ADVANCED MACHINE LEARNING ALGORITHMS IN MANUFACTURING SCHEDULING PROBLEMS

by

BILAL AL MULA ABD

A dissertation submitted to the

School of Graduate Studies

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Doctor of Philosophy

Graduate Program in Industrial and Systems Engineering

Written under the direction of

Myong K. Jeong

And approved by

_____

_____

_____

_____

New Brunswick, New Jersey

OCTOBER, 2018

ABSTRACT OF THE DISSERTATION

Advanced Machine Learning Algorithms in Manufacturing Scheduling Problems

By BILAL AL MULA ABD

Dissertation Director:

Professor Myong K. Jeong

Scheduling is a master key to succeed in the manufacturing companies in global competition. Better process scheduling leads to competitive advantage by reducing production cost and increasing productivity. Global competition has obliged the companies to expend their investments in new manufacturing systems. With the arises of these new systems, many of manufacturing problems have appeared such as scheduling problems has gained attention by researchers. Especially, it is important to develop new methodologies in order to improve manufacturing scheduling effectiveness of these new sophisticated systems. Many of researchers have used machine learning for scheduling problems because it is discovering implicit knowledge of expert schedulers that can applied for future schedules generation. In addition, machine learning is able to create flexible schedules depending on the state of the system.

In this dissertation, we present several methodologies for machine learning to scheduling in different manufacturing processes. For the scheduling problem of traditional industries, we first

present a machine learning approach for dynamic scheduling of multiple machines. Existing dynamic scheduling algorithms based on classification methods that do not utilize all the available data for the better scheduling problem. A regression-based dynamic scheduling (RDS) algorithm is proposed to improve scheduling performance of classification-based dynamic scheduling. Due to the unknown relationship between predictor variables and output variables, kernel ridge regression is presented to predict the performance of the scheduling based on system status attributes. The scheduling outputs of the proposed RDS algorithm is evaluated with scheduling results of all combinations of dispatching rules from the static job shop scheduling and a classification-based dynamic scheduling.

For the scheduling problem of semiconductor manufacturing system, we present a new machine learning algorithm for complex semiconductor scheduling. An adjustable dispatching rule (ADR) that calculates weighted sum of control factors for determining which job should be processed first. Then, to find near-optimal weight values of the ADR for improving system performance, the real coded genetic algorithm (GA) with fitness approximation is proposed. For the fitness approximation, kernel ridge regression and polynomial regression are applied by using relatively small number of fitness evaluations. The performance of the proposed algorithms is evaluated by using an extensive experiment with existing dispatching rules, fixed weights, and GAs without fitness approximation.

Finally, in order to improve the scheduling performance of semiconductor wafer fabrication, we propose new variable ranking algorithms to identify the contributions of each input variable to output variable. We present a new kernelized general dominance weight (GDW) for ranking of scheduling factors in semiconductor manufacturing system. To build kernel version of GDW, the relevance vector machine regression technique is applied.

Acknowledgement

First and foremost, I should love to extend heartfelt appreciation to my advisor, Professor Myong K. Jeong, for his support and guidance. Professor Jeong has the determination and insight of true scholars and he is the most patient and supportive mentors who have kindly concerned and considered my academic improvements. I am truly grateful to Professor Myong K. Jeong, an exceptional researcher and teacher. Since the beginning of my studies at Rutgers University, he has encouraged me and helped me with his precision and unfailing support.

I must also thank my dissertation committee members, Professor Weihong Guo, Professor Zhimin Xi and Professor Jie Gong for their expertise, support, intellectual insights and time.

I must to thank my parents, especially my father's spirit that passed away last year who have prayed and supported for the completion of my study and they have been everything I needed to go through tough times during the program.

I could not have done this dissertation and my Ph. D degree without the support from my wife, Aula Hamod, who has been there for me for the last six years. I also thank all the members of my kids (Dalya, Fatima, Linda and Ali) who have been there for me and have given all of the love, support, encouragement, and dedication. I would like to thank Dr.Minkoo Kang for his recommendations and support. I acknowledge all my friends at Rutgers University for being great colleagues and friends, especially Behnam, Ali, Bao, Mehmet and Jeongsub.

I must thank my siblings (Kassem, Hanady, Ali, Jalal, Nawers, Thora, Ansam and Tamara, These past six years, being far from you, was quite a journey for me. I could not be where I am today, without the love and support of every one of you.

I also want to thank Sayed Jaffer and his family for support and encouragement throughout the course of my Ph.D. studies.

I want to thank government of Iraq (The Ministry of Higher Education and Scientific Researcher in Iraq) for sending me to the United States for school on their expenses likewise Rutgers the State University of New Jersey for providing me one of the best educational facilities in the United States and affordable housing.

Last but not the least, my family and the one above all of us, I thank Allah for answering my prayers and giving me the strength to endure despite on the moment of wanting to give up the study. Thank you so much.

Table of Contents

List of Tables

List of Figures

CHAPTER 1
Introduction

1.1 Overview

Scheduling is defined as the resources allocation to work along time periods. The goal of scheduling is optimization of objectives by creating an effective schedule (Pinedo, 2016). Scheduling is used to allocate and plan a common set of resources such as, human resources, machinery resources, purchase materials in the manufacturing system and production processes. The area of manufacturing scheduling aims to actively assign resources as machines to some sort of jobs.

Many of researchers have used machine learning for scheduling problems because machine learning has the capability to use the scheduling knowledge to build the best schedule for the manufacturing system. In addition, Machine learning based schedulers are easy to be integrated with different decisions such as sensor monitors and diagnostic systems. Furthermore, reasoning ability enables the scheduling systems to perform more reactive in addition to predictive (Shaw et al., 1992).

Machine learning that is used by a scheduling system usually it uses a simulation system to build various manufacturing system situations and select the best schedule for each situation. Then, a machine learning technique takes experience from the training data sets to do the future decisions for scheduling (Priore et al., 2014).

The semiconductor manufacturing is identified by an expanding complication of manufacturing operations and have the multiple steps of processes such as limitations number, accurate geometries to recognize on chips, growing cases of automation combined with tools and costs, and high quality requirements from the customers (Yugma et al., 2015, Fenner et al., 2005). Semiconductor manufacturing operations may be divided into four steps: wafer probing, wafer fabrication, assembly or packing, and final test. Especially, in the wafer fabrication stage, the lot is the minimum unit to process and each lot consists of a fixed number of wafers. The wafer fabrication can be represented as a procedure composed of reentrant sequential process flows (Yugma et al., 2015).

1.2 Dissertation outline

In this dissertation, we present advanced machine learning approaches to solve complex scheduling problems in traditional manufacturing systems and semiconductor manufacturing systems. This dissertation is organized as follows. Chapter 2 presents a machine learning approach for dynamic scheduling of multiple machines. Chapter 3 proposes an adjustable dispatching rule (ADR) and real coded genetic algorithm with fitness approximation for scheduling problem of semiconductor wafer fabrication. Chapter 4 presents a scheduling algorithm based on the kernelized general dominance weight for semiconductor wafer fabrication. Finally, Chapter 5 summarizes the research results and presents the conclusion.

CHAPTER 2
Machine Learning Approach for Dynamic Scheduling in Job Shop Systems

2.1 Introduction

Scheduling is a useful way for manufacturing systems where global competition has a tremendous effect on outputs of a process. The efficient scheduling leads the industrial companies toward the success. Otherwise, the ineffectual scheduling creates bottlenecks in various manufacturing system operations. The definition of scheduling is the assignment of jobs to machines through time periods. It is an important process for optimization of the objectives (Pinedo, 2016). The target of scheduling is allocation jobs to machines to reduce the process time and the manufacturing cost (Shaw et al., 1992).

In manufacturing systems, a flow shop (Shakhlevich et al., 1998, Jeong et al., 2005) is referred to an environment where machines are set in series and each job goes through a series of operations. Except for a few extensions of the flow shop scheduling problems like a two-machine flow shop scheduling problem that can be fixed optimally but the other problems are NP-hard. All jobs go through the same route in a flow shop environment. When the sequence are not necessary the same for every job but they are fixed, the environment is called a job shop. Flow shop scheduling is also NP-hard (Sotskov et al., 1995) as it is the special case of job shop scheduling.

