologies based mould making company [Henriques, et al., 2007]

Peças & Henriques [2003]pointed out that the customer requirements have become harsher. Mould makers have to accept modifications during manufacturing cycle while keeping commited final cost and due date unchanged. Application of lean manufacturing (LM) to the mould industry is proposed to encounter this challenge. However, the one-of-a-kind production characteristic of mould making hinders the implementation of LM. Most of the conventional LM operational methodologies are irrelevant to mould making activities. The paper presents an alternative operational methodologies (Table 2.1) which are said to be able to achieve the LM global objectives. These alternative operational methodologies are simultaneous engineering, internet-based manufacturing, best practices of planning and autonomation. The objectives of LM can be aligned with Asahi's strategic goals. Although implemenation of LM on company level is beyond this project's scope, a few points about best practices of planning are taken into consideration: i) Planning should not fully begin untilmould concept design is completed and approved. ii) Lead times for all the mould’s purchasedcomponents must be known and taken into consideration. iii) Theindividual scheduling andvarious skill levels of each worker must be known.

Level	Strategic	Management	Operational
Concepts and tools	• Lean thinking	• Value Stream Mapping • People empowerment • Kaizen (Continuous Improvement) • TPM (Total Productive Maintenance)	• Concurrent engineering • Internet-based manufacturing • Best-practices of planning • Autonomous
Type of Change	Cultural Change	Management Change	Procedures Change

Table 2.1 Scheme of the LM model for mould making industry

[Peças & Henriques, 2003]

An intelligent mould shop has been developed by Korean researchers Choi, et al. [2005] to enhance production efficiency and lower dependency on human skills. This intelligent mould shop is composed of three main stations: Technical Data Processing Station, Loading Schedule Station, and Real-time Monitoring Station. The framework of the transformation is illustrated in Figures 2.5 & 2.6. The research points out that while most of the required technologies are available, the collaboration from the end-users is critical. Human factors should be considered when developing a system to replace manual works.

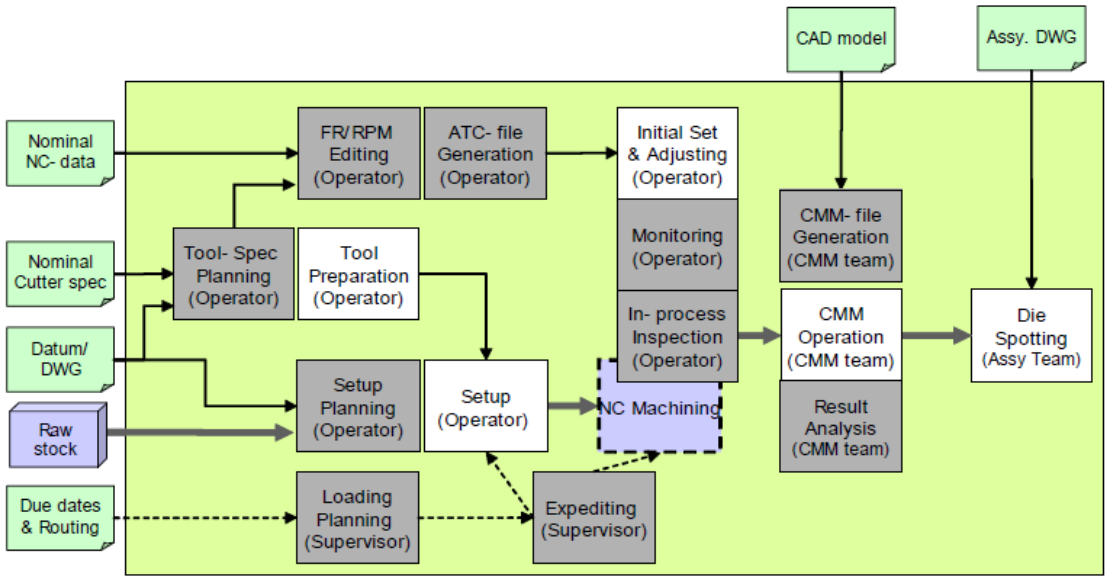

Figure 2.5 Functional model of skill-based machine shop

(Funtions which are replaced by intelligent mould shop are in grey.)

[Choi, et al., 2005]

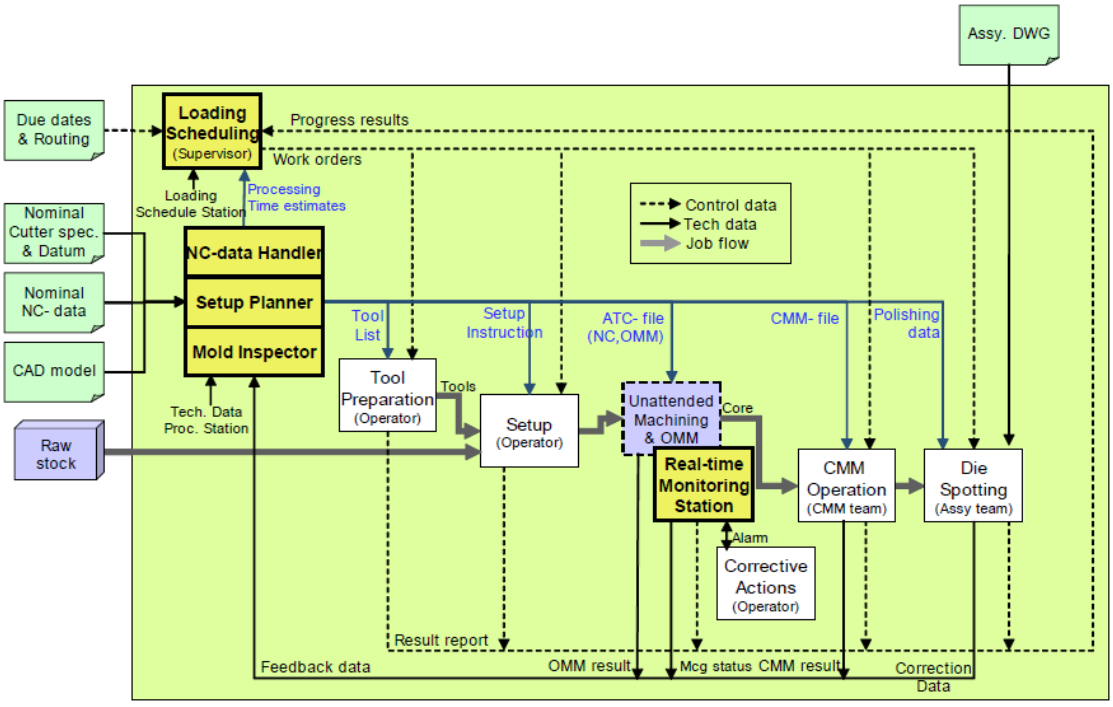

Figure 2.6 Functional model of intelligent mould shop

(Funtions which are replaced by intelligent mould shop are in yellow.)

[Choi, et al., 2005]

2.2. Product Characteristics & ProductionEnvironments

The product process continuity is classified by Sule [1997]into jumbled flow, intermittent flow, line flow and continuous flow (Figure 2.7). Mould making process should be labelled as jumled flow since there is no standard production flow for making mould components. The product characteristic of the injection mould is unique. It is the tool made for injection moulding. One mould can produce up to millions products. As the mould is one-of-a-kind, it is hard forproduction planner to predict the future workload. The materials are ordered after engineering design of a mould is finalized. It is a typical engineer-to-order production. The orgnaization of the mould shop is process specialization. The shop organizes production units according production process kind.

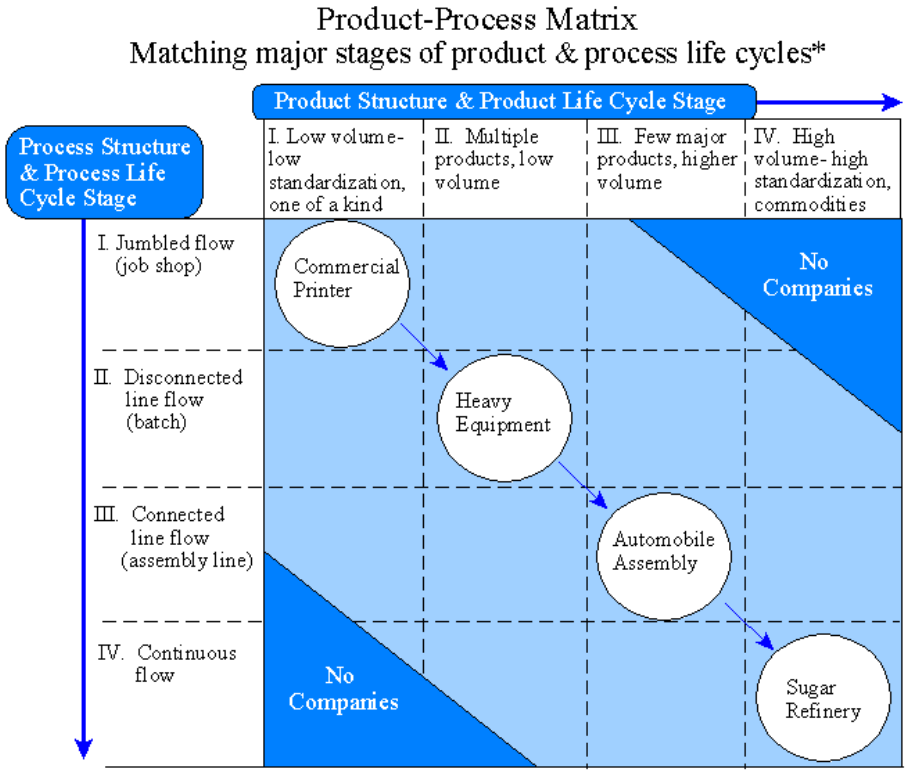

Figure 2.7 Product-process matrix [Hayes & Wheelwright., 1979]

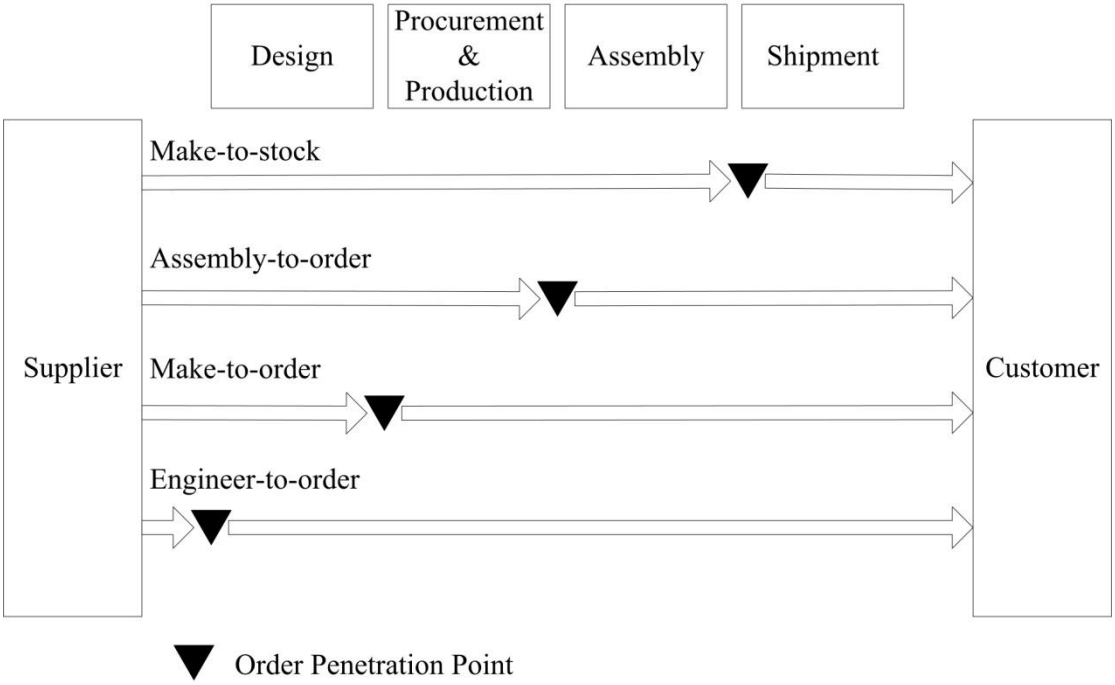

Figure 2.8Position of order penetration point in different manufacturing environment

A mould is purely customised.It is one of a kind, specially designed and made for mass producing specific parts. The manufacturing environment is classified by the position of the order penetration point (OPP) in the manufacturing value chain (Figure 2.8). There are four types of manufacturing environments in general: make-to-stock (MTS), assebmle-to-order (ATO), make-to-order(MTO) and engineer-to-order (ETO). The OPP is the point where the product is related to particular customer order [Olhager, 2003]. The upstream of OPP is forecast-driven while the downstream of OPP is customer-order-driven. The customised services provided by Asahi include mould design, engineering specification, material purchasing, component processing, assembly and delivery. Therefore, the manufacturing environment of Asahi can be classified as ETO, where most of the activities along the manufacturing value chain are hardly forecasted and customer-order-driven. Flexibility and delivery speed are the typical order winning criteria for ETO companies.

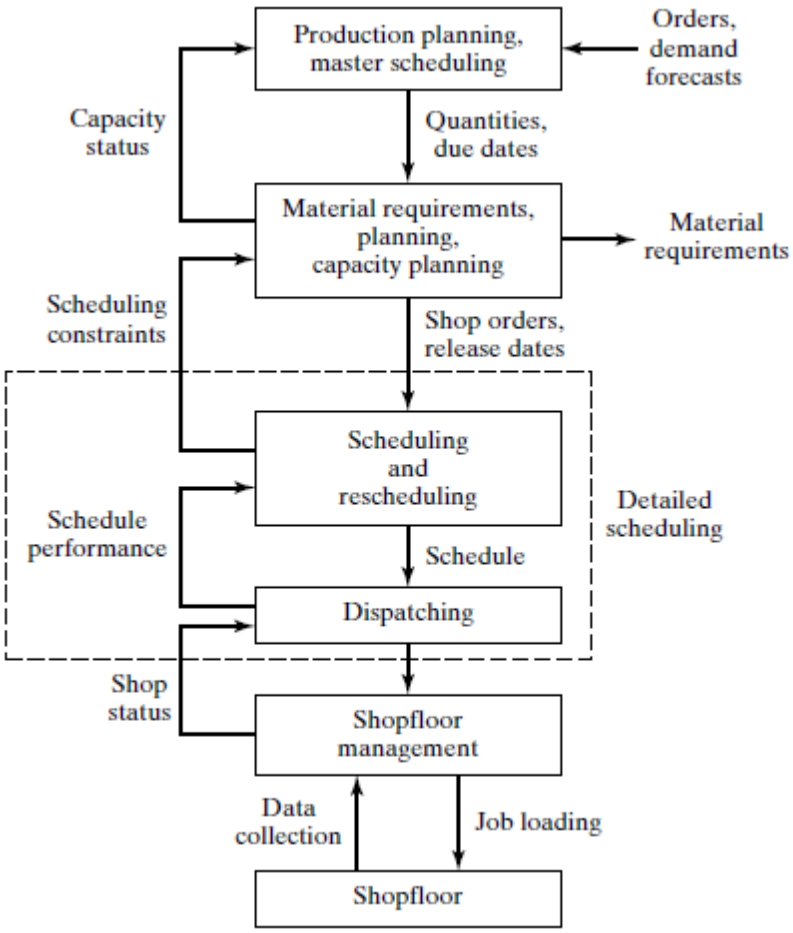

Figure 2.9 Information flow in a manufacturing system

[Pinedo M. L., 2008]

2.3. Overview of Production Scheduling Problems

Production scheduling is a decision-making process to handle the allocation of machines to operations over a specific period of time in order to achieve one or multiple objectives [Pinedo M. L., 2008]. Figure 2.9shows how the generic scheduling-related information flows in a manufacturing system.

Study of scheduling problems has been considerably increased since 1950s.

Despite scheduling is a widely studied topic, there is no single solution to tackle all the scheduling problems. A great variety of the problems in practice causes the difficulty to formulate a common model for scheduling problems. A workable algorithm for one problem may not be effective on another slightly different problem. Many algorithms have been developed to deal with the variants of the problems.

A three-field classification, $\alpha|\ \beta|\ \gamma$ [Graham, Lawler, Lenstra, & Rinnooy Kan, 1979] is commonly used to classify different scheduling problems. These three fields are: machine environment (α), job characteristics (β) and optimality criteria (γ). Brucker[2007] has further extended this classification. The possible values of each field are explained in Tables 2.2- 2.4.