Many approaches for solving the flow shop and job shop scheduling problems may be utilized. These approaches can be broken down to a few groups: the heuristic, the analytical, the artificial

intelligence based algorithms and the simulation (Priore et al., 2014). In the analytical approach, the job shop and flow shop scheduling problems are captured with optimization models. Lack of efficiency for large-scale problems is the main drawback of this approach. Therefore, in order to overcome this kind of drawbacks, heuristic approaches that are usually dispatching rules, can be utilized. A dispatching rule means process first a job that is waiting in the queue with the most important of priority. In case that a machine has been freed. Changing the rules dynamically depending on the state of the system can produce interesting outcomes.

In the literature, there are two types of methodologies to change dispatching rules. In the first one is selecting the dispatching rule with best performance from a group of simulated dispatching rules. In the second technique i.e. artificial intelligence, simulation is used to generate different processing system situations and the best scheduling for each specific case is chosen. In this approach a set of simulations are considered as training examples. Then, a machine learning algorithm is applied on the training examples to build models that can be used for future scheduling. Examples of such algorithms include inductive learning and neural networks.

Su and Shiue (2003) developed an algorithm that incorporate genetic algorithm (GA) with inductive learning. In their technique, for a given system attributes subset, a decision tree is generated by applying the inductive learning algorithm. Metan and Sabuncuoglu (2005) proposed improved techniques that use operation control charts, inductive learning and simulations. Their technique, first a decision tree is generated by applying the features of the system. Then, dispatching rules are chosen from the decision tree for per period of scheduling. C4.5 (Quinlan, 1993) is used as the learning algorithm and the proposed technique is performed on a job shop system (Baker, 1984) to minimize the average tardiness. This technique was later improved by

Metan et al. (2010). Li and Olafsson (2005) proposed a new method for creating new scheduling rules from the data of each job. Their proposed method has the advantage of discovering implicit knowledge of expert schedulers that can applied for future schedules generation. Olafsson and Li (2010) proposed an extended version of their initial work as well.

For the second approach i.e. neural networks, a work has been conducted by Min and Yih (2003) which proposed a method for scheduling by picking of dispatching rules. Competitive neural networks (CNNs) are presented to get the scheduling knowledge in this work. This work also uses the semiconductor fab imitation model. Guh et al. (2011) developed a system that allocates different dispatching rules for each of the machines using self-organizing map (SOM) neural networks. They present a case study that involves a modification of the model used by Montazeri and Van Wassenhove (1990). They compare their proposed approach with two alternatives that one of them uses inductive learning (Su and Shiue, 2003) and the other one uses SVMs (Shiue, 2009). They both employ the same dispatching rule in all machines.

Exterkate et al. (2016) have proposed a kernel ridge regression as a technique for predicting nonlinear relationships. They have expanded the present kernel methods to become its employ in time-series conditions model for financial enforcements and macroeconomic. The experimental implementation is to estimating four predictors U.S. macroeconomic variables, output, revenue, sales, and labor. The experimental studies state that kernel based algorithms are often show a high accuracy for predicting the performances more than the conventional linear methods that not using kernel. Cho et al. (2008) have presented a technique that consists of two parts, off-line part which uses to build a model as predicting the observations and on-line part to monitor and diagnosis a model to make a decision depending on the information that are collected from off-line part.

In most of the existing approaches for dynamic-based classification, they used only small fraction of the available training data for selecting a dispatching rule for given system state. Consequentially, other available training data, e.g. training data that do not have the best result, were completely disregarded. So, the future scheduling decisions from the result of the classification-based dynamic scheduling (CDS) may not cover all possible combinations of dispatching rules for each machine without relatively large amount of training data. To overcome this drawback, in this chapter, we proposed a dynamic scheduling algorithm that finds the best combination of dispatching rules using regression technique for job shop scheduling problem. The proposed algorithm can predict a scheduling performance with various combinations of dispatching rules.

This chapter is arranged as follows. In Section 2.2, several dispatching rules and control attributes used in this chapter are briefly described. Then, the dynamic scheduling algorithm using regression technique for job shop scheduling problem is proposed in Section 2.3. In Section 2.4, the accomplishment of the proposed algorithms is compared with that of all combinations of dispatching rules and a classification-based dynamic scheduling, and the conclusions are presented in Section 2.5.

2.2 Background

2.2.1 Dispatching Rules

Dispatching rules determine which job will be processed on next at a given machine. Many of researches in dispatching rules has been effective for long time and many various principles have

been proposed in the literature (Pinedo, 2016). We introduce the popular four dispatching rules as follows:

First In First Out (FIFO): This rule selects a job that has been arrived at the queue first. FIFO, a dispatching rule under which the jobs are sequenced by their arrival times, is easy to implement and an effective rule for minimizing the maximum flow time and its variance.

Shortest Processing Time (SPT): SPT chooses the jobs that have shortest processing time and handles to complete them first. When most of the jobs cannot meet their due dates SPT effectively minimizes the total tardiness of all jobs. This rule is one of the most used one for its simplicity.

Minimum Slack Time (MST): The MST rule detects the insistence of a job by its slack time. Slack time is expressed as the temporal difference between the latest time that a job must be started and the earliest time that a job can be started. This rule selects a job that has least slack time.

Earliest Due Date (EDD): According to EDD rule, earliest due date that belongs to a job is chosen first. If a group of jobs that not dependent, each job has different arrival time and different a due date, to be ensures all the jobs perform by their due date, this rule can create an efficient schedule for all jobs in a group by their due date.

2.2.2 Control Attributes

The control attributes which include information of jobs and system state can be used as the input variables for machine learning algorithms. We introduce several control attributes as follows:

Work In Process (WIP): This attribute is a partially finished jobs waiting for completion. Optimal production management aims to minimize WIP because it requires storage space. This attribute can be calculated in the system as the rate number of jobs.

Average System Utilization (ASU): This attribute is the mean utilization of the manufacturing system. In the single machine model, the ASU can be calculated as mean arrival time of jobs divided by mean processing time of jobs.

Average Remaining Processing time (ARP): ARP is the sum of processing time jobs that is remaining in system divided by jobs number that need to be processed.

Average Slack Time (AST): Slack time is the rate of time a job can be delayed without causing another job to be delayed or impacting the finishing time. Usually a smaller AST is more desirable in real-time systems.

Maximum Relative Machine Workload (MRMW): MRMW which checks workload for the machines to detect a bottleneck through time. This attribute is defined as the rate of the maximum workload to system utilization.

Average Queue Length (AQL): This attribute stores the average queue length of all queues for the scheduling period.

2.3 Dynamic Scheduling Algorithm for Job Shop Problems

2.3.1 Problem Description

Let us define the scheduling problem for a job shop. There is a set $\boldsymbol{\mathcal{J}} = \{J_1, \dots, J_n\}$ of n jobs that are to be processed on a machine set $\boldsymbol{\mathcal{M}} = \{M_1, \dots, M_m\}$ of m machines as shown in Figure 2.1. For each job J_i, there are a due date d_i and release time r_i. Each machine $M_k \in \boldsymbol{\mathcal{M}}$ allows to implement one job $J_i \in \boldsymbol{\mathcal{J}}$. The job J_i composes of a sequence of n_i operations $o_1^i, \cdots, o_{n_i}^i$, where $M_{o_q^i} \in \boldsymbol{\mathcal{M}}$ and $1 \leq q \leq n_i$, being given in advance. The execution of o_q^i cannot start before the execution of o_{q-1}^i has been completed, for $q = 2, \dots, n_i$. Each operation o_q^i requires an uninterrupted period of processing time $p_{iq} \geq 0$.

We assume that there are no failures for all machines, no rework for all jobs and an unlimited buffer space in between two machines.

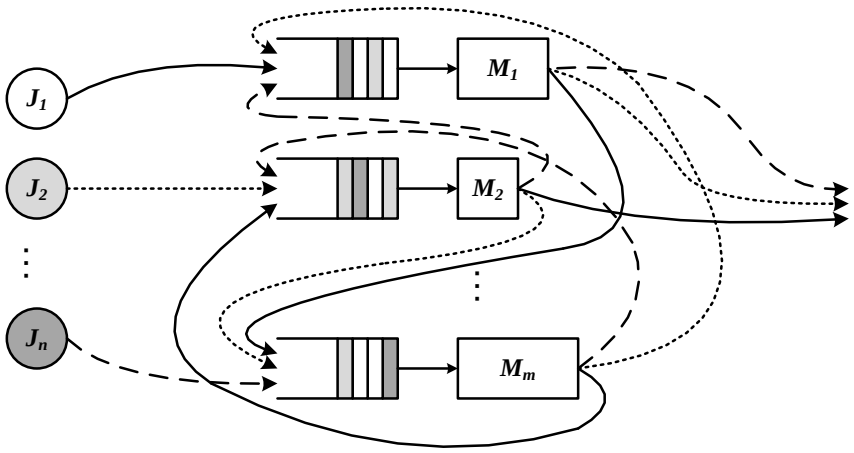


Figure 2.1 Job shop scheduling problem with n jobs on m machines

In this chapter, we consider the mean tardiness of all jobs as a performance measure for job shop scheduling problem. The tardiness of job J_i is defined as $t_i = \max(C_i - d_i, 0)$, where C_i and d_i is the completion time and due date of the job J_i, respectively. It represents how long after the due date a job was completed and can measure due date performance of the schedule. So, the mean tardiness can be calculated as $T = \sum_{i=1}^{n} t_i \, / n$.

A preliminary categorization could deal with static and dynamic scheduling. In static scheduling problem, a limitation is represented by scheduling all jobs at the same time. In static case, the schedule is obtained one time. While, in dynamic scheduling problem, only one job is scheduled in advance and the other jobs are scheduled dynamically in the system. In this chapter, it is defined as static scheduling when a combination of dispatching rules is fixed for all machines until all jobs are processed. For dynamic scheduling, a combination of dispatching rules can be chosen depending on the system state.

2.3.2 Classification-based Dynamic Scheduling (CDS)

Most of the existing approaches for dynamic scheduling of job shop problems that based on data mining methodologies, the researchers used the classification techniques to determine the solutions of dynamic scheduling. It means that training examples are obtained for various system states in advance. Each training example has the values of control attributes for representing the system state, and the best combination of dispatching rules. Then, the most similar one to current system state can be chosen for dynamic scheduling. The procedures of the classification-based dynamic scheduling are shown in Figure 2.2.

Off-line Classification Model Development

Step 1. Generate observations

1-a. Repeat for each combination of dispatching rules

1) Select a combination of dispatching rules for all machines
2) Run the simulation until all jobs are processed
3) Note the performance value

1-b. Select the combination of dispatching rules corresponding to the best performance value

1-c. Generate an observation with a vector of control attribute values with corresponding best combination of dispatching rules

1-d. Go to step 1-a until n_{obs} observations are obtained

Step 2. Develop the classification model

2-a. Divide n_{obs} collected observations into n_{tr} training data and n_{te} test data

2-b. Build the classification model to fit on n_{tr} training data

2-c. Using n_{te} test data to evaluate the achievement of the classification model

On-line Dynamic Scheduling

Step 3. Classification-based dynamic scheduling

3-a. When a job is finished in any machines, evaluate the values of control attributes

3-b. Select the best combination of dispatching rules for all machine using classification model in step 2

3-c. Go to step 3-a until all jobs are processed

Figure 2.2 The procedures of the classification-based dynamic scheduling

Figure 2.3 shows an example of the results of the classification. In this example, the CART algorithm is used to generate a decision tree as the results. In the classification problem, nodes except leaf nodes represent the conditions for input variables and the leaf nodes represent each class. The decision tree can be analyzed into decision rules, where the result is the involvements of the leaf node, and the conditions along the path from a conjunction in the if-then clause.

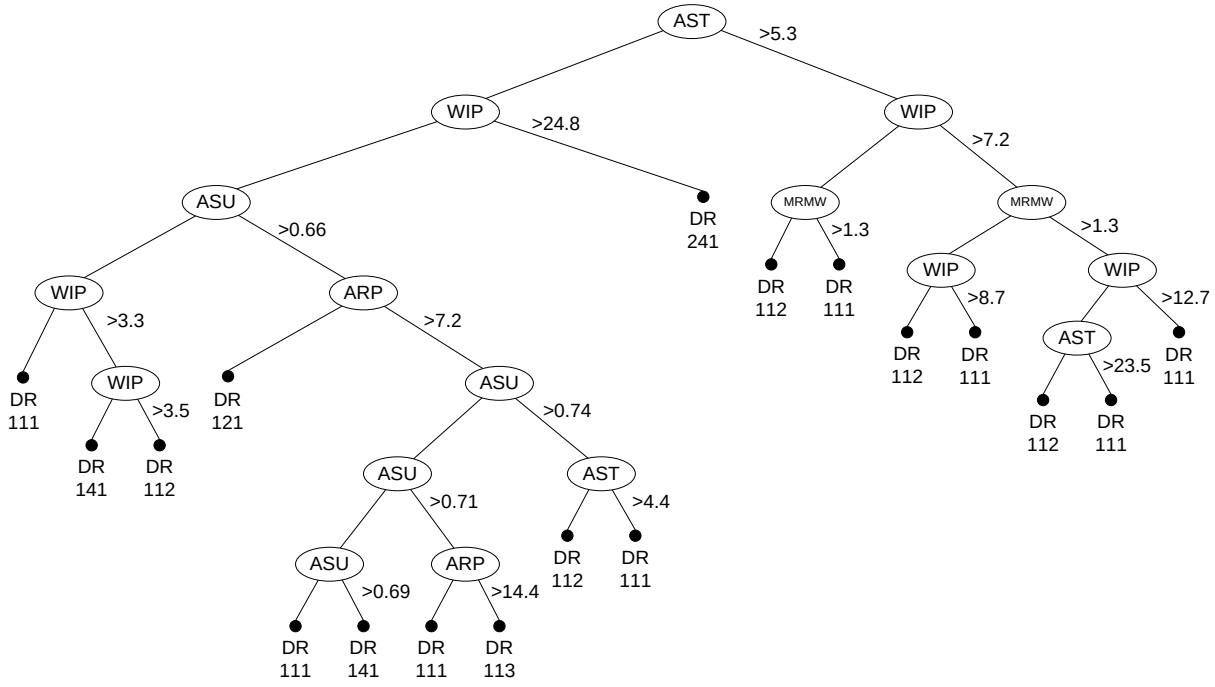


Figure 2.3 An example of the results of the classification

Note that in Step 1-a to 1-b, to obtain one observation, $(n_{dr})^m$ simulation runs are required where n_{dr} is the number of dispatching rules and m is the number of machines. In spite of the results from $(n_{dr})^m$ simulation runs, only one result that shows the best performance is obtained as an observation for the classification-based dynamic scheduling. It means that results of $(n_{dr})^m - 1$ simulation runs are completely wasted. Moreover, relatively large number of observations may be required to get the all possible combination of dispatching rules in the classification-based dynamic scheduling because the result of the classification-based dynamic scheduling with small number of observations cannot cover all combinations of dispatching rules. For example, the result of the decision tree in Figure 2.3 has only 18 leaf nodes by using 100 observations. To overcome this drawback, we proposed a regression-based dynamic scheduling algorithm that utilizes all results

from $(n_{dr})^m$ simulation runs to find the best combination of dispatching rules and estimate their performance value.

2.3.3 Regression-based Dynamic Scheduling (RDS)

In order to find the best combination of dispatching rules by estimating their performance value using the values of control attributes, a set of regression models should be developed in advance. The total number of regression models can be calculated as $n_{reg} = (n_{dr})^m$, where n_{dr} is the number of dispatching rules, and m is the number of machines. If n_{obs} observations are required for each regression model, the total number of observations for all regression models should be $n_{obs} \cdot (n_{dr})^m$. The procedures of the regression-based dynamic scheduling are shown in Figure 2.4.