Possible values	**Descriptions**	**Notations**
Single machine	Each job must be processed on a specified dedicated machine.	○
Parallel machines	Each job can be processed on each of the machines.	
- Identical parallel machines	The machines have identical processing speed.	*P*
- Uniform parallel machines	The machines have different processing speeds.	*Q*
- Unrelated parallel machines	The machines have job-dependent processing speeds.	*R*
General shop	Each job must be processed with a set of operations. Each operation can be carried out by a group of machines.	*G*
Job shop	There are precedence relations between operations. Each job has different sets of operations.	*J*
Flow shop	There are precedence relations between operations. Each job has the same set of operations.	*F*
Open shop	There are no precedence relations between operations. Each job has the same set of operations.	*O*
Mixed shop	A combination of a job shop and an open shop.	*X*

Table 2.2 Possible values of machine environment (α)

Possible values	**Descriptions**	**Notations**
Pre-emption	A job or operation is allowed to be interrupted.	*pmtn*
Precedence relations	There are precedence relations between jobs.	*prec*
Chains	The precedence relations are like chains. The outdegree and indegree for each vertex is at most one.	*chains*
Intree	The precedence relations are like a intree directed towards a root	*intree*
Outtree	The precedence relations are like a outtree directed away from a root	*outtree*
Series-parallel	The precedence relations are a series-parallel network.	*sp-graph*
Release date	The release date may be specified for each job.	*r*
Processing times	Restrictions on the processing times.	*p*
Deadline	The deadline may be specified for each job.	*d*
Batching	Jobs are grouped into batches.	
p-batching	The length of a batch is equal to the maximum of processing times of all jobs in the batch.	*p-batch*
s-batching	The length of a batch is equal to the sum of processing times of all jobs in the batch.	*s-batch*

Table 2.3Possible values of job characteristics (β)

The cost functions to be minimized	**Descriptions**	**Notations**
Maximum lateness	The greatest lateness among the jobs $(i = 1\ to\ n)$	$\max_{i=1\ to\ n} L_i$
Total tardiness	The total times of that the jobs exceed the deadlines	$\sum_{i=1\ to\ n} T_i$
Makespan (Maximum job completion time)	The completion time of the last job	$\max_{i=1\ to\ n} C_i$

Table 2.4 Possible values of optimality criteria (γ)

2.4. Production Planning & Scheduling for Mould Making

Research on planning for mould making is usually focused on the business processes rather than the fabrication processes in the mould shop.

A business information model for mould making is proposed by Nia, et al. [2007] to facilitate information flows among different systems and integration of the business processes. This model consists of four critical processes include order fulfillment, design, production and material fulfillment (Figure 2.10). It is believed that collaborations between different departments can be improved and automation of business processes can be obtained. However, inflexibility of different systems and inconsistency of information can hinder the sucessful implementation of this approach.

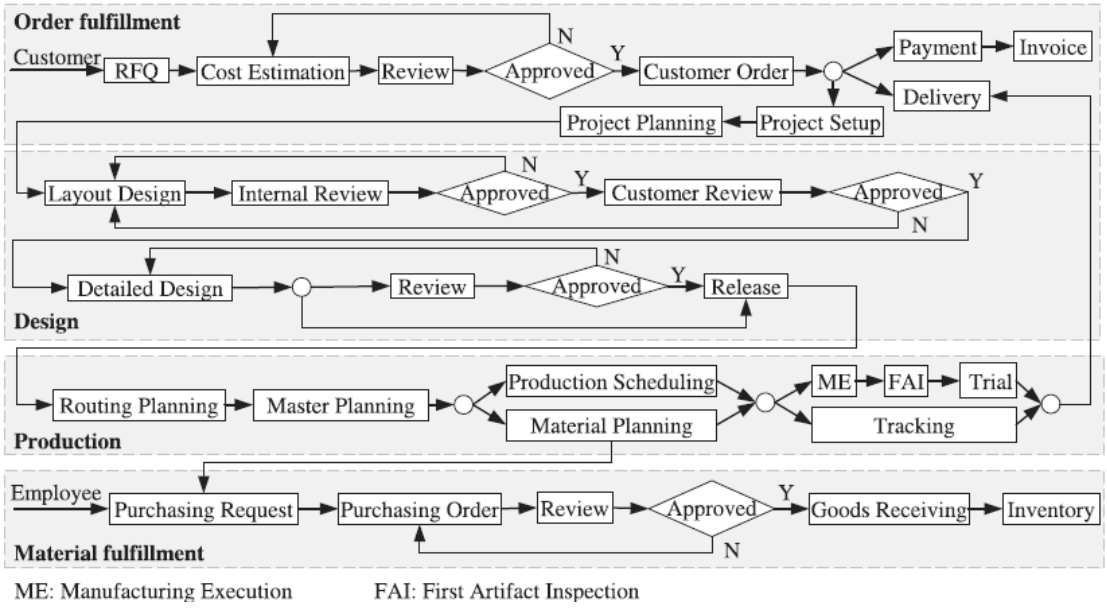

Figure 2.10Critical processes in mould making industry

A computer-based frameworkis suggested by Lee, et al. [1998]to assist concurrent mould manufacturing process planning. A knowledge base is built to assess the manufacturability of the mould so that design and planning can be carried out concurrently. The applicability of the system is highly depended on the completeness of the tailor-made knowledge base. It is time-consuming and costly for a SME to develop such system. The system only concerns the sequence of operations in each planning module without considering the feasibility of the global job sequence and the capacity of the mould shop.

Hon Hai Precision Industry Company Limited [Yeh, Cheng, Guo, Xiao, Zhong, & Liu, 2007], trading as Foxconn, has developed a system (Figure 2.11) for scheduling mould manufacturing. It is an interactive online system that allows

users to simulate, generate and transmit scheduling results. It can track and feedback the latest processes and workloads on the machines and update the workload list correspondingly. Although this system is very ideal for any mould makers, developing such a system demands long times, R&D staff, hardware and softward investment. Also, successful implementation of the system requires changing of existing workflow in coordination with all related staff. The costs and risks invloved are relatively high to SMEs.

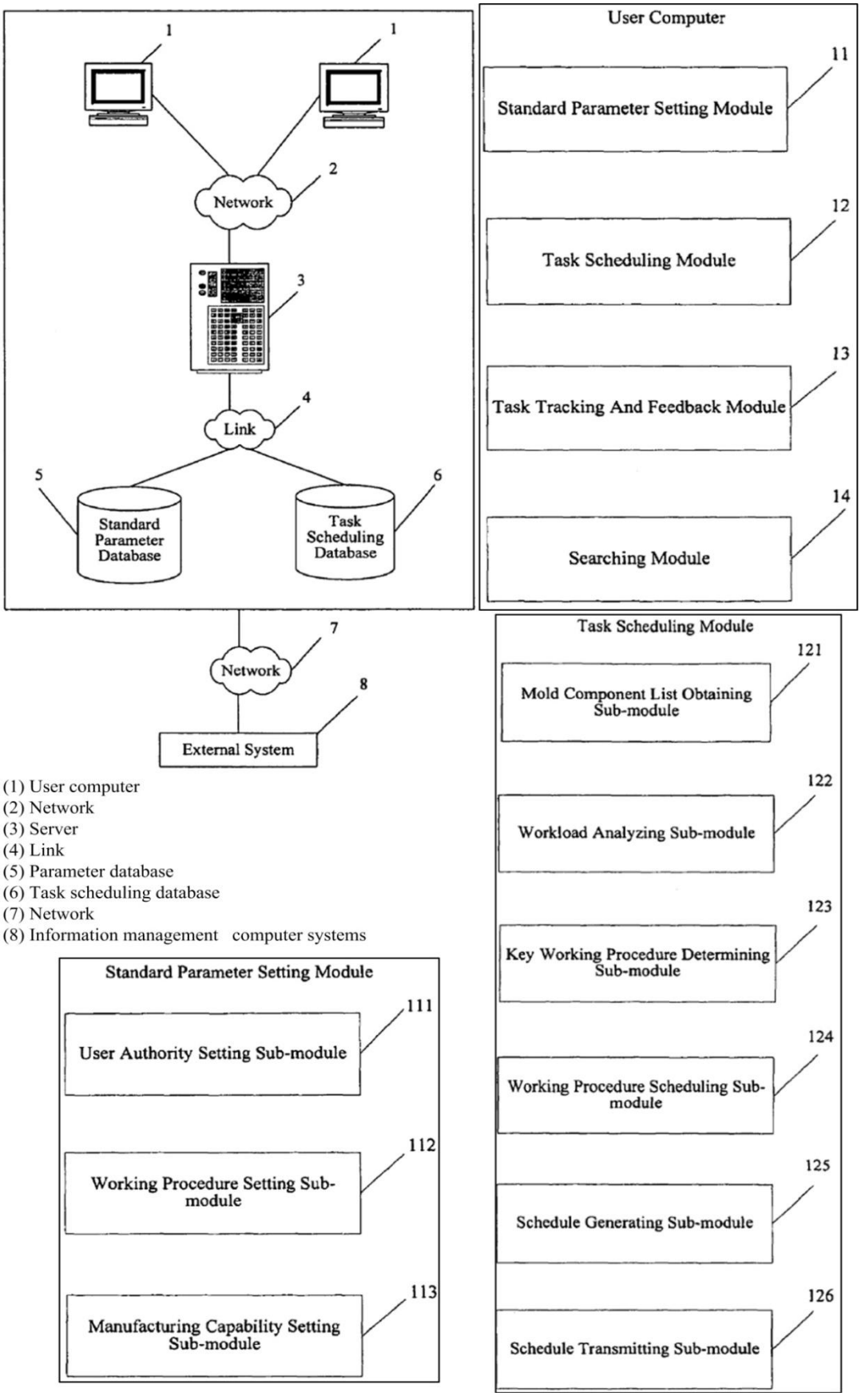

Figure 2.11 Foxconn's system design for scheduling mould manufacturing

2.5. Intelligent Production Planning & Scheduling for Mould Making

A computer-aided process planning (CAPP) system is developed by Gan, et al. [2001] to generate production plans for mould base companies with consideration of precedence constraints, available machines, tool types and machining directions. Minimization of the overall machining time is set as the objective. Branch and bound (B&B) algorithm is applied to search the optimised solution. Its performance is compared with a genetic algorithm (GA) based CAPP. The results (Figure 2.12)show that B&B algorithm can obtain comparable quality solutions with GA with a shorter computation time.

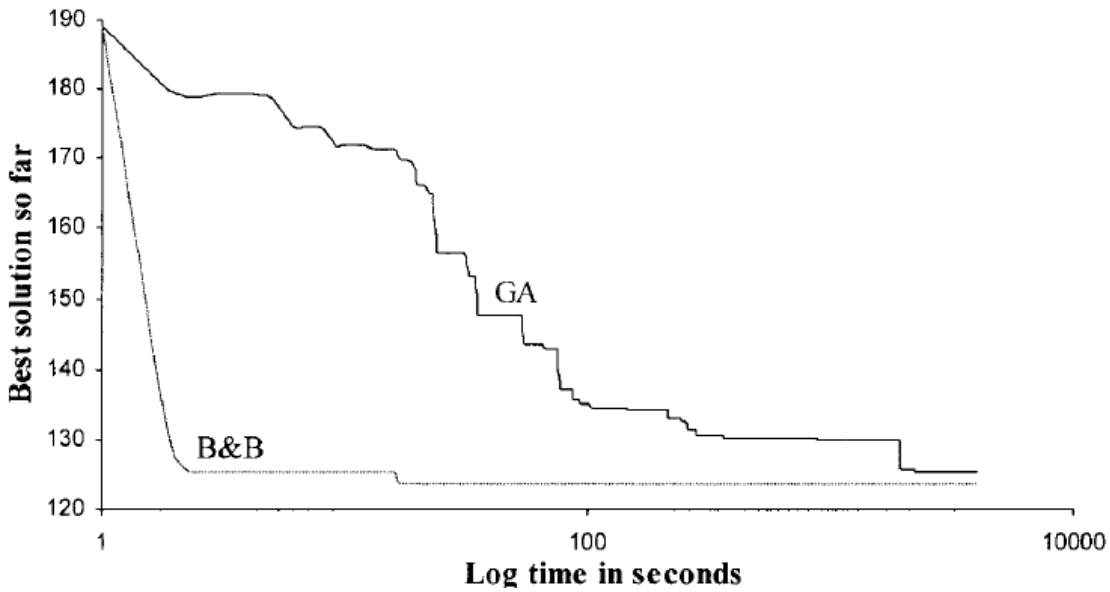

Figure 2.12Generation of solutions for B&B and GA based systems

Miyuan, et al. [2007] classified the production scheduling problem in mould

shop as a multi-mode multi-project scheduling problem. Ant colony optimization (ACO) algorithm is employed to find the schedule with minimum makespan. In this study, each mould is considered as a project and each operation can have several processing modes where processing times and costs vary. The experiment tests problem instances of two to five projects and 10, 14, 16, 20 and 30activities. The results show that 70% instances can obtain optimal results with average deviation ranging from 5% to 9%.The study highlights that parameter selection has great effect on the solution quality.

A studied problem of the mould shop [Liu, Liao, Yang, Wang, & Zhao, 2010] is classified as job shop scheduling problem. Genetic algorithm (GA) is employed to find the schedule with minimum makespan. The problem consists of seven machines and nine different production routes with 24 operations. Operation dependent transportation times and machine setup times are considered. The chromosomes are encoded in integers. Each gene represents a job, so the number of occurrences of a number is equal to the number of operations it undergoes. The operations of GA include roulette wheel method for chromosome selection, position based crossover, and two point exchange mutation. The population size is 5000, crossover probability is 0.8; and mutation probability is 0.1. The computation time used is 49.36 seconds.

Another studied problem of the mould shop [Iima, Kudo, Sannomiya, & Kobayashi, 1999] is classified as job shop scheduling problem with parallel machines, and precedence constraints among both operations and jobs. One of the parallel machines is single function machine and another one is multi-function machine. The problem has two objectives: minimize sum of tardiness and maximize the working time of the multi-function machine. An autonomous decentralized scheduling algorithm (ADSA), which is a meta-heuristic method, is suggested to solve this problem. Three problem instances are tested. The results show that the proposed algorithm performs better than a random method and a heuristic method. The advantage of this method over the other meta-heuristic methods, such as GA, is tuning of parameters is not required.

Choy[2011] proposed a hybrid scheduling decision support model (SDSM) (Figure 2.13) to solve the mould making scheduling problem. The studied problem of the mould shop is classified as job shop scheduling problem with identical parallel machines. The model is comprised of two modules: scheduling module and optimization module. The scheduling module generates the schedules with GA with the objective to minimize the makespan. The optimization module finds the most economic option to handle tardiness problem. It is proved that this model is more effective than manual scheduling.

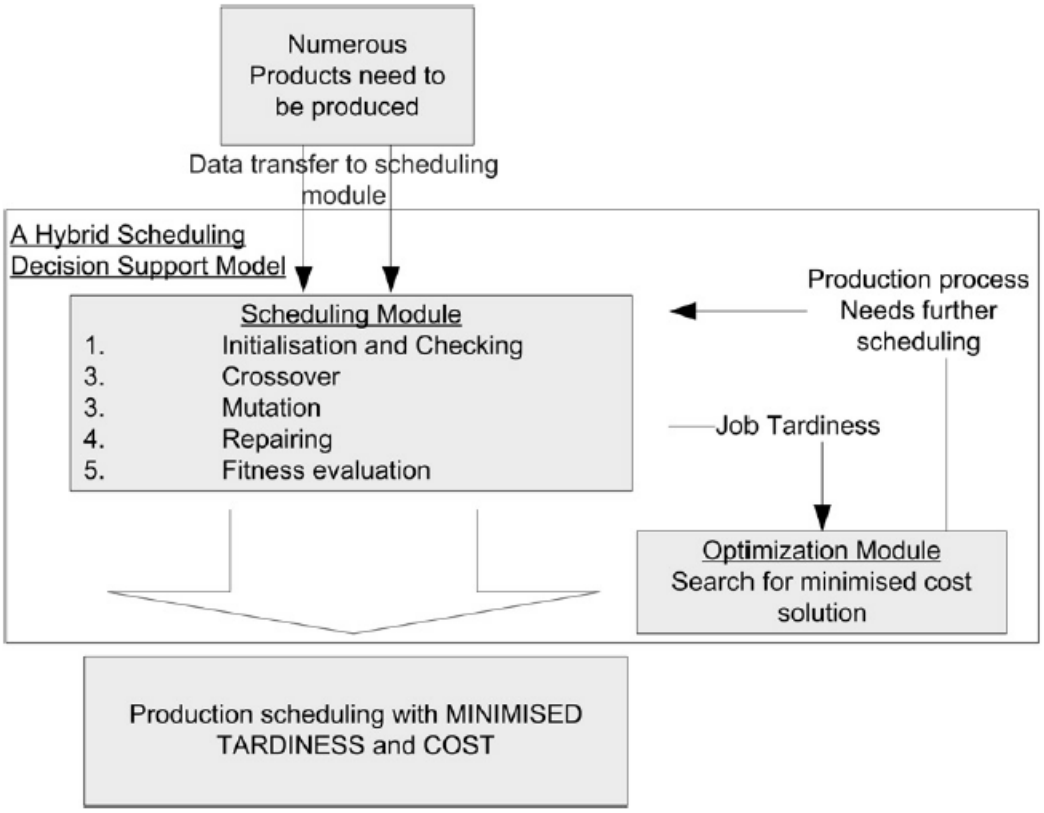

Figure 2.13 Architecture of hybrid scheduling decision support model

2.6. Production Scheduling Techniques

The production scheduling problems are usually solved by different scheduling algorithms. The scheduling algorithms [Pinedo & Chao, 1999] can be classified into two types: constructive and improvement.