Off-line Regression Model Development

Step 1. Generate observations

1-a. Repeat for each combination of dispatching rules

(1) Select a combination of dispatching rules for all machines
(2) Run the simulation until all jobs are processed
(3) Note the performance value
(4) Generate an observation with a vector of control attribute values with corresponding performance value for given combination of dispatching rules

1-b. Go to step 1-a until n_{obs} observations are obtained for each combination of dispatching rules

Step 2. Develop the regression model for each combination of dispatching rules

2-a. Repeat for each combination of dispatching rules

(1) Divide n_{obs} collected observations into n_{tr} training data, n_{va} validating data and n_{te} testing data
(2) Build the regression models to fit on n_{tr} training data for given combination of dispatching rules
(3) Adjust the regression models based on the result of the comparison in n_{va}
(4) Assess the performance of the regression model using n_{te} test data

On-line Dynamic Scheduling

Step 3. Regression-based dynamic scheduling

3-a. When a job is finished in any machines, evaluate the values of control attributes

3-b. Repeat for each combination of dispatching rules

(1) Select the regression model for given combination of dispatching rules
(2) Calculate the estimated performance value using selected regression model and the values of control attributes from Step 3-a
(3) Note the estimated performance value

3-c. Select the combination of dispatching rules corresponding to the best estimated performance value in step 3-b

3-d. Go to step 3-a until all jobs are processed

Figure 2.4 The procedures of the regression-based dynamic scheduling

In step 1, we initialize and generate the observations that system needs to get the performances. Simulation studies section have more details for this step.

In step 2, we present a kernel ridge regression (KRR). The mechanism of KRR depends on usual ridge regression and least squares regression. The result prediction for ridge regression is shown in equation (2.1) and (2.2).

$$\beta = (X^T X + \lambda I)^{-1} X^T y \tag{2.1}$$

$$\hat{y} = X\beta \tag{2.2}$$

where β is the regression coefficients and $\boldsymbol{\lambda}$ is the parameter of ridge regression. The main idea of KRR, presents a high-dimensional space of predictors by using shrinkage or ridge term to avoid overfitting to predict the regression coefficients. So, the result of KRR can be written as

$$\hat{y} = y^T (K + \lambda I)^{-1} k \tag{2.3}$$

where $K_{ij} = x_i x_j$ and it is called kernel matrix and $k_i = x_i x^{'}$ and it is called kernel function. To get on $\hat{y}$ in kernel ridge regression we need to recalculate K and k instead of explicit transformations $x \to \phi(x)$.

As is shown in equation 2.3 kernel ridge regression has kernel trick that is distinguished from ordinary ridge regression. Kernel trick increases the improvement of computational efficiency. The important process to use of kernel trick is selecting ϕ that produces to determine an accurate

kernel function k_i (means choosing the appropriate kernel). So, we present polynomial kernel to incorporate with ridge regression.

In off-line case the observations have three parts, training data, validating data and testing data. The parameters for kernel polynomial ridge regression as number of order and lambda are optimized. As shown in Step 2-a in Figure 2.4, n_{obs} collected observations are divided into n_{tr} training data, validation data n_{va} and n_{te} test data for each combination of dispatching rules. A regression model for given combination of dispatching rules is initially fit on n_{tr} training data. The training data has the pairs of input variables and the corresponding answer, which is commonly denoted as the target. Then, the regression model is running with the training data and produces a result, which is then compare with the target, for each observation. The coefficients of the regression model are adjusted based on the result of the comparison in n_{va}. The test data is used to provide an unbiased evaluation of the regression model. The regression models for all combination are built after modified those models in validation data and examined them in testing data.

In step 3, on line situation is started. When a job is processed on any machine, the proposed algorithm calculates the values of control attributes and here the values of control attributes consider a vector has $(1 \times N)$ where 1 represents an observation and N represents the number of predictor variables. Depending on regression models and the parameters that already optimized in off-line situation, the system can be predicted the new performance values for all combination of dispatching rules. After that the system starts comparing between the performance of current dispatching rule and all other combination of dispatching rule to choose the dispatching rule that gives best performance. So, the system can change to the dispatching rule that has best

performance. Usually, the ordinary least squares procedure presupposes that the number of predictor variables should be less or equivalent to the number of observations to prevent overfitting problems. But even we have that case, KRR may obtain a good in-sample fit in dynamic scheduling. The structure of the observations for each regression model consists of the values of control attributes as the independent variables and the performance value as the dependent variable as shown in Table 2.1.

Table 2.1 The structure of the observations for given combination of dispatching rules

Observations		**Input variables (control attributes)**				**Output Variable**
		$\mathbf{x_1}$	$\mathbf{x_2}$	…	$\mathbf{x_{n_{ca}}}$	**y**
Training data	**1**	$x_{1,1}$	$x_{1,2}$	…	$x_{1,n_{ca}}$	y_1
	…	…	…	…	…	…
	n_{tr}	$x_{n_{tr},1}$	$x_{n_{tr},2}$	…	$x_{n_{tr},n_{ca}}$	$y_{n_{tr}}$
Validation data	$n_{tr}+1$	$x_{n_{tr}+1,1}$	$x_{n_{tr}+1,2}$	…	$x_{n_{tr}+1,n_{ca}}$	$y_{n_{tr}+1}$
	…	…	…	…	…	…
	n_{va}	$x_{n_{va},1}$	$x_{n_{va},2}$	…	$x_{n_{va},n_{ca}}$	$y_{n_{va}}$
Test data	$n_{va}+1$	$x_{n_{va}+1,1}$	$x_{n_{va}+1,2}$	…	$x_{n_{va}+1,n_{ca}}$	$y_{n_{va}+1}$
	$n_{va}+2$	$x_{n_{va}+2,1}$	$x_{n_{va}+2,2}$	…	$x_{n_{va}+2,n_{ca}}$	$y_{n_{va}+2}$
	…	…	…	…	…	…
	n_{obs}	$x_{n_{obs},1}$	$x_{n_{obs},2}$	…	$x_{n_{obs},n_{ca}}$	$y_{n_{obs}}$

For each observation, n_{ca} control attributes for input variables and the performance measure for output variable are calculated after simulation running with randomly generated n jobs. These jobs can be reused for all combinations of dispatching rules. In other words, a set of n jobs can be used for each regression model because of the values of control attributes and performance measures

from the set of jobs with different combination of dispatching rules may be different. Thus, only n_{obs} set of jobs are required for developing n_{reg} regression models.

The proposed regression-based dynamic scheduling algorithm performs as follows. When a job is processed on any machines in the system, the proposed algorithm calculates all estimated performance values for each combination of dispatching rules using the current values of control attributes and regression models that are developed in advance. Then, the proposed algorithm can switch to the combination of dispatching rules that has the best estimated performance value. If the best combination of dispatching rules from the regression models and the current combination of dispatching rules are different, the proposed algorithm calculates the expected benefit of changing combination of dispatching rules using the estimated performance value of both best and current combinations of dispatching rules. Consequentially, if the expected benefit of changing combination of dispatching rules is bigger than a current value, the RDS algorithm changes the current combination of dispatching rules to the best combination of dispatching rules. The pseudocode of on-line dynamic scheduling part of the proposed algorithm is shown in Figure 2.5.

Regression-based Dynamic Scheduling (RDS) for Job Shop Problems

```
/* CDR is the combination of dispatching rules */
/* PV is the performance value */
1 begin
2   while all jobs are processed on the system do
3     if a job is processed for any machines do
4       Calculate the current values of control attributes;
5       for each CDR do
6         Calculate the estimated PV using the regression model for given CDR;
7         Keep best CDR and its estimated PV;
8       end
9       if best CDR /= current CDR do
10        if threshold < difference of PVs for best and current CDRs do
11          Current CDR ← best CDR;
12        end
13      end
14    end
15  end
16 end
```

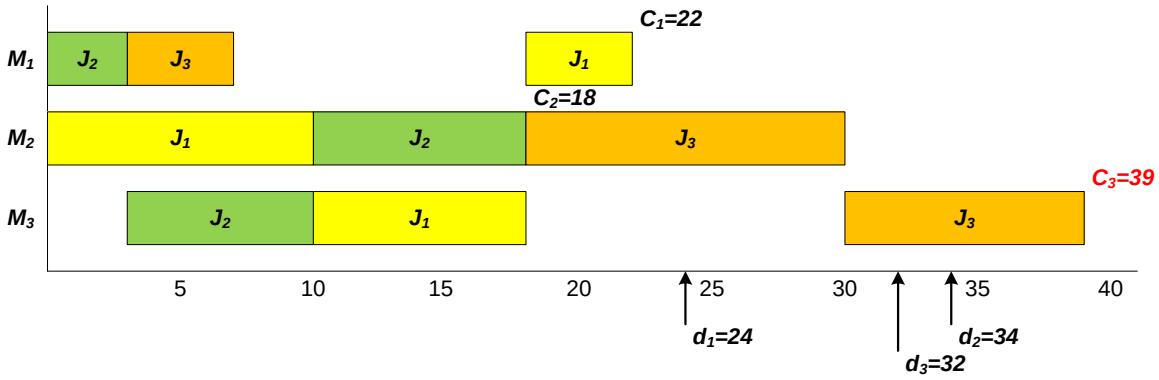
Figure 2.5 Pseudocode of the RDS algorithm