The constructive type algorithms start without a schedule and gradually construct a schedule by adding one job at a time. This type of algorithms includes simple dispatching rules, composite dispatching rules, branch-and-bound and beam search. Examples of simple dispatching rules are listed as follows: Service in random order rule, Earliest release date first rule, Earliest due date first rule,

Minimum slack first rule, Shortest processing time first rule, Weighted shortest processing time first rule, Longest processing time first rule, Shortest setup time first rule and Least flexible job first rule. One of the advantages of simple dispatching rules is easy employment. No complicated calculation is needed when applying these rules. The composite dispatching rules are composed of several simple dispatching rules. Apparent tardiness cost rule is the composition of weighted shortest processing time and minimum slack rules. Apparent tardiness cost with setups rule is the composition of weighted shortest processing time minimum slack and shortest setup time rules. Branch and bound algorithm [Gan, Lee, & Zhang, 2001; Moursli & Pochet, 2000] performs better than the dispatching rules when handling NP-hard scheduling. However, the computation is very time-consuming when the problem is large. Beam search consumes less computation time as only the most promising nodes is searched.

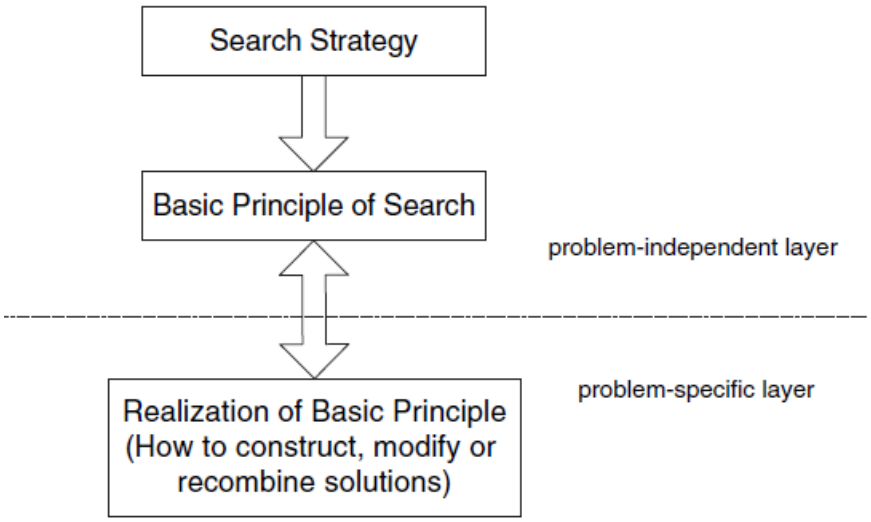

Figure 2.14 Working principle of meta-heuristic algorithms [Zäpfe, Braune, & Bögl, 2010]

The improvement type algorithms start with a complete schedule, which may be selected arbitrarily, and try to obtain a better schedule by manipulating the current schedule. This kind of algorithms is also called meta-heuristic algorithms (Figure 2.14). The high-level strategy is problem-independent and can be seen as an abstract framework. The high-level strategy has to properly realize a balance between the two forces: intensification and diversification.

2.7. Rescheduling Techniques

Herrmann, et al. [2006] suggests a rescheduling framework to improve production scheduling in response to disruptions.

In the dynamic manufacturing environment, rescheduling is essential to reduce the effects of the unexpected events，which are named rescheduling factors. They include machine breakdowns, job delays, job cancellations, job insertions, etc.

The rescheduling framework consists of rescheduling environments, rescheduling strategies, rescheduling policies, and rescheduling methods.

The rescheduling environment (Table 2.5) identifies the set of jobs that need to be scheduled.

Static (finite set of jobs)		Dynamic (infinite set of jobs)		
Deterministic (all information given)	Stochastic (some information uncertain)	No arrival variability (cycle production)	Arrival variability (flow shop)	Process flow variability (job shop

Table 2.5 Rescheduling environment

A rescheduling policy specifies when rescheduling should occur. The policies define the following aspects:

(1) Rescheduling point: the point in time when rescheduling is triggered.

(2) Rescheduling period and frequency: the time between two consecutive scheduling points

(3) Scheduling stability: The number of revisions made on a schedule after its execution

A rescheduling strategy describes whether or not production schedules are generated.

Dynamic (no schedule)		Predictive-reactive (generate and update)		
Dispatching rules	Control-theoretic	Rescheduling policies		
		Periodic	Event-driven	Hybrid

Table 2.6 Rescheduling strategies

There are two rescheduling strategies (Table 2.6) to deal with uncertain job arrivals: dynamic scheduling and predictive-reactive scheduling.

Dynamic scheduling dispatches jobs from the queue when a machine becomes available. Dispatching rules or other heuristics, such as Shortest Processing Time and Earliest Due Date, are generally applied to obtain the job queues.

Predictive-reactive scheduling first generates a schedule and then reviews the original schedule in response to a disruption. When to trigger rescheduling depends on the rescheduling policies employed. There are three typical rescheduling policies associates with this strategy: periodic, event-driven and hybrid. Periodic policy updates the schedule at regular intervals. This policy is more suitable for offline scheduling environment when real-time progress data is not available. However, it is difficult to determine the optimal rescheduling interval. Event-driven policy revises the existing schedule when a certain events occur, for example, machine breakdowns. Since the number of revisions is unpredictable and can be excessive, a reliable real-time progress monitoring system must be provided in order to give immediate response to the disruptions. Comparing with periodic policy, event-driven policy offers schedules with higher

nervousness but lower stability.

Hybrid policy combines the characteristics of periodic policy and event-driven policy. The schedule is updated periodically and also under a certain events.

Schedule generation		Schedule repair		
Nominal schedules	Robust schedules	Right-shift rescheduling	Partial rescheduling	Complete regeneration

Table 2.7 Rescheduling methods

Rescheduling methods (Table 2.7) describe how schedules are generated and updated.

Robust schedules consider the possible occurrences of changes on the schedule while Nominal schedules do not consider this situation. Robust schedules intend to generate a schedule that can perform well even if adjustments have to be made. This adjustment is called "schedule repair". Normally, idle time is included in the schedule to serve as the buffer for any disruptions in order to keep the schedule robust. The three common schedule repair methods are: right-shift rescheduling, partial rescheduling, and complete regeneration. Right-shift

rescheduling postpones the succeeding operations when the disruption happens. Partial rescheduling rearranges the affected operations only and makes minimum changes on original schedule. Complete regeneration reschedules everything after the rescheduling point.

Efficiency, stability and cost are the three common performance measures of rescheduling.

Schedule efficiency is measured by time-based objectives, such as makespan, mean tardiness, mean flow time, average resource utilization, and maximum lateness.

Schedule stability is measured by starting time deviations and the sequence difference between the updated schedule and the initial schedule.

Rescheduling costs include computation costs, setup costs, and transportation costs. Computational costs are any costs involved during new schedule computing, for instance, the time spending on schedule update and computation calculation. Setup costs are incurred when the tooling has to be reallocated after rescheduling. Transportation costs occur when extra material handling work has to be done according to new schedule.

For mould making, the mould components are considered to be the jobs in mould shop. Although there is a finite set of components in each mould, these components are not processed mould by mould. Instead, the mould shop continues receiving processing orders from different moulds. Therefore, the rescheduling environment can be defined as dynamic. The job arrivals are uncertain since some materials are non-stock and wait for delivery. The processing routes vary by component types. Components of the same type have similar processing route. Job outsourcing and overtime work are considered when necessary.

Considering the available resources in Asahi, there is no sophisticated system which can provide shop floor progress data in real-time. Therefore, rescheduling methods which require online data and frequent updates are not feasible to implement in Asahi.

2.8. Theory of Constraints

The theory of constraints (TOC) [Goldratt, E.M., 1993] suggests that constraints determine system performance. The constraint resources should operate at full capacity. Only by relaxing the constraints can a production system be improved. There are five basic steps[Fawcett & Pearson, 1991]: 1) identify the constraint resources; 2) establish key buffer locations to protect throughput; 3)

schedule the constraint resource; 4) release materials at gateway operations to maximize production at the constraint resource; and 5) use forward scheduling of work centers that follow the constraint resource.

Fredendall & Lea [1997] has studied the application of TOC in master production scheduling. Graham [2000] points out that the PERT/CPM approach can cause project overrun as people misuse the safety buffer time (Figure 2.15). They tend to start the activity as late as possible. TOC applies the buffer at the end of the whole project, so no feeding buffer in between activities.

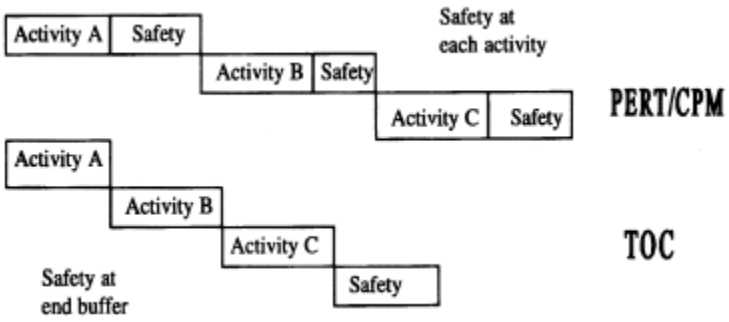

Figure 2.15 Comparison between PERT/CPM and TOC with regard to safety time

Under tight project schedule, TOC approach is better for agile mould manufacturing. All activities should finish as soon as possible instead of leaving buffer time between operations.

Chapter 3 Methodologies

3.1. Problem Background

3.1.1. Optimality criteria

As mould making is only one of the activities along the whole project, due date for Asahi mould shop can be re-arranged flexibly. Instead of meeting the due date, minimization of the makespan of mould making can provide more buffer time for the whole project. Therefore, much more spare time can be left for contingency use.

Among the various components of a mold, mould inserts normally undergo the most processing operations. The production route for this kind of components usually is the critical path for a mould production. According to TOC, minimization of the critical path is the best way to improve the whole system's performance. The main concern is how to shorten the makespan of these key components. The makespan of the key components would be taken as the reference for entire project scheduling.

3.1.2. Machine Environment

In Asahi mould shop, the machines with same functions are grouped together. Table 3.1 shows the resources of Asahi mould shop allocated for processing key components. The numbers of available machines in each processing groups are

different. All the machines in the same group are identical except Group D. There are two advanced machines with higher performance. It is estimated that the processing speed for these two machines is about 1.5 faster than the old machines.

Processing Groups	**Machine types**	**Number of machines**	**Processing speed ratio**
A	Manual-controlled Milling	5	NA
B	Deep hole drilling	1	NA
C	Grinding	5	NA
D	Program-controlled Milling	2	1
		2	1.5
E	Wire-cutting	1	1
		1	1.5
F	Die-sinking EDM	5	NA
G	Polishing	4	NA

Table 3.1 Processing groups for key mould components production in Asahi's mould shop

The scheduling problem in Asahi mould shop can be classified as a flexible flow shop, also known as flexible flow line, hybrid flow shop, flow shop with parallel machines or multiprocessor flow shop. In flexible flow shop scheduling problem (FFSSP), there are parallel machines in a series of production stages. These parallel machines can be identical or unrelated. The number of machines in each stage can vary. Figure 3.1illustrates an example of flexible flow shop. Suppose that there are two machines at Stage 1, three machines at Stage 3, and two machines at Stage 3. A job can be processed on any one and at most one machine at

each stage. After finishing the operation, it is transported to the buffer area until there is any free machine at the next stage. Each job has to follow the same production route but it can bypass the stage if it needs not be processed on that stage. For example, a job can go through Stage 1, bypasses Stage 2, and go directly to Stage 3.

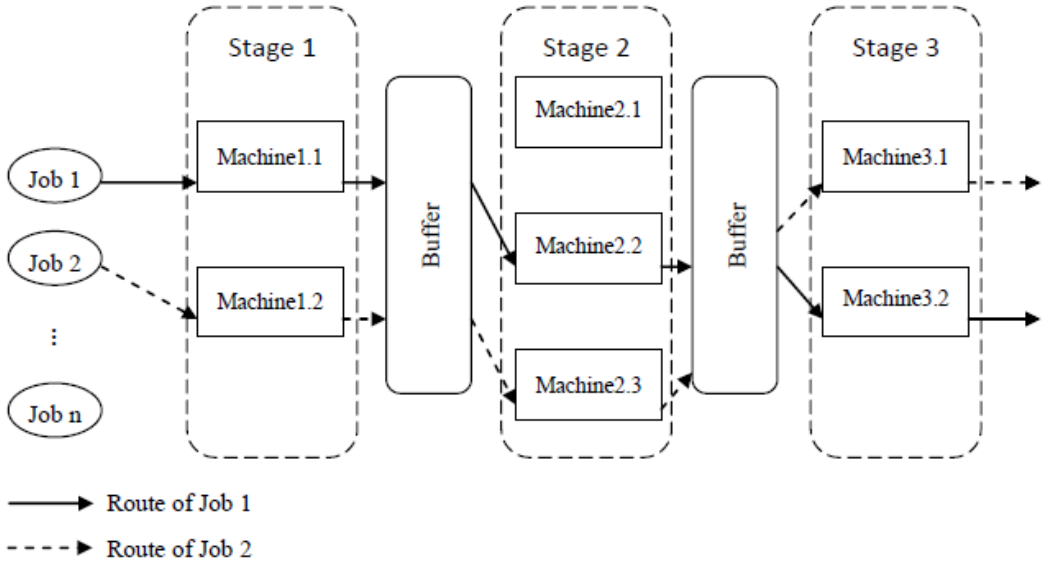

Figure 3.1AThree-stage Flexible Flow Shop

3.1.3. Job Characteristics

Based on the company's current situation, the number of mould inserts for a mould ranges from 2 to 6. For each project, the number of moulds required ranges from 2 to 10. After undergoing each process, quality checking is needed. Transportation and set-up time should be considered when scheduling the jobs. An estimated sum of preparation time before each process is set. This preparation time is for quality checking, transportation and setup.

Table 3.2 shows the range of processing times, the occurrences of the process, the range of processing times and preparation times. The value of occurrence ranges from 0 to 1, in which 0 means none of inserts undergo this process; 1 means all of the jobs must undergo this process.

Processing Groups	Occurrences	Processing times (H)	Preparation times (H)
A	1	0.25-4	0.25
B	0.5	0.25-4	0.25
C	1	0.25-6	0.25
D	1	0.5-12	0.5
E	0.5	1-4	0.25
F	1	0.5-13	0.5
G	1	0.25-8	0.25

Table 3.2 General processing sequence of key mould components

3.2. Problem formulation

The FFSSP is proved to be NP-hard, which has not been solved optimality in a reasonable time. It consists of two sub-problems:1) determining the Entry Point Sequence (EPS) of the jobs at the beginning of production; 2) selecting the machine to process next job.

The problem studied in this paper is the non-permutation FFSSP, which allows jobs to re-sequence before entering the next stage. An integrated approach

of heuristic and random keys representation is proposed to minimize the makespan which is the completion time of the last job. The entry point sequence [Kochhar, Morris, & Wong, 1988; Riane, 1998] is the order of jobs entering the first stage. Random keys [Bean, 1994] in each harmony vector are converted into job permutation, which is the entry point sequence. The jobs are then dispatched in the later stages according FIFO rule [Santos, Hunsucker, & Deal, 1996]. Once a job has finished its operation, it lines up at the end of the queue towards the next stage at the buffer area for dispatching to the next available machine.