2.3.4 An illustrative Example with *n* Jobs on *m* Machines

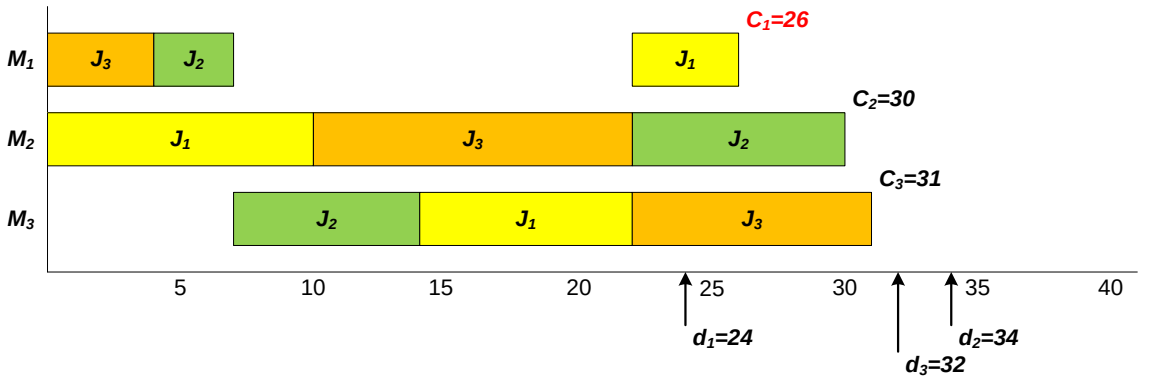
Let us consider a simple job shop scheduling problem as follows. Let assume there are 3 jobs (J_1, J_2 and J_3) that are to be processed on 3 machines (M_1, M_2 and M_3). For each job J_i, there are a release time r_i, processing time on the k-th machine p_{ik}, due date d_i and their processing order o_k^i as follows.

$$\vec{r} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \vec{p} = \begin{bmatrix} 4 & 10 & 8 \\ 3 & 8 & 7 \\ 4 & 12 & 9 \end{bmatrix}, \vec{d} = \begin{bmatrix} 24 \\ 34 \\ 32 \end{bmatrix}, \vec{o} = \begin{bmatrix} 2 & 3 & 1 \\ 1 & 3 & 2 \\ 1 & 2 & 3 \end{bmatrix} \quad (2.4)$$

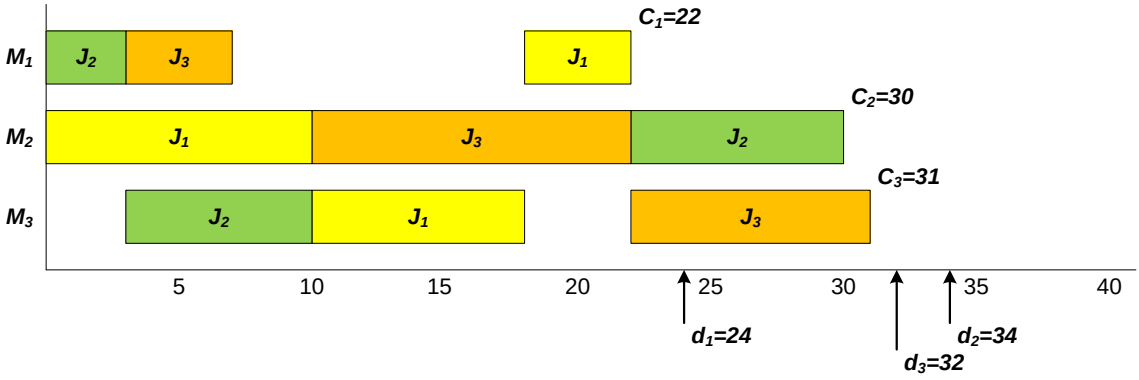
In this example, we consider three cases as follows: (a) each job is processed by SPT rule for all machine, (b) each job is processed by EDD rule for all machine, and (c) each job is processed by SPT rule before time 10, then the rule is change to EDD rule for all machine. Figure 2.6 is shown the results of these cases.



(a) Each job is processed by SPT rule for all machines



(b) Each job is processed by EDD rule for all machines



(c) Each job is processed by SPT rule (before time 10) and EDD rule (after time 10) for all machines

Figure 2.6 Scheduling results of three cases for the example of job shop scheduling problem

For case (a) and case (c), when job 2 and job 3 are arrived at the same time on machine 1, job 2 can be processed first by the SPT rule because p_{21} is less than p_{31}. On the other hand, job 3 should be processed first by the EDD rule because d_3 is less than d_2 at time 0 for case (b). At time 10 in case (a) and case (c), job 2 and job 3 compete for processing on machine 2. According to the SPT rule, job 2 is processed first because p_{22} is less than p_{32} in case (a). For case (c), job 3 is processed first because d_3 is less than d_2.

As a performance measure, mean tardiness of the scheduling results of three cases can be compared as follows. The mean tardiness of all jobs can be calculated as $\frac{\sum_{i=1}^{n} \max(0, C_i - d_i)}{n}$. So, the mean tardiness of the case (a), (b) and (c) is $\frac{7}{3}, \frac{2}{3}$ and 0, respectively. The result shows that the case (c) is the best solution with minimum mean tardiness for the example of job shop scheduling problem.

2.4 Simulation Studies

In order to measure of efficiency of the presented algorithm, Matlab simulation studies are performed for scheduling n jobs on three machines in job shop problem. There are four dispatching rules i.e. SPT (1), EDD (2), MST (3) and FIFO (4), for each machine. Then, the combination of dispatching rules for three machines is denoted by using 3-digit number from 111 to 444. For example, the combination 134 means that SPT (1), MST (3) and FIFO (4) is selected for machine 1, 2 and 3, respectively. So, the total number of all combinations of dispatching rules is $(4^3) = 64$.

To get 100 observations for each combination of dispatching rules, the following simulation runs are required repeatedly. For each simulation run, 50 jobs are randomly generated. Each job has its release time, processing times for each machine, due date and orders for processing route over

machines. When all jobs are processed, the values of control attributes i.e. WIP, ASU, ARP, AST, MRMW and AQL, and the mean tardiness of all jobs can be calculated for generating an observation. As an example, Table 2.2 states the simulation results for all combinations of dispatching rules with a set of jobs. The right most column of Table 2.2, named CDR, means the combination of dispatching rules.

Table 2.2 An example of the simulation results for all combinations with a set of jobs

	Input variables						**Output variable**	**CDR**
	WIP	**ASU**	**ARP**	**AST**	**MRMW**	**AQL**	**Mean Tardiness**	
1	6.97	0.82	10.44	3.64	1.14	4.51	14.73	111
2	7.31	0.83	10.01	4.50	1.14	4.83	15.66	112
3	7.32	0.83	9.95	4.65	1.14	4.85	15.52	113
4	7.30	0.83	9.86	4.59	1.14	4.82	15.61	114
5	8.74	0.81	9.49	11.32	1.14	6.31	22.12	121
6	8.88	0.81	9.78	11.71	1.14	6.45	22.58	122
7	8.99	0.81	9.76	12.03	1.14	6.56	23.11	123
8	8.85	0.81	9.62	11.72	1.14	6.42	22.56	124
…	…	…	…	…	…	…	…	…
61	10.15	0.82	7.78	16.21	1.14	7.68	28.41	441
62	10.25	0.82	7.91	16.19	1.14	7.79	28.98	442
63	10.19	0.82	7.80	15.97	1.14	7.72	28.49	443
64	10.45	0.82	7.73	16.84	1.14	7.98	29.82	444

In the classification-based dynamic scheduling, only one result with the minimum value of the mean tardiness i.e. the first row, represents an observation, then rest of them are wasted. However, in our proposed regression-based dynamic scheduling, all results can be contributed for improving regression models.

After generating observations, to develop the regression model for each combination of dispatching rules, 100 collected observations are divided into 70 training data, 10 validation data and 20 test data. Then, the regression model is built to fit on training data. Validation data is used to modify the parameters. To assess the performance of the regression model, test data is used. Table 2.3 shows that the example of the observation data for given combination of dispatching rules.

Table 2.3 The example of the observations for given combination of dispatching rules

		Input variables						**Output variable**
		WIP	**ASU**	**ARP**	**AST**	**MRMW**	**AQL**	**Mean Tardiness**
Training data	1	6.97	0.82	10.44	3.64	1.14	4.51	14.73
	2	6.12	0.78	10.31	1.26	1.12	3.79	13.08
	…	…	…	…	…	…	…	…
	70	5.25	0.68	8.79	3.07	1.32	3.22	15.29
Validating Data	71	7.34	0.76	10.43	2.45	1.28	4.54	16.68
	72	5.98	0.81	9.93	1.94	1.49	6.51	15.92
	…	…	…	…	…	…	…	…
	80	6.41	0.79	12.94	0.34	1.17	6.08	14.07
Test data	81	5.90	0.73	10.99	1.77	1.29	3.70	15.82
	82	5.22	0.72	9.59	0.14	1.15	3.05	10.17
	…	…	…	…	…	…	…	…
	100	6.85	0.74	16.91	-0.58	1.33	4.63	18.33

The scheduling result of the regression-based dynamic scheduling is compared with all combinations of static scheduling and a classification-based dynamic scheduling for the mean tardiness. There are two simulation models with different job environments to generate jobs randomly. For model 1, the inter-arrival time, processing times for each machine and due date of

all jobs are generated by exponential distribution with different values of lambda. On the other hand, the uniform distribution with different lower and upper bounds are applied for model 2 as shown in Table 2.4.

Table 2.4 Two simulation models with different job environments

	Model 1	**Model 2**
Interarrival time	Exp. (5)	Uniform [15,90]
Processing time for machine 1	Exp. (4)	Uniform [30,140]
Processing time for machine 2	Exp. (5)	Uniform [40,160]
Processing time for machine 3	Exp. (3)	Uniform [35,200]
Due date	Release time + Exp. (20)	Release time + Uniform [500,1000]

For model 1, the results of the mean tardiness of static and dynamic scheduling are compared for all test data as shown in Table 2.5. For example, the result of static scheduling with test data 1 show that the mean tardiness is minimum of 4.3 with combination of dispatching rules 122, i.e. SPT for machine 1, EDD for machine 2 and machine 3, and maximum of 8.2 with combination of dispatching rules 421, i.e. FIFO for machine 1, EDD for machine 2, and SPT for machine 3. The mean tardiness of CDS and RDS with test data 1 is 6.9 and 5, respectively. The relative improvement between CDS and RDS is calculated as 38% $\left(=\frac{6.9-5}{6.9}\right)$ for test data 1. As shown in Table 2.5, the mean tardiness of the RDS is reduced from a minimum of 1.48% to a maximum of 70.34% from that of the CDS for all test data.

Table 2.5 Comparison of the mean tardiness of static and dynamic scheduling for model 1

Test data	Static Scheduling		Dynamic Scheduling		
	Best (CDR)	Worst (CDR)	CDS	RDS	Relative Improvement CDS to RDS
1	4.3 (122)	8.2 (421)	6.9	5	38%
2	11.2 (212)	28.4 (143)	24.6	22.4	9.82%
3	3.1 (221)	8.1 (144)	7.2	6.4	12.5%
4	12.1 (211)	31.9 (434)	26.1	25.6	1.95%
5	4.1 (212)	10.4 (444)	10.3	6.71	53.5%
6	5.3 (133)	17.2 (442)	10.6	6.4	65.62%
7	4.2 (222)	9.4 (441)	9.2	5.4	70.37%
8	7.6 (211)	17 (444)	15.2	9.9	53.53%
9	2.8 (224)	11.6 (442)	7.35	4.6	59.78%
10	16.4 (211)	33.5 (444)	29.79	27.1	9.92%
11	7.7 (211)	20.1 (443)	18.14	14.2	27.74%
12	4.5 (111)	9.8 (434)	7.3	6.9	5.79%
13	16.4 (213)	44.5 (344)	41	40.4	1.48%
14	7.6 (121)	15.6 (434)	14.4	11.1	29.72%
15	10.1 (313)	29.7 (443)	21.1	18.7	12.83%
16	7.2 (121)	19.8 (333)	11.4	10.1	12.87%
17	9.6 (122)	24.2 (243)	17.5	11.4	53.5%
18	12 (411)	33.3 (443)	30.2	27.3	10.62%
19	20.2 (112)	41.7 (444)	32	30.3	5.61%
20	8.5 (214)	22.4 (444)	16.9	14.5	16.55%

For model 2, the results of the mean tardiness of static and dynamic scheduling are also compared for all test data as shown in Table 2.6. As shown in Table 2.6, the mean tardiness of the RDS is reduced from a minimum of 0.76% to a maximum of 19.59% from that of the CDS for all test data.

Table 2.6 Comparison of the mean tardiness of static and dynamic scheduling for model 2

Test data	Static Scheduling		Dynamic Scheduling		
	Best (CDR)	Worst (CDR)	CDS	RDS	Relative Improvement CDS to RDS
1	972.1 (222)	1478.2 (211)	982.3	884.3	11.08%
2	962.8 (122)	1309.2 (144)	948.4	888.7	6.71%
3	798.9 (211)	1287.0 (144)	869.7	727.2	19.59%
4	736.5 (331)	971.7 (144)	746.7	724.1	3.12%
5	778.7 (422)	1137.5 (214)	732.9	723.8	1.25%
6	922.8 (311)	1366.7 (444)	960.7	849.4	13.10%
7	862.0 (111)	1269.7 (444)	884.1	780.2	13.31%
8	1020.0 (221)	1344.6 (444)	1192.1	1017.4	17.17%
9	898.0 (322)	1310.2 (444)	935.7	870.9	7.44%
10	1061.1 (321)	1381.8 (444)	1035.8	985.3	5.12%
11	702.25 (321)	1120.2(444)	784.45	749.1	4.71%
12	795.24 (311)	1220.6 (344)	926.03	848.1	9.18%
13	728.7 (321)	1074.4 (444)	794.32	788.3	0.76%
14	747.61 (221)	1164.5 (344)	907.51	760.6	19.31%
15	857.12 (321)	1316.6 (444)	903.78	870.2	3.85%
16	825.52 (221)	1344.3 (444)	946.78	890.2	6.35%
17	839.87 (211)	1252.9 (144)	861.38	847.6	1.62%
18	900.1 (311)	1217.8 (442)	1061.10	972.6	9.09%
19	897.99 (322)	1310.2 (444)	935.67	870.9	7.43%
20	941.54 (331)	1326.7 (144)	1033.52	1001.5	3.19%

As is shown in Table 2.5 and 2.6, RDS is significantly given the best results comparing with classification dynamic scheduling. For all testing data, the average of the mean tardiness with RDS is the best. Furthermore, in those tables, the dispatching rules number 2 (earliest due date), number 1 (Shortest processing time) and number 3 (minimum slack time) are the best respectively and the worst dispatching rule is number 4(fist in first out). The proposed algorithm is improved to optimize the performance measures (mean tardiness).So, the mean tardiness is effected directly by EDD and SPT and indirectly by MST. The proposed technique focuses to execute the most

effective dispatching rules (EDD and SPT) continually to insure minimize the performance measure. Minimum slack time (MST) is the dispatching rule that has a main impact on the fourth control attribute (AST) in input data. So, MST has a role to calculate and minimize the performance measure but less than EDD and SPT. Kernel ridge regression dynamic scheduling has ability to discover the dispatching rule that gives the best results for performance measure. In order to fit high accuracy regression models with unknown relationship between independent variables and the target, polynomial kernel ridge regression is presented to get predictive model with high accuracy. Polynomial kernel discovers the implicit control attributes in training data by computing the similarity of vectors (training set).