The problem considered in this paper is formulated as follows. There is a set J of independent jobs,$J = \{1, \ldots, n\}$, which have no precedence constraint. Each Job $j \in J$ has to undergo a set O of operation stages, $O = \{1, \ldots, l\}$. At every stage $i \in O$, there is a set $M_i = \{1, \ldots, m_i\}$ of uniform parallel machines where $m_i \geq 1$. A Job j requiresS_i setup time andU_{ij} standard processing time to finish the operation at Stage i.If job $j$skips Stage $e$mits standard processing time$U_{ij} = 0$. The setup is non-anticipatory; therefore it can begin only if both job and machine are available. The processing speed of Machine k at Stage i is$V_{i,k}$where $k \in M_i$, $V_{i,k} \geq 1$. Every job and machine is available at *time 0*. Each machine can process at most one job at a time. A job can start setup and processing at next stage only after it had finished the operation at the previous stage. The buffer between any two consecutive stages is assumed to be unlimited.

3.3. Entry Point Sequence

In FFSSP, there are multiple stages. The entry point sequence of the jobs in the first stage would highly affect the job sequence in subsequent stages. The following algorithms are proposed to find out the best EPS.

3.3.1. Simple Dispatching Rules

The simple dispatching rules can be categorized into process-time based rules, due-date based rules and combination rules. As the objective of this paper is to minimize the makespan, process-time based rules are chosen.

Abbr.	Rules	Explanations
ERT	Earliest release time	The job which is released first has highest priority.
SPT	Shortest processing time	The job with the shortest total processing time is assigned first.
LPT	Longest processing time	The job with the longest total processing time is assigned first.

Table 3.3 Process-time based dispatching rules

3.3.2. Hybrid NEH

NEH (Nawaz, Enscore, & Ham, 1983) is a constructive heuristic algorithm for solving permutation flow shop scheduling problem (PFSSP). In PFSSP, there is only one machine in every stage. All jobs have to undergo each stage following initial entry sequence. Jobs are sequenced in the same order throughout the chain of stages. NEH is developed based on Longest Process Time First (LPT) rule. Jobs are sorted according to the non-increasing order of their total processing times. The first two jobs are selected to form a partial sequence. A best sequence is found after considering all possible sequences. The next job is inserted into the partial sequence without changing its relative job positions. The new partial sequence is formed after considering all possible insertion positions. The final sequence is obtained after all the jobs are inserted into the partial sequence.

The following is an example showing how to solve PFSSP by NEH. Table 4shows the processing times of three jobs which have to undergo a 3-stage processing.

Processing Time (PT)		**Job**		
		1	**2**	**3**
Stage	**1**	12	7	5
	2	7	9	10
	3	4	6	9
Total Processing Time		23	22	24

Table 3.4 Processing times of the PFSSP

The sequence of the jobs would be {Job 3, Job 1, Job 2} after sorting in non-increasing order of their total processing times. The first two jobs, Job 3 & Job 1, are selected to form a partial sequence.

There are two possible sequences: {Job 3, Job 1} and {Job 1, Job 3}. The former sequence would give a schedule with makespan of 28 while the latter one's makespan is 38. As a result, the sequence {Job 3, Job 1} is selected.

		Job 3			Job 1		
		PT	**Start**	**End**	**PT**	**Start**	**End**
Stage	**1**	5	0	5	12	5	17
	2	10	5	15	7	17	24
	3	9	15	24	4	24	28

Table 3.5Processing times (PT), start time and end time of the jobs at each stage when Job 3 is firstly sequenced

		Job 1			Job 3		
		PT	**Start**	**End**	**PT**	**Start**	**End**
Stage	1	12	0	12	5	12	17
	2	7	12	19	10	19	29
	3	4	19	23	9	29	38

Table 3.6 Processing times (PT), start time and end time of the jobs at each stage when Job 1 is firstly sequenced

The remaining job to be inserted is Job 2. There are three possible insertion positions. The makespans of these three possible sequences is given in Table 3.6. Among the three sequences, sequence {Job 3, Job *2*, Job 1} gives the shortest makespan.

Sequence	Makespan
Job 2, Job 3, Job 1	41
Job 3, *Job 2*, Job 1	35
Job 3, Job 1, *Job 2*	39

Table 3.6 Makespan of the three possible sequences after inserting Job 2

A hybrid NEH is proposed to solve the FFSSP stated in this paper. Instead of using LPT rules only, the other two dispatching rules, Earliest Release Time (ERT) and Shortest Processing Time (SPT) are also considered. The jobs are firstly sequenced according one of the above rules. Then, the job sequence is built step by step based on insertion rule of classic NEH. If there is more than a sequence having the best result, the one which is considered first would be selected.

3.3.3. Random Keys Harmony Search

Harmony search (HS) is a music-inspired meta-heuristic algorithm imitating jazz improvisation [Geem, Kim, & Loganathan, 2001]. The perfectness of the harmony is determined by the audio aesthetic standard in improvisation whereas the fitness of the solution of optimization problem is determined by the objective function. Although it is a newly developed algorithm, it has been proved to be effective in solving different engineering problems including flow shop scheduling [Wang, Pan, & Tasgetiren, 2010], vehicle routing [Geem, Lee, & Park,

2005], structural optimization [Lee & Geem, 2004], task assignment problem [Zou, Gao, Li, Wu, & Wang, 2010],and material properties determination [Mun & Geem, 2009]. However, application on solving flexible flow shop scheduling problem has not been addressed yet.

The HS algorithm can be simplified as the following five steps: (1) parameter initialization; (2) harmony memory initialization; (3) new harmony improvisation; (4) harmony memory update; and (5) termination criterion check. Firstly, the entry point sequence of jobs is created by HS. The number of decision variables for HS is the number of jobs. Each harmony vector represents one set of variables. The fitness of the harmony vector is determined by the makespan of the schedule: the shorter makespan, the better the fitness.

(1) Parameter initialization

The HS algorithm parameters are specified as follows:

N Number of decision variables

HMS Harmony memory size, i.e., number of harmony vectors in HM

HMCR Harmony memory considering rate, i.e., the possibility of randomly choosing one value from HM

PAR Pitch adjusting rate, i.e., the rate of adjusting the value chosen from HM

BW Bandwidth, i.e., the range of adjusting value

(2) Harmony memory initialization

The initial HM matrix is formed by standard uniformly distributed random numbers (0,1). Each row of HM represents one harmony vector. Z is a vector recording the fitness of the respective harmony vector.

$$\left[\begin{array}{cccc|c} X_1^1 & X_2^1 & \cdots & X_{n-1}^1 \quad X_n^1 & Z^1 \\ X_1^2 & X_2^2 & \cdots & X_{n-1}^2 \quad X_n^2 & Z^2 \\ \vdots & \vdots & \ddots & \vdots \quad \vdots & \vdots \\ X_1^{HMS-1} & X_2^{HMS-1} & \cdots & X_{n-1}^{HMS-1} \quad X_n^{HMS-1} & Z^{HMS-1} \\ X_1^{HMS} & X_2^{1HMS} & \cdots & X_{n-1}^{HMS} \quad X_n^{HMS} & Z^{HMS} \end{array}\right]$$

(3) New harmony improvisation

The improvisation entails three rules: memory consideration, pitch adjustment, and randomization. The value of the new harmony vector is opted one by one. First, a uniform random number is generated. If the number is smaller than *HMCR*, a value is chosen from HM, otherwise, a random number is chosen from the entire feasible range. The pitch adjustment is executed only if the value is chosen from HM. Thus, the probability of pitch adjustment is ($HMCR \times PAR$) while the range of the adjustment is $\pm BW/2$. To overcome the problem of exceeding the feasible range of the decision variable after adjustment, the modulus of the value is divided by the difference of the upper and the lower bounds of the feasible range. The following pseudo-code depicts the improvisation process.

for $j = 1\ toN$ **do**

$rand1 = rand(0,1)$

if $rand1 < HMCR$ **then**

$rand2 = randi(1, HMS)$

$X_j^{new} = X_j^{rand2}$

$rand3 = rand(0,1)$

if $rand3 < PAR$ **then**

$X_j^{new} = \left(X_j^{new} + (rand(-0.5,0.5) \times BW\right) mod\ 1$

end if

else

$X_j^{new} = rand(0,1)$

end if

end for j

// $rand(x, y)$ is a random number drawn from the standard uniform distribution on (x, y).

// $randi(x, y)$ is a random integer drawn from the standard uniform distribution on (x, y).

(4) Harmony memory update

If the fitness of the new harmony vector X^{new} is better than that of the worst harmony vector in the HM, replace the worst harmony vector with it in the HM.

(5) Termination criterion check

If the stopping criterion is satisfied, computation is terminated. Otherwise, Steps 3 and 4 are repeated.

3.3.4. Random Keys Genetic Algorithm

GA is one of the most popular meta-heuristic algorithms. It has been studied for nearly half of a century. However, there is an offspring feasibility problem when using traditional binary encoding (Table 3.7) and permutation encoding method (Table 3.8). It is difficult to decode into feasible solution for scheduling problems. The offspring produced may be infeasible to decode into solution.

Chromosome 1	1	0	0	0	1	0	0	1
Chromosome 2	0	0	1	1	1	0	1	1

Table 3.7 Binary encoding of chromosomes

Chromosome 1	1	3	2	8	4	7	6	5
Chromosome 2	2	8	4	7	1	3	5	6

Table 3.8 Permutation encoding of chromosomes

In traditional method, crossover is performed in single or multiple points. A single or multiple crossover points on both parents' chromosome are selected (Figure 3.2). Take the chromosomes in Table 3.8 as an example; the numbers in the children chromosomes cannot form a feasible sequence after crossover. Subsequent decoding operations have to be done in order to get the feasible solutions.

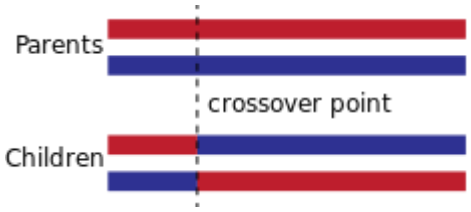

Figure 3.2 Single-point crossover

Chromosome 1	1	3	2	8	4	7	6	5
Chromosome 2	2	8	4	7	1	3	5	6

Parent

Chromosome 1	1	3	2	7	1	3	5	6
Chromosome 2	2	8	4	8	4	7	6	5

Children

Figure 3.3 Single-point crossover of permutation encoding chromosomes

Random Keys Genetic Algorithm (RKGA) is a variant of GA proposed by (Bean, 1994).This approach adopts a random search in the solution space. The chromosome space is fixed within [0,1].

This algorithm has been used for solving different scheduling problems, including single machine scheduling problem, multiple machine scheduling problem and resource constrained project scheduling problem[Mendes, Gonçalves, & Resende, 2009].One of the advantages of random keys is any

sequence of numbers can be decoded into a sequence. All offspring can be regarded as feasible solutions. This cannot be done when adopting binary and integer encoding methods. Genes can be duplicated after crossover.

There are three basic rules when employing random keys:

i. Form each chromosome by generating random numbers for each decision.

ii. From a given chromosome, derive a solution by sorting the random keys and taking the priorities from the sort.

iii. All crossovers are done on the random keys, not the derived solutions.

Like other GAs, RKGA consists of three operations: reproduction, crossover, and mutation.

(1) Reproduction

The reproduction is based on an elitist strategy. The best individuals from current generation are selected and passed down to the next generation so that conservation of the best individuals can be guaranteed. The potential drawback of population convergence can be tackled by high immigration rate.

These are the parameters which have to be initialized at the beginning.

N Number of decision variables

POP Population size, i.e. number of chromosomes

TOP The percentage of population from the top to be transited every generation.

BOT The percentage of population from the bottom to be eliminated every generation.

CX Crossover probability

(2) Crossover

Parameterized uniform crossovers are adopted rather than the conventional one-point or multiple-point crossover. One individual from the elitist group and one individual from the remaining group, excluding elitist group, is chosen as parents. For each gene, a random number in the interval [0, 1] is generated. If this number is smaller than the crossover probability, the allele of the first parent is kept; otherwise, the allele of the second parent is kept instead.

Example of parameterized uniform crossover is given in Table 3.7. The crossover probability in this example is equal to 0.6. For each gene, a random number in the interval [0, 1] is generated. When the value of the random number is smaller than 0.6, the offspring inherits the gene from Chromosome 1. Otherwise, it inherits from Chromosome 2.

Chromosome 1	0.89	0.74	0.77	0.83
Chromosome 2	0.66	0.76	0.90	0.81
Random number	0.32	0.05	0.85	0.90
Relation to crossover probability of 0.6	<	<	>	>
Offspring chromosome	0.89	0.74	0.9	0.81

Table 3.9 Example of parameterized uniform crossover

(3) Immigration (Mutation)

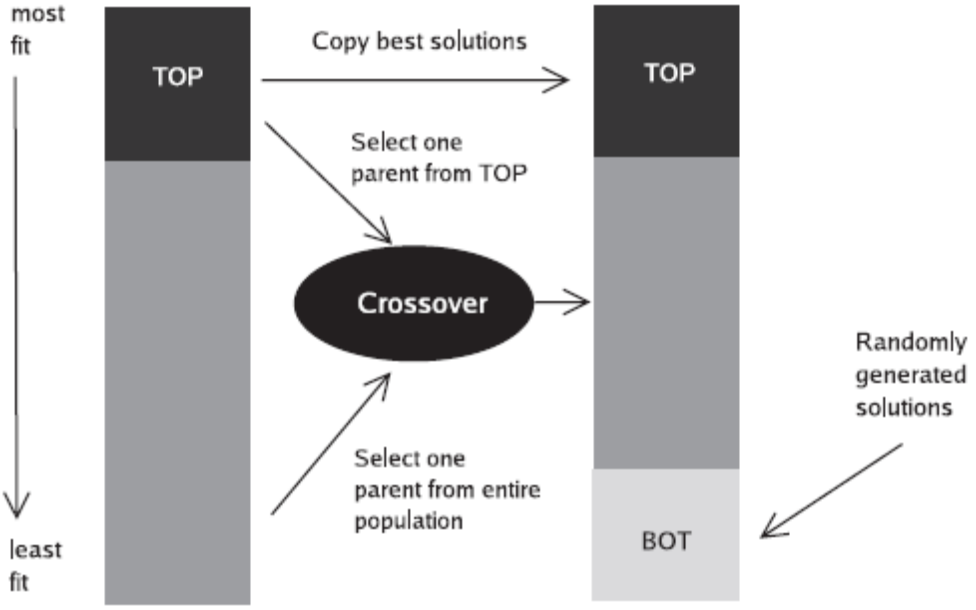

Figure 3.4 Parent selections for crossover

In RKGA, mutation operator is substituted by immigration operator to prevent premature convergence of the population. A group of new individuals are randomly generated from the same distribution as the original population. Figure 3.2 depicts the entire generation transition. The top population can be kept while the bottom population is eliminated. New immigrants, who do not carry any

genetic material of current population, are introduced. This approach can prevent premature convergence of the population.

Table 3.8 shows an example of solution vector. Each position index j represents a job. After permutation, the elements in the vector are sorted according to their values in ascending order as shown in Table 3.9. The entry point sequence generated by this vector is$\langle 3,5,6,2,1,4,7 \rangle$.

j	1	2	3	4	5	6	7
X_j^{new}	0.74	0.58	0.14	0.82	0.29	0.31	0.92

Table 3.10 Solution vector X^{new}before permutation

j	3	5	6	2	1	4	7
X_j^{new}	0.14	0.29	0.31	0.58	0.74	0.82	0.92

Table 3.11 Solution vector X^{new}after permutation

3.4. Machine Assignment

In the mould shop, operators are more likely to process the first available jobs. Re-scheduling the job sequence every stage would cause them confused. After considering the ease of implementation, First-In-First-Out (FIFO) dispatching rule is adopted for machine assignment.

A schedule can be obtained after simulation of FIFO dispatching with the entry point sequence. Below is the pseudo-code of FIFO dispatching:

Input parameters

n	Number of jobs
l	Number of stages
m_i	Number of machines at Stage i
$V_{i,k}$	Speed of Machine k at Stage i
$U_{i,j}$	Standard processing time of Job j at Stage i
S_i	Setup time of a job at Stage i
L	Entry point sequence of job set
$A_{0,k}$	Available time of Machine k at Stage 0, i.e. current machine available time
$F_{0,j}$	Finish time of Job j at Stage 0, i.e. job release time

Variables

$K_{i,j}$	Selected machine of Job j at Stage i
$P_{i,j}$	Processing time of Job j at Stage i
$R_{i,j}$	Release time of Job j at Stage i
$F_{i,j}$	Finish time of Job j at Stage i
$A_{i,k}$	Available time of Machine k at Stage i
MS	Makespan of the schedule

for $i = 1\ to\ l$ **do**//consider job sequence stage by stage

for $h = 1\ to\ n$ **do**

$j = L_h$

if $U_{i,j} = 0$ **then** //skip Stage i

$R_{i,j} = F_{i-1,j}$

$P_{i,j} = 0$

$F_{i,j} = R_{i,j}$

$K_{i,j} = 0$ //no machine is selected

continue

end if

$k = arg\ \underset{1 \le k \le mi}{min}(A_{i,*})$ //select the first available machine

$K_{i,j} = k$

$R_{i,j} = \max\ (A_{i,k}, F_{i-1,j})$

$P_{i,j} = U_i / V_{i,k}$

$F_{i,j} = R_{i,j} + S_i + P_{i,j}$

$A_{i,k} = F_{i,j}$

end for h

$L_* = sort(F_{i,*})$ //sort in ascending order

end for i

$MS = \max\ (F_{l,*})$

Chapter 4 Implementation

The proposed algorithm is coded in MATLAB (R2010b) and executed on a laptop with 1.83GHz Intel Core 2 Duo processor, 3GB RAM and Windows Vista operating system.