One of the important point in incorporation KRR with dynamic scheduling is capability of KRR to predict a high accuracy value for performance with situation that has number of control variables (input vales) more than number of observations and also when the relationship between predictor variables and dependent variables are nonlinear. So, KRR can perfectly predict the performance in dynamic scheduling case with one observation and six control attributes (input values). RDS gives the results better than CDS because RDS utilizes fraction data bigger than CDS to learn. In other words, RDS improve all combination of dispatching rules in advance to find the best combination in dynamic case while CDS just uses small fraction data that represent the best performance is obtained for each simulation running and is ignored all other combinations of dispatching rule. Therefore, the results of CDS are not quality and lacked because they are not included all combinations of dispatching rules

2.5 Conclusions

In this chapter, a dynamic scheduling algorithm based on a regression technique for job shop scheduling problems is proposed to overcome following drawbacks in the existing approaches. First, small fraction of the available data is only used and other available data were completely wasted because a dynamic scheduling was solved as a classification problem in most of existing works. Second, the classification-based dynamic scheduling (CDS) algorithm requires relatively large number of observations to cover all possible future scheduling decisions. Thus, we proposed the regression-based dynamic scheduling (RDS) algorithm using kernel ridge regression (KRR). KRR has a capability to predict a high accuracy value for performance with situation that has number of control variables (input vales) more than number of observations and also when the relationship between predictor variables and dependent variables are nonlinear. To evaluate the efficiency of the presented RDS algorithm, the comparison between the mean tardiness of the RDS algorithm and the CDS algorithm is done. As the simulation results, the mean tardiness of the RDS is reduced from a minimum of 1.48% to a maximum of 70.34% from that of the CDS for simulation model 1 and from a minimum of 0.76% to a maximum of 19.59% from that of the CDS for simulation model 2. So, the regression-based dynamic scheduling (RDS) algorithm is significantly given the best results comparing with classification dynamic scheduling (CDS).

CHAPTER 3
New Machine Learning Algorithm for Scheduling of Complex Semiconductor Manufacturing System

3.1 Introduction

The semiconductor manufacturing is identified by an expanding complication of manufacturing operations such as limitations number, accurate geometries to recognize on chips, growing cases of automation combined with tools and costs, and high quality requirements from the customers. There are four steps of operations of semiconductor manufacturing: wafer probing, wafer fabrication, packing or assembly, and final test. In the wafer fabrication stage, the lot is the minimum unit to process, and each lot consists of a fixed number of wafers. The wafer fabrication can be represented as a procedure composed of reentrant sequential process flows. (Yugma et al., 2015).

Depending on a semiconductor manufacturing system features such as probability return a job to a same machine many times, changing demand through short periods, a lot of processing steps, capacity is not balance, transforming bottlenecks, Jobs with different product types and different proportions of each type and various machines can process same jobs, make the scheduling of a semiconductor wafer fabrication (SWF) problem strongly NP-hard (Garey and Johnson, 1979, Chen, 2010).

The scheduling problem in SWF is connected with the allocation multistage process with reentrant flows on the limited number of machines. Several processing steps is required to produce a chip layer such as photolithography, chemical-mechanical polishing (CMP), diffusion, doping, etching

and film deposition. Depending on the type of product, more than 700 steps may need to produce a lot over several weeks (Mönch et al., 2011). Figure 3.1 is applied from (Yugma et al., 2015) and presents the major of the wafer fabrication process steps.

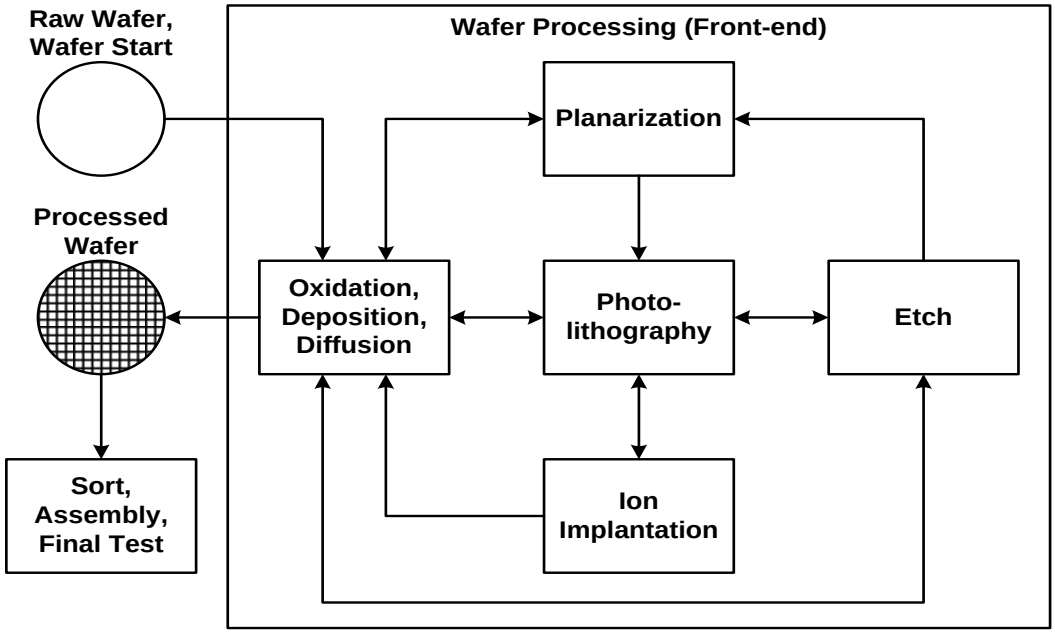


Figure 3.1 Main process steps in a wafer fabrication

Priority dispatching rules are extensively used in SWF scheduling problem. Priority dispatching rule determines which job among those waiting for service is to be scheduled in preference to other. The popular shortest processing time (SPT) rule leads to maximum throughput (Conway and Maxwell, 1962), and for minimizing the tardiness of jobs, the critical ratio (CR) rule is a best dispatching rule. Moreover, the starvation avoidance (SA) rule can ensure high utilization of bottlenecks (Glassey and Resende, 1988). However, there is no one dispatching rule that can be effective well with all situations (Uzsoy et al., 1992; Holthaus and Rajendran, 1997). These dispatching rules are easy to implement, but generally produce inconsistent results for various accomplishment metrics (Tyan et al., 2004, Thiesse and Fleisch, 2008). As a result, the advanced

priority dispatching rules have been developed. Dabbas and Fowler (2003) proposed a scheduling algorithm that combines multiple performance matrices. A dynamic multi-objective scheduling approach for semiconductor back-end processes has been developed by Sivakumar and Gupta (2006). Also, decision tree-based scheduling approaches have been proposed (Zhang et al., 2009; Olafsson and Li, 2010).

We present a novel algorithm is called adjustable dispatching rule (ADR). This algorithm calculates the weight value (w_i) for a group of factors. When a lot arrived at a station according to its predetermined order, a set of all weight factors should be calculated to determine the priority of processing. A lot has the maximum sum of all weight factors can be processed first.

Then, we proposed a real coded genetic algorithm (RCGA) to optimize the weight factor values. RCGA is considered as the effective algorithm that uses to solve the complex scheduling problems (Mahmudy et al., 2013). RCGA consists of a machine learning technique as kernel ridge regression or polynomial regression models for estimating fitness function of RCGA. The initial population is generated by using Arena simulation model and it also is used as an observation data to build the kernel ridge regression and polynomial regression models. This regression model estimates the fitness function for RCGA. Then, the results of RCGA can be added as new observation so as to enhance the efficiency of the regression models.

A hybrid genetic algorithm (GA) and data mining approach are presented in job shop scheduling problems. In this work, GA is applied for generating a learning population of good solutions (Harrath et al., 2002). Chien and Chen (2007) proposed a GA for reducing makespan of the jobs on furnace machines under complicated process conditions and time constraints. A loading problem in flexible manufacturing system (FMS) is proposed by using real coded GA which codes

a chromosome as real numbers. The real coded GA improves the system performances such as system throughput and the balance of the system (Mahmudy et al., 2013).

One of the difficulties in applying GAs to semiconductor scheduling problems is that GAs usually need a large amount of time to evaluate fitness. However, fitness evaluations are always complicated due to the cost and time. Thus, it may be useful to estimate the fitness values by developing a fitness approximation model.

Because in real manufacturing system, it will take much more effort and cost to observe the system performance under different sets of weight values for factors, our proposed RCGA with estimated fitness function from machine learning algorithms can be a good alternative to optimize the values of weight values in a real fabrication processes. This approach needs small number of observations for system performance under different sets of weight values for factors to build the prediction model for the fitness function. Then, we can proceed to optimize the weight factors over the generations of RCGA without further physical simulation run (or observing the actual system performance) under different sets of weight values for factors.