MATLAB is chosen over the other programming languages because it has matrix manipulation ability. Several mathematical operations that work on arrays or matrices are built-in to the MATLAB environment. The graphical output is optimized for interaction. Plotting is easy using the graphical interactive tools.

MATLAB's functionality can be greatly expanded by the addition of toolboxes. These are sets of specific functions that provided more specialized functionality. Excel link allows data to be written in a format recognized by Excel. There are numeric resources about coding in MATLAB on internet.

The add-on product, MATLAB Coder, can generate standalone C and C++ code from MATLAB code. The functions produced in the tests are reusable. All the MATLAB functions created are in Appendix A.

Problem instances based on current situation are generated for testing the effectiveness of the proposed scheduling algorithms. The static data, such as

machine number, processing speed and preparation time, would be the same in every problem instance. Three different machine sets are combined with five different sets of job to generate 15 problem instances (Appendix B).

In each machine set, machine available time for each machine is provided. A large value, 999999, in machine available time represents the unavailability of the machine. Standard processing times and release times of each job can be found in each job set. The number of jobs in each job set is different from one another.

For heuristic algorithms, same solution would be generated for the same problem. There are six different heuristic algorithms to be tested: ERT, LPT, SPT, ERT-NEH, LPT-NEH and SPT-NEH. The first three algorithms are simple dispatching rules. The latter three algorithms are hybrid NEHs. Two meta-heuristic algorithms are tested: RKHS and RKGA.

The relative percentage deviation (RPD) [Naderi, Ruiz, & M.Zandieh, 2010] is used as an indicator to compare the performance of the algorithms (Eq. 4.1). Based on the objective of finding the minimum makespan, the algorithm which returns the minimum makespan is the best. Therefore, the lower value the RPD, the better is the algorithm. Since meta-heuristic algorithms return different

solution every computation. The minimum value obtained from the six heuristic algorithms will be set as Min_{sol}.

$$RPD = \frac{Alg_{sol} - Min_{sol}}{Min_{sol}} \times 100\% \quad \text{(Eq. 4.1)}$$

where

Alg_{sol}= the makespan obtained from the particular algorithm

Min_{sol}= the minimum makespan obtained among the heuristic algorithms

Parameter tuning is required for the two meta-heuristic algorithms. Different combinations of the parameters are tested in order to find out the best choice of parameter values (Table 4.1 & 4.2).In total, there are 54 combinations of parameter setting for each algorithm. For each setting, three runs would be carried out. Therefore, 2430 runs are carried out for each meta-heuristic algorithm.

Parameters	**Values**
HMS	[n, 2n]
HMCR	[0.1, 0.5, 0.9]
PAR	[0.1, 0.5, 0.9]
BW	[0.1, 0.5, 0.9]

Table 4.1 Tuning of RKHS Parameters

Parameters	**Values**
POP	[30, 50]
CX	[0.1, 0.5, 0.9]
TOP	[0.1, 0.2, 0.3]
BOT	[0.1, 0.2, 0.3]

Table 4.2 Tuning of RKGA Parameters

Chapter 5 Results

The computational results are shown in Appendix C. The analysis of the results is presented in this chapter.

5.1. Heuristic algorithms

Among the six heuristic algorithms (Figure 5.1 & Table 5.1), hybrid NEH outperforms simple dispatching rules. ERT-NEH performs the best with the lowest average RPD. Simple dispatching rules perform as well as hybrid NEH algorithms in problem instances with job size of 4. Their performances generally fall behind hybrid NEH with larger job sizes.

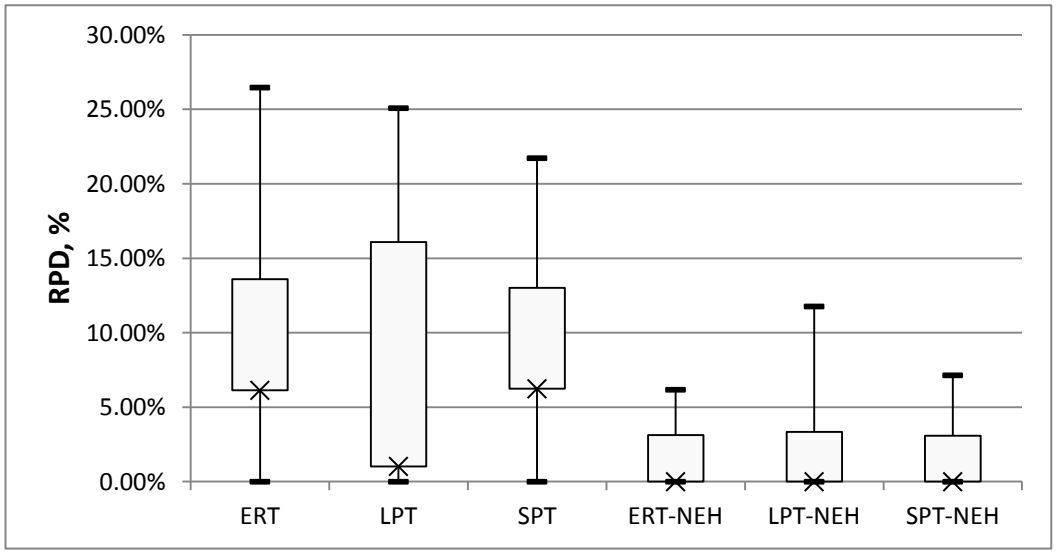

Figure 5.1 Box plot of RPDs obtained from the six heuristic algorithms (Conventional dispatching rules: ERT, LPT, SPT; Hybrid dispatching rules: ERT-NEH, LPT-NEH, SPT-NEH)

Instance	ERT	LPT	SPT	ERT-NEH	LPT-NEH	SPT-NEH
1	12.18%	0.00%	12.18%	6.17%	0.00%	6.17%
2	0.00%	2.06%	0.00%	0.00%	0.00%	2.06%
3	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
4	6.81%	0.00%	10.06%	0.00%	0.00%	0.00%
5	4.61%	0.00%	6.74%	3.12%	1.81%	2.30%
6	11.00%	8.17%	10.83%	0.00%	4.50%	3.17%
7	18.49%	7.25%	8.28%	0.00%	2.96%	2.96%
8	14.13%	9.68%	5.07%	1.38%	5.07%	0.00%
9	26.46%	25.08%	5.85%	1.54%	0.00%	5.85%
10	5.46%	12.46%	13.87%	1.54%	0.00%	7.14%
11	7.39%	15.84%	21.72%	0.00%	11.76%	2.71%
12	8.11%	7.60%	16.73%	2.66%	0.25%	0.00%
13	13.07%	19.03%	9.94%	3.12%	0.00%	2.98%
14	17.31%	17.31%	16.57%	3.73%	3.73%	0.00%
15	12.53%	16.34%	6.63%	6.14%	0.00%	2.46%
Average	10.50%	9.39%	9.63%	1.96%	2.01%	2.52%

Table 5.1 RPDs obtained from the six heuristic algorithms

The minimum makespan of each problem instances obtained from the six heuristic algorithms is set to be Min_{sol} (Table 5.2). These values would be used as the benchmarks to evaluate meta-heuristic algorithms' performances. The benchmark schedules are in Appendix D.

Instance	$\boldsymbol{Sol_{min}}$
1	48.58
2	48.58
3	51.50
4	53.83
5	50.67
6	50.00
7	56.33
8	54.25
9	54.17
10	59.50
11	55.25
12	65.75
13	58.67
14	55.83
15	67.83

Table 5.2 BenchmarkMin_{sol} of each problem instance

5.2. RKHS algorithm

Table 5.3, 5.4, 5.5 and 5.6 presents the RPDs of the average of the best makespan generated by RKHS for each problem instance. Among the different settings, parameters with the following values perform better: HMS=2n, HMCR=0.9, PAR=0.1, BW=0.1. From the computation results, RKHS's performance is as good as the best hybrid NEH algorithm.

Average Best Makespan	***HMS***	
Problem Instance	***n***	**2*n***
1	0.23%	0.00%
2	0.00%	0.00%
3	0.00%	0.00%
4	0.06%	0.01%
5	-1.18%	-1.24%
6	-0.80%	-1.27%
7	0.64%	0.63%
8	-0.34%	-0.46%
9	2.20%	2.28%
10	-1.05%	-1.58%
11	3.37%	2.68%
12	1.37%	1.38%
13	0.88%	0.99%
14	3.66%	3.37%
15	-3.89%	-4.15%
Average	0.34%	0.18%

Table 5.3 Average best makespan obtained with HMS = [n, 2n]

Average Best Makespan	***HMS***		
Problem Instance	**0.1**	**0.5**	**0.9**
1	0.00%	0.23%	0.11%
2	0.00%	0.00%	0.00%
3	0.00%	0.00%	0.00%
4	0.03%	0.03%	0.03%
5	-1.27%	-1.18%	-1.17%
6	-0.68%	-1.35%	-1.08%
7	1.00%	0.30%	0.60%
8	-0.03%	-0.51%	-0.64%
9	2.87%	2.40%	1.46%
10	-0.55%	-1.22%	-2.18%
11	3.84%	2.75%	2.49%
12	1.70%	1.44%	0.99%
13	1.43%	0.94%	0.43%
14	3.51%	3.38%	3.65%
15	-3.99%	-3.91%	-4.15%
Average	0.52%	0.22%	0.04%

Table 5.4 Average best makespan obtained with HMCR = [0.1,0.5,0.9]

Average Best	***PAR***		
Instance	**0.1**	**0.5**	**0.9**
1	0.11%	0.11%	0.11%
2	0.00%	0.00%	0.00%
3	0.00%	0.00%	0.00%
4	0.02%	0.03%	0.04%
5	-1.27%	-1.19%	-1.16%
6	-1.48%	-0.79%	-0.84%
7	0.36%	0.66%	0.88%
8	-0.86%	-0.20%	-0.13%
9	2.17%	2.08%	2.48%
10	-1.88%	-1.65%	-0.41%
11	2.29%	3.55%	3.24%
12	1.37%	1.75%	1.02%
13	0.90%	0.84%	1.07%
14	3.63%	3.15%	3.77%
15	-4.29%	-3.88%	-3.88%
Average	0.07%	0.30%	0.41%

Table 5.5 Average best makespan obtained with PAR = [0.1,0.5,0.9]

Average Best	***BW***		
Problem Instance	**0.1**	**0.5**	**0.9**
1	0.11%	0.11%	0.11%
2	0.00%	0.00%	0.00%
3	0.00%	0.00%	0.00%
4	0.02%	0.04%	0.03%
5	-1.24%	-1.15%	-1.23%
6	-0.64%	-1.05%	-1.41%
7	0.56%	0.77%	0.57%
8	-0.27%	-0.53%	-0.38%
9	1.77%	2.66%	2.30%
10	-1.82%	-1.06%	-1.07%
11	3.23%	3.00%	2.85%
12	1.47%	1.05%	1.61%
13	0.83%	1.14%	0.83%
14	3.19%	3.60%	3.76%
15	-4.42%	-3.77%	-3.86%
Average	0.19%	0.32%	0.27%

Table 5.6 Average best makespan obtained with BW = [0.1,0.5,0.9]

5.3. RKGA algorithm

Table 5.7, 5.8, 5.9 and 5.10 presents the RPDs of the average of the best makespan generated by RKGA for each problem instance. Among the different settings, parameters with the following values perform better: POP=60, CX=0.5, TOP=0.2, BOT=0.2. From the computation results, RKGA's performance is much better than the other proposed algorithms. It can obtain better result is each problem instance.

Average Best	***POP***	
Problem Instance	***30***	**60**
1	0.00%	0.00%
2	0.00%	0.00%
3	0.00%	0.00%
4	0.00%	0.00%
5	-1.64%	-1.64%
6	-4.02%	-4.63%
7	-2.73%	-3.30%
8	-4.30%	-5.07%
9	-3.94%	-4.63%
10	-7.97%	-8.30%
11	-4.39%	-5.58%
12	-4.05%	-4.65%
13	-7.33%	-8.98%
14	-4.12%	-5.73%
15	-8.78%	-9.43%
Average	-3.55%	-4.13%

Table 5.7 Average best makespan obtained with POP = [30, 60]

Average Best	***CX***		
Problem Instance	**0.1**	**0.5**	**0.9**
1	0.00%	0.00%	0.00%
2	0.00%	0.00%	0.00%
3	0.00%	0.00%	0.00%
4	0.00%	0.00%	0.00%
5	-1.64%	-1.64%	-1.64%
6	-4.53%	-4.30%	-4.15%
7	-3.02%	-3.07%	-2.96%
8	-4.66%	-4.82%	-4.57%
9	-4.29%	-4.56%	-4.00%
10	-8.05%	-8.24%	-8.12%
11	-4.85%	-5.09%	-5.01%
12	-4.28%	-4.45%	-4.32%
13	-7.50%	-8.88%	-8.08%
14	-4.34%	-5.66%	-4.77%
15	-8.77%	-9.15%	-9.40%
Average	-3.73%	-3.99%	-3.80%

Table 5.8 Average best makespan obtained with CX = [0.1, 0.5, 0.9]

Average Best	***TOP***		
Problem Instance	**0.1**	**0.2**	**0.3**
1	0.00%	0.00%	0.00%
2	0.00%	0.00%	0.00%
3	0.00%	0.00%	0.00%
4	0.00%	0.00%	0.00%
5	-1.64%	-1.64%	-1.64%
6	-4.22%	-4.33%	-4.43%
7	-2.77%	-3.17%	-3.10%
8	-4.48%	-4.88%	-4.69%
9	-4.28%	-4.34%	-4.24%
10	-8.07%	-8.13%	-8.22%
11	-4.83%	-5.21%	-4.91%
12	-4.19%	-4.57%	-4.29%
13	-7.97%	-8.31%	-8.18%
14	-4.65%	-5.06%	-5.05%
15	-8.99%	-9.32%	-9.01%
Average	-3.74%	-3.93%	-3.85%

Table 5.9 Average best makespan obtained with TOP = [0.1, 0.2, 0.3]

Average Best	***BOT***		
Instance	**0.1**	**0.2**	**0.3**
1	0.00%	0.00%	0.00%
2	0.00%	0.00%	0.00%
3	0.00%	0.00%	0.00%
4	0.00%	0.00%	0.00%
5	-1.64%	-1.64%	-1.64%
6	-4.15%	-4.57%	-4.25%
7	-3.02%	-3.08%	-2.94%
8	-4.71%	-4.70%	-4.65%
9	-4.29%	-4.28%	-4.28%
10	-8.07%	-8.22%	-8.13%
11	-5.09%	-4.97%	-4.90%
12	-4.40%	-4.54%	-4.11%
13	-8.51%	-8.39%	-7.56%
14	-5.10%	-5.14%	-4.53%
15	-9.27%	-9.12%	-8.92%
Average	-3.88%	-3.91%	-3.73%

Table 5.10 Average best makespan obtained with BOT = [0.1, 0.2, 0.3]

5.4. Computation times

The average computation times of each algorithm in each problem instances are presented in Table 5.11. Except RKGA, all of the algorithms can finish computation within 1 second. Figure 5.2 shows the computation times of hybrid NEHs increase exponentially with the job number. Figure 5.3 illustrates that computation times of RKGA increases much more than RKHS with increasing job size. The computation times for RKGA are significantly longer than the other algorithms.

j	**ERT**	**LPT**	**SPT**	**ERT-NEH**	**LPT-NEH**	**SPT-NEH**	**RKHS**	**RKGA**
4	0.0014	0.0014	0.0014	0.0104	0.0104	0.0104	0.03	0.86
8	0.0021	0.0021	0.0021	0.0644	0.0543	0.0544	0.08	1.80
12	0.0030	0.0030	0.0030	0.1666	0.1669	0.1653	0.14	4.29
16	0.0038	0.0037	0.0037	0.3858	0.3649	0.3638	0.20	5.75
20	0.0050	0.0049	0.0048	0.7059	0.7111	0.7100	0.28	9.02

Table 5.11 Average computation times (second) of each algorithm in problems with different job sizes

5.5. Performance evaluation

The proposed constraint-handling-free algorithm is simple to implement in

real life. The random key representation can avoid the existence of duplicated position value in sequencing which is a common problem when employing meta-heuristic algorithms to solve scheduling problems. The computational results demonstrate that the suggested approach can generate schedules with short makespan and balanced resource allocation for problems with job size below 20. The computation time of all the proposed algorithms is acceptable.

Table 5.12 shows that RKGA performs the best in different problem instances. RKHS performs the second best. Considering the computation time of RKHS is affected much less than RKGA with increasing problem size, it is suggested that RKHS should be used when the problem size is large.

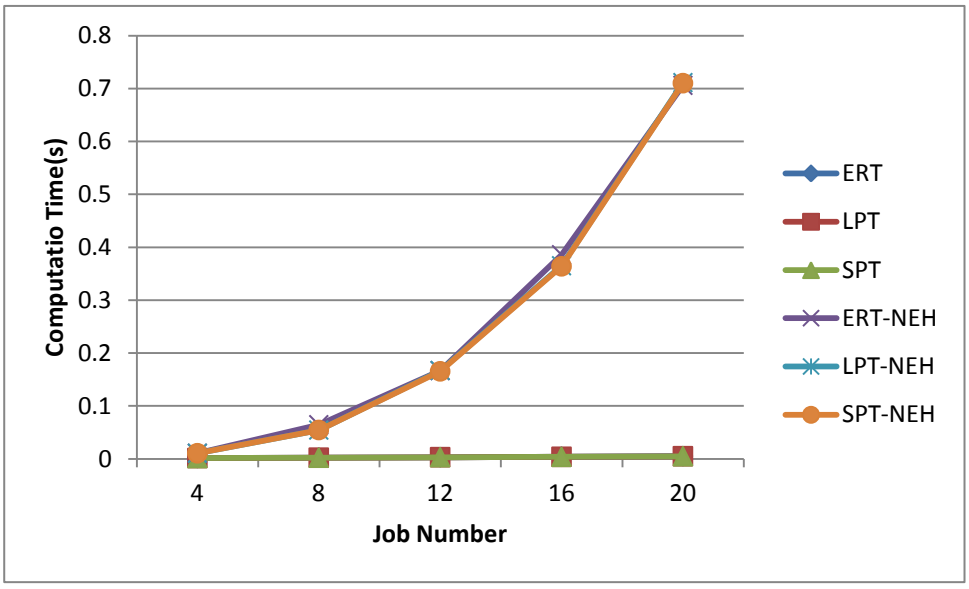

Figure 5.2 Comparison of computation times of the six heuristic algorithms

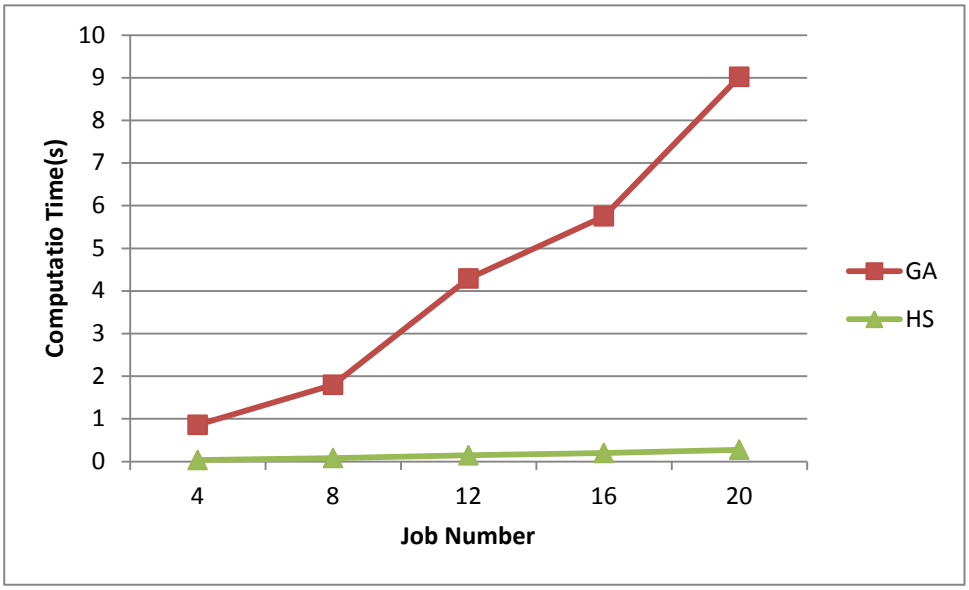

Figure 5.3 Comparison of computation times of the two meta-heuristic algorithms

Instance	Job Size	NEH	RKGA	RKHS
1	**4**	48.58	48.58	48.58
2	**4**	48.58	48.58	48.58
3	**4**	51.5	51.5	51.5
4	**8**	53.83	53.83	53.83
5	**8**	50.67	49.83	49.83
6	**8**	50	47.33	47.33
7	**12**	56.33	54.08	54.5
8	**12**	54.25	50.92	51.5
9	**12**	54.17	50.75	52.5
10	**16**	59.5	54.5	55.08
11	**16**	55.25	51.67	53.75
12	**16**	65.75	61.5	63.5
13	**20**	58.67	52.33	55.42
14	**20**	55.83	51.42	55
15	**20**	67.83	60.33	61.83

Table 5.12 The best result (minimum makespan) of each algorithm in problems with different job sizes

Chapter 6 Discussions

In today's competitive business environment, demand for shorter mould production cycle is increasing. Improvement in machine utilization is essential to obtain profitable operation and sustainable business. Due to the one-of-a-kind nature of the injection mould, the processing routes and times of the mould components are varied from mould to mould. It is difficult for schedulers to give a detailed production schedule with such a wide variation. As a result, study of an appropriate methodology to tackle this problem is raised by Asahi.

6.1. Difficulties of the Research

From the perspective of a commercial company, investment on any research and infrastructure is under serious scrutiny. In Asahi, the absence of planning system causes operation difficulties. Collection of job data is difficult since proper data handling practice is absent in the mould shop. Management is not aware of the power of data. Decisions are usually made by gut feeling.

Data analysis and management is not conducted in the mould shop. The production is planned manually without any realization of current production capacity. There is not any estimation of processing time. Manual tracking is the

only way to collect necessary data.

Data used in testing is based on the samples collected in the mould shop. The test cases may not fully reflect the entire mould shop production as multiple moulds are manufactured in the mould shop concurrently. Only jobs of key components are studied when the other components are neglected.

Similar research project [Leung, 2009] has been done recently. An intelligent business process management decision support system is developed for a mould manufacturer. In that research, the studied company has installed data tracking devices in the mould shop. All mould components are placed in the RFID-tagged trays. Every entrance of the working zone is installed with RFID readers.

Unlike that company, there are no such tracking devices in Asahi mould shop. Jobs are distributed, tracked and transported by experienced mould engineers who know the next process of the work-in-process components. Information is transmitted by word of mouth. Real-time data cannot be collected for evaluation. Considering the limitations of the experiment environment, simulation of the current mould shop is adopted instead of study of real-time case to evaluate the proposed methodologies.

6.2. Limitations of the Proposed Methodology

Due to the above constraints, this research is focused on scheduling algorithm development. The algorithm should be able to generate feasible schedule in a short time.

Currently, the production schedule cannot align with the shop capacity. The scheduler generates a schedule roughly without considering the workload in shop floor. The algorithms are able to simulate the workload of the shop floor so that the scheduler can rearrange the jobs accordingly. However, the utilization of each machine is not considered in this algorithm. The costs and manpower associated with the proposed schedule have not been studied. Occurrence of unexpected incidents which can alter the schedule is not considered in this research.

6.3. Deficiencies of Implementation

The problem instances are generated based on historical data. Performance in real time has not been studied.

Chapter 7 Conclusion and Future Work

Asahi aims to improve mould shop operation efficiency through enhancement of production scheduling. Application of scheduling algorithms is suggested to assist decision making of schedulers after study of mould shop workflow. Research on suitable scheduling algorithms for Asahi is carried out.

7.1. Summary of the Research

Workflow of mould making in Asahi is studied. Classification of Machine environment and job characteristics of Asahi's mould shop is performed. The machine environment of the mould shop is classified as flexible flow shop, where multiple machines with varied processing speeds are set up in multiple production stages. Non-anticipatory setup times exist in between consecutive production stages. Skipping of one or more production stages is allowed as long as the job follows the production sequence.

In mould manufacturing, the critical components of the moulds require the most processing times and machining operations. Proper allocation of resources on these components can highly improve the efficiency of the mould shop. Therefore, the research focuses on scheduling the mould critical components with the objective of minimum makespan.

Simple dispatching rules (ERT, LPT, SPT), heuristic algorithms (ERT-NEH, LPT-NEH and SPT-NEH) and meta-heuristic algorithms (RKHS and RKGA) are studied to solve the scheduling problems. In order to test the effectiveness of the algorithms, problem instances with different job sizes and machine sets are prepared. For the two meta-heuristic algorithms, study on parameter selection is also carried out.

The computational results show that all the proposed algorithms can generate feasible schedules in a short time. Among these algorithms, RKGA performs the best. New schedule can be generated within 10 seconds. It is suggested that this algorithm should be chosen for integrating into future production system of Asahi (H.K.) Limited. It is found that the schedule generated by the algorithm can provide better estimation of future workload in the shop floor.

7.2. Contributions of the Research

This research presents a methodology for scheduling mould shop using Artificial Intelligence technologies, including hybrid heuristic and meta-heuristic algorithms. The proposed approach overcomes the inadequacy of scheduling large and complex scheduling problems using Asahi's current practice. Compared

with current manual scheduling approach, this approach allows schedulers to generate more detailed schedules with much less time. It can highly improve the work efficiency of Asahi's administrative staff by saving time in schedule generation.

The existing manual approach cannot provide any schedule of each machine's or each work group's workload while the proposed algorithm can give the start time and finish time of each job on each machine. More detailed information can be generated for decision making. Workload of mould shop can be foreseen when detailed machine allocation plan can be provided. Machine utilization in the mould shop can be increased after employment of the suggested algorithms.

The proposed algorithms can be applied in different scheduling problems even if the machine environments and job characteristics vary from time to time. Schedulers can amend the schedule quickly when any unexpected incidents, such as machine breakdown, happen. The mould shop can be more responsive to the changes as a whole. .

The proposed methodology adopts random key representation in solution encoding on GA and HS. It is found that little research has applied these

algorithms on mould shop scheduling problem. This study shows that the effectiveness of solving mould shop scheduling problem with this approach.

7.3. Future Work

In this study, only scheduling of key mould components is done. More problem instances with different characteristics should be studied in the future to test the robustness of the proposed algorithms. Scheduling of entire mould project should be carried out when such kind of data is collected.

Since the jobs in Asahi mould shop are unpredictable, rescheduling is needed when job arrives or cancels. The proposed scheduling algorithms are designed in consideration of rescheduling in future development. The initial available machine times can be captured if production tracking system is available. Rescheduling of the jobs in queue can be done any time when needed. Since the main concern of the production planner is to clear the job queue as soon as possible, a new schedule can be generated by forward scheduling. Rescheduling can be studied when tracking devices is installed in the future.

Beside time, cost is important in the business world. Cost-related functions can be considered as the scheduling objectives in further study. Balancing time and cost of a schedule would be an interesting multi-objective scheduling

problem.

As MATLAB is for algorithm development, it is not suitable for system development. However, the functions developed by MATLAB (Appendix A) can be converted to C/C++ and integrated in other system in the future. It is reusable for the company when production system is developed later. A user-friendly graphic user interface should be developed to enable users to employ the algorithm.

References

Baker, K. R. (1974). *Introduction to Sequencing and Scheduling* (1st ed.). Canada: John Wiley & Sons.

Bean, J. C. (1994). Genetic Algorithms and Random Keys for Sequencing and Optimization. *ORSA Joural on Computing, 6* (2), 154-160.

Brucker, P. (2007). *Scheduling Algorithms* (5th ed.). Berlin: Springer-Verlag.

Choi, B. K., Ko, K., & Kim, B. H. (2005). Development of Intelligent Mold Shop. *Computer-Aided Design & Applications, 2* (5), pp. 619-626.

Choy, K. L., Leung, Y. K., Chow, H. K., Poon, T. C., Kwong, C. K., Ho, G. T., et al. (2011). A hybrid scheduling decision support model for minimizing job tardiness in a make-to-order based mould manufacturing environment. *Expert Systems with Applications, 38*, pp. 1931-1941.

Fallböhmer, P., Altan, T., Tönshoff, H. K., & Nakagawa, T. (1996). Survey of the die and mold manufacturing industry-Practices in Germany, Japan, and the United States. *Journal of Materials Processing Technology, 59* (1), pp. 158-168.

Fawcett, S., & Pearson, J. (1991). Understanding and applying constraint management in today's manufacturing environment. *Production and Inventory Management Journal* , 46-55.

Fredendall, L. D., & Lea, B. R. (1997). Improving the product mix heuristic in the theory of constraints. *International Journal of Production Research, 35* (6), 1535-1544.

Gan, P. Y., Lee, K. S., & Zhang, Y. F. (2001). A branch and bound algorithm based process-planning system for plastic injection mould bases. *International Journal of Advanced Manufacturing Technology, 18*, pp. 624-632.

Garey, M. R., Johnson, D. S., & Sethi, R. (1976). The complexity of flowshops and jobshop scheduling. *Mathematics of Operations Research, 1* (2), 117-129.

Geem, Z. W., Kim, J. H., & Loganathan, G. V. (2001). A new heuristic optimization algorithm: harmony search. *Simulation, 76* (2), 60-68.

Geem, Z. W., Lee, K. S., & Park, Y. (2005). Application of Harmony Search to Vehicle Routing. *American Journal of Applied Sciences, 2* (12), 1552-1557.

Goldratt, E.M. (1993). What is the theory of constraints? *APICS-The Perform Advant, 3*, 46-55.

Graham, K. (2000). Critical chain: the theory of constraints applied to project management. *International Journal of Project Management* , 173-177.

Graham, R., Lawler, E., Lenstra, J., & Rinnooy Kan, A. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics, 5*, 287-326.

Gupta, J. (1988). Two-stage, hybrid flowshop scheduling problem. *Journal of the Operational Research Society, 4*, 359-364.

Hayes, R. H., & Wheelwright., S. C. (1979). Link manufacturing process and product life cycles. *Harvard Business Review (January-February)* , 133-140.

Henriques, E., Peças, P., & Cunha, P. F. (2007). Perspectives of mould making industry for digital global manufacturing. *Digital Enterprise Technology* , pp. 449-456.

Herrmann, J. W. (2006). Rescheduling strategies, policies, and methods. In J. W. Herrmann, *Handbook of production scheduling* (pp. 135-148). New York: Springer Science+Business Media, Inc.

Hicks, C., Song, D. P., & Earl, C. F. (2007). Dynamic scheduling for complex engineer-to-order products. *International Journal of Production Research,, 45* (15), 3477-3503.

Iima, H., Kudo, R., Sannomiya, N., & Kobayashi, Y. (1999). An autonomous decentralized scheduling algorithm for a scheduling problem in a metal mould assembly process. *Journal of Intelligent Manufacturing, 10*, pp. 161-167.

Kochhar, S., Morris, R. J., & Wong, W. S. (1988). The local search approach to flexible flow line scheduling. *Engineering Costs and Production Economics, 14*, 25-37.

Lee, K. S., & Geem, Z. W. (2004). A new structural optimization method based on the harmony search algorithm. *Computers and Structures, 82*, 781-798.

Lee, R.-S., Chen, Y.-M., Cheng, H. Y., & Kuo, M.-D. (1998). A framework of a concurrent process planning system for mold manufacturing. *Computer Integrated Manufacturing Systems, 11* (3), pp. 171-190.

Lei, D. (2009). Genetic Algorithm for Job Shop Scheduling under Uncertainty. In U. Chakraborty, *Comput. Intel. in Flow Shop and Job Shop Sched.* (pp. 191-228). Springer-Verlag Berlin Heidelberg.

Leung, Y. K. (2009). *Development of an Intelligent Business Process Management Decision Support System for Mould Manufacturer.* The Hong Kong Polytechnic University.

Liu, Z., Liao, L., Yang, W., Wang, J., & Zhao, Y. (2010). Genetic Algorithm on Solving Mould Enterprise's Job-Shop Scheduling Problem. *International Conference on Computing, Control and Industrial Engineering* , pp. 38-41.

Loendorf, W. R. (2008). *Transition of the Tooling Industry in a Competitive Global Environment.* Ann Arbor, United States: ProQuest.

Mendes, J., Gonçalves, J., & Resende, M. (2009). A random key based genetic algorithm for the resource constrained project scheduling problem. *Computers & Operations Research* (36), 92-109.

Miyuan, S., hong, Q., & Juan, W. (2007, Sept 21-25). Multi-mode multi-project scheduling problem for mould production in MC enterprise. *International Conference on Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007.* , pp. 5316-5320.

Moursli, O., & Pochet, Y. (2000). A branch-and-bound algorithm for the hybrid flowshop. *Int. J. Production Economics, 64*, pp. 113-125.

Mun, S., & Geem, Z. W. (2009). Determination of viscoelastic and damage properties of hot mix asphalt concrete using a harmony search algorithm. *Mechanics of Materials, 41*, 339-353.

Naderi, B., Ruiz, R., & M.Zandieh. (2010). Algorithms for a realistic variant of flowshop scheduling. *Computers & Operations Research, 37*, 236-246.

Nawaz, M., Enscore, E. E., & Ham, I. (1983). A Heuristic Algorithm for the m-Machine, n-Job Flow-shop Sequencing Problem. *Omega, 11* (1), 91-95.

Nia, Q., Lu, W. F., Yarlagadda, P. K., & Ming, X. (2007). Business information modeling for process integration in the mold making industry. *Robotics and Computer-Integrated Manufacturing*, *23*, pp. 195-207.

Olhager, J. (2003). Strategic positioning of the order penetration point. *Int. J. Production Economics, 85*, 319-329.

Peças, P., & Henriques, E. (2003). The need for agile manufacturing implementation in mould making business. *Proceedings of the Business Excellence I - Performance Measures, Bechmarking and Best Practices in New Economy* , pp. 321-326.

Pinedo, M. L. (2008). *Scheduling: Theory, Algorithms, and Systems.* NewYork: Springer.

Pinedo, M., & Chao, X. (1999). *Operations scheduling with applications in manufacturing and services.* USA: McGraw-Hill Companies.

RianeF. (1998). Scheduling Hybrid Flowshops: Algorithms and Applications. Ph.D. Thesis, Facultés Universitaires Catholiques de Mons.

Santos, D., Hunsucker, J., & Deal, D. (1996). An evaluation of sequencing heuristics in flow shops with multiple processors. *Computers and IndustrialEngineering, 30* (4), 681-692.

Spragg, J. E. (1996). A discipline for reactive rescheduling., (pp. 199-204).

Sule, D. R. (1997). *Production planning and industrial scheduling:examples, case studies, and applications.* FL: CRC Press.

The local search approach to flexible flow line scheduling1988*Engineering Costs and Production Economics14*25-37

Vieira, G. E., Herrmann, J. W., & Lin, E. (2003). Rescheduling manufacturing systems: a framework of strategies, policies, and methods. *Journal of Scheduling, 6* (1), pp. 39-62.

Wang, L., Pan, Q.-K., & Tasgetiren, M. F. (2010). Minimizing the total flow time in a flow shop with blocking by using hybrid harmony search algorithms. *Expert Systems with Applications, 37*, 7926-7936.

Yeh, P.-Y., Cheng, T., Guo, X.-L., Xiao, Z., Zhong, Q., & Liu, G.-G. (2007). *Patent No. US 7,310,562 B2.* United States.

Zäpfe, G., Braune, R., & Bögl, M. (2010). *Metaheuristic Search Concepts.* Springer-Verlag Berlin Heidelberg.

Zou, D., Gao, L., Li, S., Wu, J., & Wang, X. (2010). A novel global harmony search algorithm for task assignment problem. *The Journal of Systems and Software, 83*, 1678-1688.

Appendix A MATLAB Functions

getMakespan.m

```
% Purpose: use FIFO rule to schedule the jobs
% jMax = total number of jobs
% hMax = total number of stages
% mNum(h) = number of machines at stage h
% mSpeed(k,h) = speed ratio of machine k at stage h
% processTime(j,h) = standard processing time of job j at
stage h
% setupTime(h) = setup time of a job at stage h
% jInitial(j) = initial release time of job j
% mInitial(k,h) = initial release time of machine k in
operation h
% x(j)  = job sequence

function [makespan,mSchedule] = getMakespan
(jMax,hMax,mNum,mSpeed,processTime,setupTime,jInitial,
mInitial,x)
[~,jSeq] = sort(x);
% rt(j.h) = release time of job j at stage h
rt = zeros(jMax,hMax);
% ft(j,h) = finish time of job j at stage h
ft = zeros(jMax,hMax);
% pt(j,h) = processing time of job j at stage h
pt = zeros(jMax,hMax);
% M(j,h) = selected machine of job j at stage h
M = zeros(jMax,hMax);
% mSchedule(k,h) = schedule of machine k at stage h
% in each cell, there are 3 columns:
% column 1 | column 2    | column 3
% jobIndex | release time | finish time
mSchedule = cell(max(mNum,[],2),hMax);
mAvaiTime = mInitial;
% mScheduleSize(k,h) = number of jobs under machine k for
operation h
mScheduleSize = zeros(size(mSchedule));
for h = 1:1:hMax
for j = 1:1:jMax
jCur = jSeq(j);
ifprocessTime(jCur,h) == 0
```

```
if h == 1
rt(jCur,h) = jInitial(jCur);
else
rt(jCur,h) = ft(jCur,h-1);
end
pt(jCur,h) = 0;
ft(jCur,h) = rt(jCur,h);
M(jCur,h) = 0;
continue
end
       [~,mCur] = min(mAvaiTime(1:mNum(h),h),[],1);% get
first available machine
if h == 1
rt(jCur,h) = max(mAvaiTime(mCur,h),jInitial(jCur));
else
rt(jCur,h) = max(mAvaiTime(mCur,h),ft(jCur,h-1));
end
pt(jCur,h) = processTime(jCur,h)/mSpeed(mCur,h);
ft(jCur,h) = rt(jCur,h)+setupTime(h)+pt(jCur,h);
M(jCur,h) = mCur;
mAvaiTime(mCur,h) = ft(jCur,h);
mScheduleSize(mCur,h) = mScheduleSize(mCur,h)+1;
mSchedule{mCur,h}(mScheduleSize(mCur,h),:) =
[jCur,rt(jCur,h),ft(jCur,h)];
end
    [~,jSeqTemp] = sort(ft(:,h));
    p = 1;
for q = 1:1:size(jSeqTemp,1)
ifft(jSeqTemp(q),h) ~= 0
jSeq(p) = jSeqTemp(q);
          p = p+1;
end
end
end
makespan = max(max(ft(:,hMax),1));
```

SPT.m

```
% Purpose: sort the job using SPT rule
% jMax = total number of jobs
% hMax = total number of stages
% mNum(h) = number of machines at stage h
% mSpeed(k,h) = speed ratio of machine k at stage h
% processTime(j,h) = standard processing time of job j at
stage h
% setupTime(h) = setup time of a job at stage h
% jInitial(j) = initial release time of job j
% mInitial(k,h) = initial release time of machine k in
operation h
% time = computation time
function [makespan,time]=SPT
(jMax,hMax,mNum,mSpeed,processTime,setupTime,jInitial,
mInitial)
tic;
pSum = sum(processTime,2);
[spt(:,1),spt(:,2)] = sort(pSum,'ascend');
jobList = spt(:,2);
[makespan,mSchedule] = getMakespan
(jMax,hMax,mNum,mSpeed,processTime,setupTime,jInitial,
mInitial,jobList);
time=toc;
```

ERT.m

```
% Purpose: sort the job using ERT rule
% jMax = total number of jobs
% hMax = total number of stages
% mNum(h) = number of machines at stage h
% mSpeed(k,h) = speed ratio of machine k at stage h
% processTime(j,h) = standard processing time of job j at
stage h
% setupTime(h) = setup time of a job at stage h
% jInitial(j) = initial release time of job j
% mInitial(k,h) = initial release time of machine k in
operation h
% time = computation time

function [makespan,time]=ERT
(jMax,hMax,mNum,mSpeed,processTime,setupTime,jInitial,
mInitial)
tic;
[ert(:,1),ert(:,2)] = sort(jInitial,'ascend');
jobList = ert(:,2);
[makespan,mSchedule] = getMakespan
(jMax,hMax,mNum,mSpeed,processTime,setupTime,jInitial,
mInitial,jobList);
time=toc;
```

LPT.m

```
% Purpose: sort the job using LPT rule
% jMax = total number of jobs
% hMax = total number of stages
% mNum(h) = number of machines at stage h
% mSpeed(k,h) = speed ratio of machine k at stage h
% processTime(j,h) = standard processing time of job j at
stage h
% setupTime(h) = setup time of a job at stage h
% jInitial(j) = initial release time of job j
% mInitial(k,h) = initial release time of machine k in
operation h
% time = computation time

function [makespan,time]=LPT
(jMax,hMax,mNum,mSpeed,processTime,setupTime,jInitial,
mInitial)
tic;
pSum = sum(processTime,2);
[lpt(:,1),lpt(:,2)] = sort(pSum,'descend');
jobList = lpt(:,2);
[makespan,mSchedule] = getMakespan
(jMax,hMax,mNum,mSpeed,processTime,setupTime,jInitial,
mInitial,jobList);
time=toc;
```

SPT_NEH.m

```
% Purpose: sort the job using SPT-NEH rule
% jMax = total number of jobs
% hMax = total number of stages
% mNum(h) = number of machines at stage h
% mSpeed(k,h) = speed ratio of machine k at stage h
% processTime(j,h) = standard processing time of job j at
stage h
% setupTime(h) = setup time of a job at stage h
% jInitial(j) = initial release time of job j
% mInitial(k,h) = initial release time of machine k in
operation h
% time = computation time

function [bestMS,time]=
SPT_NEH(jMax,hMax,mNum,mSpeed,processTime,setupTime,jI
nitial,mInitial)
tic;
pSum = sum(processTime,2);
[spt(:,1),spt(:,2)] = sort(pSum,'ascend');
jobList0 = spt(1,2);
for a = 2:1:jMax % select job from lpt one by one
jobInsert = spt(a,2);
bestMS = 9999999;
for b = 1:1:size(jobList0,1)
for f = 1:1:a % identify the job insertion position,
f=insert position
          k = 1;
jobList = zeros(1,a);
for g = 1:1:a % insert job into the sequence, g=position
in sequence
if g == f
jobList(g) = jobInsert;
else
jobList(g) = jobList0(b,k);
                k = k+1;
end
end
          [makespan,mSchedule] = getMakespan
(a,hMax,mNum,mSpeed,processTime,setupTime,jInitial,mIn
itial,jobList);
```

```
if makespan <bestMS
bestMS = makespan;
bestSch = mSchedule;
bestSeq = jobList;
end
end
end
jobListO = bestSeq;
time=toc;
end
```

ERT_NEH.m

```
% Purpose: sort the job using ERT-NEH rule
% jMax = total number of jobs
% hMax = total number of stages
% mNum(h) = number of machines at stage h
% mSpeed(k,h) = speed ratio of machine k at stage h
% processTime(j,h) = standard processing time of job j at
stage h
% setupTime(h) = setup time of a job at stage h
% jInitial(j) = initial release time of job j
% mInitial(k,h) = initial release time of machine k in
operation h
% time = computation time

function [bestMS,time]=
ERT_NEH(jMax,hMax,mNum,mSpeed,processTime,setupTime,jI
nitial,mInitial)
tic;
[ert(:,1),ert(:,2)] = sort(jInitial,'ascend');
jobList0 = ert(1,2);
for a = 2:1:jMax % select job from lpt one by one
jobInsert = ert(a,2);
bestMS = 9999999;
for b = 1:1:size(jobList0,1)
for f = 1:1:a % identify the job insertion position,
f=insert position
          k = 1;
jobList = zeros(1,a);
for g = 1:1:a % insert job into the sequence, g=position
in sequence
if g == f
jobList(g) = jobInsert;
else
jobList(g) = jobList0(b,k);
                 k = k+1;
end
end
          [makespan,mSchedule] = getMakespan
(a,hMax,mNum,mSpeed,processTime,setupTime,jInitial,mIn
itial,jobList);
if makespan <bestMS
```

```
bestMS = makespan;
bestSch = mSchedule;
bestSeq = jobList;
end
end
end
jobListO = bestSeq;
time=toc;
end
```

LPT_NEH.m

```
% Purpose: sort the job using LPT-NEH rule
% jMax = total number of jobs
% hMax = total number of stages
% mNum(h) = number of machines at stage h
% mSpeed(k,h) = speed ratio of machine k at stage h
% processTime(j,h) = standard processing time of job j at
stage h
% setupTime(h) = setup time of a job at stage h
% jInitial(j) = initial release time of job j
% mInitial(k,h) = initial release time of machine k in
operation h
% time = computation time

function [bestMS,time]=LPT_NEH
(jMax,hMax,mNum,mSpeed,processTime,setupTime,jInitial,
mInitial)
tic;
pSum = sum(processTime,2);
[lpt(:,1),lpt(:,2)] = sort(pSum,'descend');
jobList0 = lpt(1,2);
for a = 2:1:jMax % select job from lpt one by one
jobInsert = lpt(a,2);
bestMS = 9999999;
for b = 1:1:size(jobList0,1)
for f = 1:1:a % identify the job insertion position,
f=insert position
          k = 1;
jobList = zeros(1,a);
for g = 1:1:a % insert job into the sequence, g=position
in sequence
if g == f
jobList(g) = jobInsert;
else
jobList(g) = jobList0(b,k);
                k = k+1;
end
end
          [makespan,mSchedule] = getMakespan
(a,hMax,mNum,mSpeed,processTime,setupTime,jInitial,mIn
itial,jobList);
```

```
if makespan <bestMS
bestMS = makespan;
bestSch = mSchedule;
bestSeq = jobList;
end
end
end
jobListO = bestSeq;
end
```

HS.m

```
% Purpose: Harmony search
% jMax = total number of jobs
% hMax = total number of stages
% mNum(h) = number of machines at stage h
% mSpeed(k,h) = speed ratio of machine k at stage h
% processTime(j,h) = standard processing time of job j at
stage h
% setupTime(h) = setup time of a job at stage h
% jInitial(j) = initial release time of job j
% mInitial(k,h) = initial release time of machine k in
operation h
% hmMax = harmony memory size
% hmcr = harmony considering rate
% par = pitch adjusting rate
% bw = bandwidth
% hmX = harmony vector
% genMax = the number of generation run
% bestCur = the current best result
% time = computation time
% steadyGen = the last generation

function [bestCur,time,steadyGen] = HS
(jMax,hMax,mNum,mSpeed,processTime,setupTime,jInitial,
mInitial,hmcr,par,bw,genMax,hmMax)
tic;
steadyGen = 0;
steadyGenTime = 0;
bestCount=0;
fx = size(hmMax);
bestFx = size(genMax);
worstFx = size(genMax);
for gen = 1:1:genMax
if gen == 1
hmX = zeros(jMax,hmMax);
for i = 1:1:hmMax
          x = rand(jMax,1);
          [makespan,mSchedule] = getMakespan
(jMax,hMax,mNum,mSpeed,processTime,setupTime,jInitial,
mInitial,x);
hmX(:,i) = x;
```

```
fx(i) = makespan;
schedule{i,:} = mSchedule;
end
else
    x = HS_Improvisation (jMax,hmMax,hmcr,par,bw,hmX);
    [makespan,mSchedule] = getMakespan
(jMax,hMax,mNum,mSpeed,processTime,setupTime,jInitial,
mInitial,x);
if makespan <fx(worstHM)
hmX(:,worstHM) = x;
fx(worstHM) = makespan;
schedule{worstHM,:} = mSchedule;
end
end
    [bestCur,bestHM] = min(fx);
    bestFx(gen) = bestCur;
bestSchedule = schedule{bestHM};
    [worstCur,worstHM] = max(fx);
worstFx(gen) = worstCur;
if gen ==1
bestOld = bestCur;
end
ifbestCur == bestOld
bestCount = bestCount+1;
steadyGen = gen;
else
bestCount = 0;
bestOld=bestCur;
end
ifbestCount== 20;
break;
end
end
time=toc;
```

HS_Improvisation.m

```
% Purpose: Harmony search improvisation
% jMax = total number of jobs
% hmMax = harmony memory size
% hmcr = harmony considering rate
% par = pitch adjusting rate
% bw = bandwidth
% hmX = harmony vector

function [x] = HS_Improvisation
(jMax,hmMax,hmcr,par,bw,hmX)
x = zeros(jMax,1);
for j = 1:1:jMax
    xrand1 = rand;
if xrand1 <hmcr
x(j) = hmX(j,randi(hmMax));
       xrand2 = rand;
if xrand2 < par
x(j) = mod(x(j)+ (rand-0.5)*bw,1);
end
else
x(j) = rand;
end
end
```

GA.m

```
% Purpose: Genetic algorithm
% jMax = total number of jobs
% hMax = total number of stages
% mNum(h) = number of machines at stage h
% mSpeed(k,h) = speed ratio of machine k at stage h
% processTime(j,h) = standard processing time of job j at
stage h
% setupTime(h) = setup time of a job at stage h
% jInitial(j) = initial release time of job j
% mInitial(k,h) = initial release time of machine k in
operation h
% hmMax = harmony memory size
% hmcr = harmony considering rate
% par = pitch adjusting rate
% bw = bandwidth
% hmX = harmony vector
% genMax = the number of generation run
% bestCur = the current best result
% time = computation time
% steadyGen = the last generation

function [bestCur,time,steadyGen] =
GA(jMax,hMax,mNum,mSpeed,processTime,setupTime,jInitia
l,mInitial,cx,top,bot,genMax,popMax)
tic;
bestCount = 0;
p1Num = floor(top*popMax);
p2Num = popMax-p1Num;
botNum = floor(bot*popMax);
cxNum = popMax-p1Num-botNum;
allpopX=zeros(popMax,jMax+1,genMax);
for gen = 1:1:genMax
if gen == 1
popX = zeros(popMax,jMax+1);
popX(:,2:jMax+1) = rand(popMax,jMax);
for pop = 1:1:popMax
          [popX(pop,1),mSchedule] = getMakespan
(jMax,hMax,mNum,mSpeed,processTime,setupTime,jInitial,
mInitial,popX(pop,2:jMax+1));
end
```

```
popX = sortrows(popX);
else
popX =
GA_Reproduction(jMax,popMax,p1Num,p2Num,botNum,cxNum,c
x,popX);
for pop = p1Num+1:1:popMax
          [popX(pop,1),mSchedule] = getMakespan
(jMax,hMax,mNum,mSpeed,processTime,setupTime,jInitial,
mInitial,popX(pop,2:jMax+1));
end
popX = sortrows(popX);
end
allpopX(:,:,gen)=popX;
bestCur = popX(1,1);
if gen ==1
bestOld = bestCur;
end
ifbestCur == bestOld
bestCount = bestCount+1;
steadyGen = gen;
else
bestCount = 0;
bestOld=bestCur;
end
ifbestCount == 20;
break;
end
end
time=toc;
```

GA_Reproduction.m

```
% Purpose: GA reproduction
% popMax = size of population
% jMax = total number of jobs
% p1Num = top population
% p2Num = popMax-p1Num
% botNum = bottom population
% cxNum = popMax-p1Num-botNum
% cx = crossover rate between one parent from top population
and one parent
% from the remaining population
% popX = chromosome

function [popNew] =
GA_Reproduction(jMax,popMax,p1Num,p2Num,botNum,cxNum,c
x,popX)
popNew = size(popMax,jMax+1);
for c = p1Num+1:1:p1Num+cxNum
    p1 = randi(p1Num);
    p2 = randi(p2Num)+p1Num;
for j = 2:1:jMax+1
randcx = rand;
ifrandcx< cx
popNew(c,j) = popX(p1,j);
else
popNew(c,j) = popX(p2,j);
end
end
end
popNew(1:p1Num,:) = popX(1:p1Num,:);
popNew(p1Num+cxNum+1:popMax,2:jMax+1) =
rand(botNum,jMax);
```

Appendix B Problem Instances

Instance	Job Number	Machine sets	Job sets
1	4	A	a
2	4	B	a
3	4	C	a
4	8	A	b
5	8	B	b
6	8	C	b
7	12	A	c
8	12	B	c
9	12	C	c
10	16	A	d
11	16	B	d
12	16	C	d
13	20	A	e
14	20	B	e
15	20	C	e

Table B.1 Problem instances for testing

$A_{i,k}$	**Stage *i***						
Machine*k*	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	22	22	9	20	19	23	9
2	21	/	21	9	0	20	24
3	2	/	11	2	/	18	23
4	4	/	23	20	/	20	10
5	23	/	6	/	/	5	/

Table B.2 Available time of Machine *k* at Stage *i* of Machine SetA

$A_{i,k}$	**Stage *i***						
Machine*k*	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	12	18	14	2	2	2	12
2	3	/	14	15	14	1	22
3	17	/	1	8	/	19	15
4	999999	/	14	16	/	5	15
5	24	/	23	/	/	11	/

Table B.3 Available time of Machine *k* at Stage *i* of Machine Set B

$A_{i,k}$	**Stage *i***						
Machine*k*	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	19	5	18	22	1	999999	8
2	3	/	/	12	11	999999	12
3	16	/	6	17	/	13	3
4	21	/	15	999999	/	9	16
5	3	/	7	/	/	3	/

Table B.4 Available time of Machine *k* at Stage *i* of Machine Set C

Job *j*	$R_{1,j}$
1	21
2	24
3	0
4	13

Table B.5 Release time of Job *j* at Stage *I*of Job Set a

U_{ij}	Stage *i*						
Job *j*	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	2	2.25	1.5	8.75	2	2.25	1.5
2	1	0	1	2.25	1	0	1
3	4	0	6	3.75	4	0	6
4	2.75	4	0.75	2.5	2.75	4	0.75

Table B.6 Standard processing time of Job *j* at Stage *i* of Job Set a

Job *j*	$R_{1,j}$
1	16
2	14
3	9
4	5
5	19
6	3
7	12
8	0

Table B.7Release time of Job *j* at Stage *i* of Job Set b

U_{ij}	**Stage *i***						
Job *j*	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	3.25	2	0.5	1.5	3	0.75	4.5
2	3	0	1.5	10	0	11.5	7.5
3	0.25	0	1.25	6.75	0	1	7.5
4	4	0	4	1.25	0	1	4
5	0.75	1.75	5.75	9	2.75	9.25	0.25
6	3.25	0.25	5	9.5	1.5	12	5.25
7	0.5	0	4.75	9.75	0	6.25	2
8	3.75	0	5.25	3.5	1.5	10	0.5

Table B.8 Standard processing time of Job *j* at Stage *i* of Job Set b

Job *j*	$R_{1,j}$
1	15
2	11
3	0
4	14
5	15
6	16
7	8
8	12
9	17
10	22
11	1
12	2

Table B.9 Release time of Job *j* at Stage *i* of Job Set c

U_{ij}	**Stage *i***						
Job *j*	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	1.75	1.25	4.25	6.75	0	9.25	0.5
2	0.75	0	5.5	10.25	0	5.5	1
3	2.5	1.25	6	4.5	0	8.75	8
4	2.5	0	2.25	2.25	0	1	7.25
5	3.25	0	5	9.25	2.25	5	0.75
6	1.5	2	4	6.5	1.25	6.25	4.75
7	0.75	0.25	3.5	10.25	0	6	1.25
8	0.25	3.25	2	1.75	0	11	6.75
9	2.75	4	0.75	8	0	1.25	7.25
10	2	0.25	1.75	8	0	8.5	3.25
11	3.25	1.75	2	4.25	2.25	6	3.75
12	4	0	5.5	0.5	0	10.25	2.75

Table B.10 Standard processing time of Job *j* at Stage *i* of Job Set c

Job *j*	$R_{1,j}$
1	4
2	14
3	2
4	0
5	21
6	19
7	2
8	9
9	9
10	24
11	11
12	6
13	24
14	24
15	7
16	11

Table B.11 Release time of Job *j* at Stage *i* of Job Set d

U_{ij}	**Stage *i***						
Job *j*	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	1.25	1.5	5.75	9.75	0	12.75	4.25
2	1.5	0.5	1.75	4.75	2	5.75	5
3	3	1.75	0.5	4.75	0	3.5	3.5
4	0.75	0	1.25	4.5	0	11	3
5	0.75	0	5.25	4.75	0	11.75	1.25
6	1.75	1	2.75	0.75	2	4.75	2.5
7	2.25	0	6	4	2.75	4.25	5.75
8	2.75	0	0.5	7.5	0	12.5	5.5
9	3	0	5.5	11.75	2.75	2.5	6.75
10	2.25	0	1.25	8.5	0	9.25	7.5
11	1	0	2.75	6.25	1	7	7.25
12	0.75	0	4.5	2.5	0	8.25	0.75
13	3	0	3	2.5	2.5	6.5	0.5
14	3.5	0	3.5	3.75	0	7.75	0.5
15	3	0	4.25	6	0	5	0.25
16	2.25	4	3.75	1.5	0	3.25	2.75

Table B.12 Standard processing time of Job *j* at Stage *i* of Job Set d

Job *j*	$R_{1,j}$
1	3
2	5
3	16
4	4
5	16
6	19
7	1
8	14
9	7
10	21
11	17
12	7
13	13
14	15
15	10
16	20
17	23
18	13
19	4
20	2

Table B.13 Release time of Job *j* at Stage *i* of Job Set e

U_{ij}	**Stage *i***						
Job *j*	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	0.75	0.5	5.5	1.25	0	9.5	1
2	3.5	0	0.5	7	1	4.25	4
3	2	0.25	3	2	0	3.75	8
4	0.25	0	4	5	0	5	3
5	1	0	5.25	7.75	4	2.75	4.5
6	4	0	1.25	5.25	0	2.5	7
7	1.5	0.5	2.25	11	0	2.25	2.25
8	2	0.25	3.5	7.75	2	4.75	6.25
9	0.25	0	4.5	1.25	1.25	10.5	2
10	1.75	3.25	0.25	9.25	3.25	0.75	5.75
11	2.75	0.5	0.5	0.5	0	1	3.75
12	0.25	0	5.5	8.5	1	2	2
13	3.5	0	3.5	4.5	2.5	1.5	2.5
14	3	0	5.5	8.25	3.5	3	4.25
15	2.25	0	3.5	3.25	0	1.75	4.5
16	2.75	0	0.75	8.5	2	5.25	6
17	0.75	2.5	2.75	3	1.25	5	7.75
18	1.25	0.75	2	11.25	1.25	5.75	1.75
19	1	0	2.75	6.25	0	12.5	3.25
20	0.25	0.75	2.5	1	2.5	13	3.5

Table B.14 Standard processing time of Job *j* at Stage *i* of Job Set e

Appendix C Computation Results

Problem Instance	ERT	LPT	SPT	ERT-NEH	LPT-NEH	SPT-NEH
1	54.50	48.58	54.50	51.58	48.58	51.58
2	48.58	49.58	48.58	48.58	48.58	49.58
3	51.50	51.50	51.50	51.50	51.50	51.50
4	57.50	53.83	59.25	53.83	53.83	53.83
5	53.00	50.67	54.08	52.25	51.58	51.83
6	55.50	54.08	55.42	50.00	52.25	51.58
7	66.75	60.42	61.00	56.33	58.00	58.00
8	61.92	59.50	57.00	55.00	57.00	54.25
9	68.50	67.75	57.33	55.00	54.17	57.33
10	62.75	66.92	67.75	60.42	59.50	63.75
11	59.33	64.00	67.25	55.25	61.75	56.75
12	71.08	70.75	76.75	67.50	65.92	65.75
13	66.33	69.83	64.50	60.50	58.67	60.42
14	65.50	65.50	65.08	57.92	57.92	55.83
15	76.33	78.92	72.33	72.00	67.83	69.50

Table C.1 Minimum makespan generated from the six heuristic algorithms

Instance	ERT	LPT	SPT	ERT-NEH	LPT-NEH	SPT-NEH
1	0.0018	0.0019	0.0017	0.0140	0.0142	0.0140
2	0.0012	0.0012	0.0012	0.0086	0.0086	0.0087
3	0.0013	0.0012	0.0012	0.0086	0.0085	0.0086
4	0.0021	0.0021	0.0021	0.0539	0.0536	0.0547
5	0.0021	0.0021	0.0021	0.0542	0.0546	0.0544
6	0.0021	0.0021	0.0021	0.0852	0.0547	0.0541
7	0.0031	0.0030	0.0030	0.1656	0.1663	0.1656
8	0.0030	0.0031	0.0031	0.1696	0.1684	0.1660
9	0.0030	0.0030	0.0030	0.1645	0.1661	0.1644
10	0.0038	0.0037	0.0038	0.4294	0.3650	0.3635
11	0.0039	0.0037	0.0037	0.3650	0.3663	0.3645
12	0.0038	0.0037	0.0038	0.3630	0.3636	0.3636
13	0.0050	0.0049	0.0048	0.7073	0.7117	0.7203
14	0.0049	0.0049	0.0048	0.7044	0.7116	0.7052
15	0.0050	0.0048	0.0049	0.7060	0.7101	0.7045

Table C.2 Computation times (in seconds) of the six heuristic algorithms

Average Best	***HMS***		***HMCR***			***PAR***			***BW***		
Instance	***n***	**2*n***	**0.1**	**0.5**	**0.9**	**0.1**	**0.5**	**0.9**	**0.1**	**0.5**	**0.9**
1	48.69	48.58	48.58	48.69	48.64	48.64	48.64	48.64	48.64	48.64	48.64
2	48.58	48.58	48.58	48.58	48.58	48.58	48.58	48.58	48.58	48.58	48.58
3	51.5	51.5	51.5	51.5	51.5	51.5	51.5	51.5	51.5	51.5	51.5
4	53.86	53.84	53.85	53.85	53.85	53.84	53.85	53.86	53.84	53.86	53.85
5	50.07	50.04	50.02	50.07	50.07	50.02	50.06	50.08	50.04	50.08	50.04
6	49.6	49.36	49.66	49.33	49.46	49.26	49.6	49.58	49.68	49.47	49.29
7	56.69	56.69	56.9	56.5	56.67	56.54	56.71	56.83	56.65	56.77	56.65
8	54.07	54	54.23	53.97	53.9	53.78	54.14	54.18	54.1	53.96	54.04
9	55.36	55.4	55.72	55.47	54.96	55.34	55.29	55.51	55.13	55.61	55.41
10	58.87	58.56	59.17	58.77	58.21	58.38	58.52	59.25	58.42	58.87	58.86
11	57.11	56.73	57.37	56.77	56.63	56.52	57.21	57.04	57.03	56.91	56.83
12	66.65	66.66	66.87	66.7	66.4	66.65	66.9	66.42	66.72	66.44	66.81
13	59.18	59.25	59.5	59.22	58.92	59.19	59.16	59.29	59.16	59.33	59.16
14	57.88	57.71	57.79	57.72	57.87	57.86	57.59	57.94	57.61	57.84	57.93
15	65.2	65.02	65.13	65.18	65.02	64.92	65.2	65.2	64.83	65.27	65.22

Table C.3 Minimum makespan generated from RKHS

Average Best	*POP*		*CX*			*TOP*			*BOT*		
Instance	***30***	**60**	**0.1**	**0.5**	**0.9**	**0.1**	**0.2**	**0.3**	**0.1**	**0.2**	**0.3**
1	48.58	48.58	48.58	48.58	48.58	48.58	48.58	48.58	48.58	48.58	48.58
2	48.58	48.58	48.58	48.58	48.58	48.58	48.58	48.58	48.58	48.58	48.58
3	51.5	51.5	51.5	51.5	51.5	51.5	51.5	51.5	51.5	51.5	51.5
4	53.83	53.83	53.83	53.83	53.83	53.83	53.83	53.83	53.83	53.83	53.83
5	49.83	49.83	49.83	49.83	49.83	49.83	49.83	49.83	49.83	49.83	49.83
6	47.99	47.68	47.73	47.85	47.93	47.89	47.83	47.79	47.92	47.71	47.87
7	54.8	54.47	54.63	54.6	54.67	54.77	54.55	54.59	54.63	54.6	54.68
8	51.92	51.5	51.72	51.63	51.77	51.82	51.6	51.71	51.7	51.7	51.73
9	52.03	51.66	51.84	51.7	52	51.85	51.82	51.87	51.84	51.85	51.85
10	54.76	54.56	54.71	54.6	54.67	54.7	54.67	54.61	54.7	54.61	54.67
11	52.82	52.17	52.57	52.44	52.48	52.58	52.37	52.54	52.44	52.5	52.54
12	63.09	62.69	62.94	62.82	62.91	62.99	62.75	62.93	62.85	62.77	63.05
13	54.37	53.4	54.27	53.46	53.93	53.99	53.79	53.87	53.68	53.74	54.23
14	53.53	52.64	53.41	52.67	53.17	53.23	53.01	53.01	52.99	52.96	53.31
15	61.88	61.43	61.89	61.63	61.46	61.74	61.51	61.72	61.54	61.65	61.78

Table C.4 Minimum makespan generated from RKGA