In order to overcome these challenges, we proposed an adjustable dispatching rule (ADR) that calculates weighted sum of control factors for determining which job should be scheduled first. Compared to the existing dispatching rules, the advantage of the ADR is that priorities of waiting jobs can be reassigned easily by changing weight values of control factors. Based on a relatively small number of observations, the proposed algorithm can build an accurate regression model. It may has a significant role when the cost of obtaining observations is expensive.

The arrangement of this chapter is as follows. In Section 3.2, real coded GA and the regression techniques that used in this chapter are briefly described. Section 3.3 introduces the ADR and the examples of control factors and how to calculate weighted sum of control factors to realize the ADR. Then, the real coded GA with fitness approximation to solve the scheduling problem of semiconductor wafer fabrication in Section 3.4. The comparison of efficiency between the proposed algorithm and the existing dispatching rules by using a real fab simulation model in Section 3.5, and our conclusions are given in Section 3.6.

3.2 Background

3.2.1 Real Coded GA

GA is one of the most approved evolutionary algorithm that imitate the evolution in nature. The decision variables for the GA are typically represented as binary strings. In the binary representation, the number of decision variables specifies the length of the chromosome. On the other hand, the real coded GA (RCGA) presents the real number representation for decision variables instead of the binary strings (Murugan et al., 2007).

The benefits of the RCGA as against the binary GA are follows. Real number representation performs well compared with binary representation for numerical optimization problems (Michalewicz, 1996). The binary coded GA needs more time to compute then the real coded GA (Renders and Flasse, 1996). The crossover operator is considered the main operator for searching solution space in the real coded GAs. The extensive studies of different type of crossover operators are presented in Herrera et al., (2003).

3.2.2 Polynomial Regression

Polynomial regression is a well-known non-linear regression in which the relationship between input variables and output variable is formulated as a polynomial. It is also widely used in situations where the response is curvilinear. It can be considered to be a special case of linear regression because it is linear with statistical estimation problems. The formulation of polynomial regression depends on the degree of polynomial and input variables number. For example, the second order polynomial in one variable $y = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \varepsilon$, the second order polynomial in two variables $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{11} x_1^2 + \beta_{22} x_2^2 + \beta_{12} x_1 x_2 + \varepsilon$, and the general form of the second order polynomial regression model

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \beta_{1,1} x_1^2 + \beta_{2,2} x_2^2 + \cdots + \beta_{p,p} x_p^2 + \beta_{1,2} x_1 x_2 + \beta_{1,3} x_1 x_3 + \cdots + \beta_{p-1,p} x_{p-1} x_p + \varepsilon, \quad (3.1)$$

where p is the input variables number.

The number of coefficients at different number of input variables and polynomial degrees is enumerated as shown in Table 3.1.The number of coefficients increases rapidly when the polynomial degree increases because it is $\binom{p+k}{k} = \frac{(p+1)(p+2)\cdots(p+k)}{k!}$ for p input variables and k-th order polynomial regression.

Table 3.1 Number of coefficients at different polynomial degrees and number of variables

Number of Variables	Polynomial Degree				
	k = 1	**k = 2**	**k = 3**	**k = 4**	…
1	2	3	4	5	…
2	3	6	10	16	…
3	4	10	20	35	…
4	5	15	35	64	…
5	6	21	56	106	…
6	7	28	84	165	…
7	8	36	120	246	…
8	9	45	165	355	…
9	10	55	220	499	…
10	11	66	286	686	…
…	…	…	…	…	…

3.2.3 Kernel Ridge Regression

The Kernel Ridge Regression (KRR) is proposed as a framework for estimating nonlinear predictive relations. It shows relatively consistent and good performance of prediction and can build more accurate regression models than traditional linear regressions (Exterkate et al., 2016). Ordinary least squares regression and ridge regression are used in KRR. The prediction result of the ridge regression is shown in Equations (3.2) and (3.3).

$$\beta = (X^T X + \lambda I)^{-1} X^T y \tag{3.2}$$

$$\hat{y} = X\beta \tag{3.3}$$

where β is the regression coefficients and $\boldsymbol{\lambda}$ is the parameter of ridge regression. The main idea of KRR is working to change the input variables to a space of high-dimensional in a nonlinear way.

A high-dimensional space is using to estimate a regression equation, in addition to use a penalty term to avoid overfitting. So, the result of KRR can be written as

$$\hat{y} = y^T(K + \lambda I)^{-1}k \tag{3.4}$$

where $K_{ij} = x_i x_j$ is the kernel matrix and $k_i = x_i x^{'}$ is the kernel function. To get on $\hat{y}$ in KRR, we need to recalculate K and k instead of explicit transformations $x \rightarrow \phi(x)$.

3.3 Adjustable Dispatching Rule (ADR)

To solve the priority of dispatching rule, ADR has been presented. When a lot arrived at a station according to its predetermined order, a set of all weight factors should be calculated to determine the priority of processing for waiting jobs. The semiconductor wafer fabrication model is generated to mimic real semiconductor wafer fabrication system. So, Arena simulation is developed for semiconductor wafer fabrication that allows the user to build simulation models and perform experiments (Garrido, 2009). To do so we have to consider the follows.

3.3.1 Assumptions

To simplify the model, we made some assumptions for the scheduling of semiconductor manufacturing systems with uneven setup times of the different jobs as follows:

1. Jobs of the same type have the same route of processing steps with same processing times. However, a setup time before processing can vary depends on allocated machine and job type.
2. Each station has a fixed number of parallel machines. Machines in each station are identical in nature.
3. The preemption is not allowed.
4. There are no failures for all machines, no rework for all jobs, and no inventory limits in the system.

3.3.2 Control Factors

When a job arrived at a station according to its predetermined order, a set of all control factors should be calculated by using current information of the job and machines in the station. There are 10 examples of control factors with their purpose and how to calculate the values of the control factors as shown in Table 3.2.

Table 3.2 The examples of control factors for semiconductor wafer fabrication

	Factor Name	Purpose	How to calculate the value of the factor
F_1	Recipe Change Loss	Minimizing Recipe change loss	(Maximum loss – Current loss) / Maximum loss
F_2	Chamber Availability	Maximizing Chamber Availability	The number of available idle chambers to process the type of a given product / The total number of idle chambers
F_3	Chuck Efficiency	Minimizing Chuck Arm Loss	$\begin{cases} 1 & \text{if previous Chuck} \neq \text{current Chuck} \\ 0 & \text{otherwise} \end{cases}$
F_4	Device Priority Weight	Product with highest priority should be processed first.	The value for each product can be assigned between 0 and 1 according to the priority of the product.
F_5	Moving Target Weight	Product with highest priority should be processed first.	1 – (Actual output of moving / Planned output of Moving)
F_6	Lot Location Weight	Priority according to distance from the machine	The value of this factor can be assigned between 0 and 1 according to their location.
F_7	Delay Time Weight	Priority for preventing delay	MIN{(Delay time/Threshold value for delay),1}
F_8	Wait Time Weight	Product with largest waiting time should be processed first.	MIN{(Waiting time/Threshold value for waiting),1}
F_9	Lot Priority Weight	Designated lot should be processed first.	The value of this factor can be assigned between 0 and 1 according to their lot priority.
F_{10}	Designate Step Weight	Designated step should be processed first.	The value of this factor can be assigned between 0 and 1 according to their step weight.

Recipe change loss factor ($\boldsymbol{F_1}$) means a loss for setup time between previous work and next work for a machine. If both previous work and next work have same recipe, there is no recipe change loss. The value of this factor can be calculated by the ratio of (maximum loss – current loss) to

maximum loss for a given lot. If there is no recipe change loss (i.e., current loss is 0), the value of $\boldsymbol{F_1}$ is 1. Chamber availability factor ($\boldsymbol{F_2}$) means how many chambers are available to process a given lot on the machine. If all chambers are available to process a given lot, $\boldsymbol{F_2}$ is 1. The value of this factor can be calculated by the proportion of the number of available idle chambers to the idle chambers for a given lot. It is better to process first a lot with higher value of $\boldsymbol{F_2}$ for maximizing chamber availability. The actual processing time depends on the value of $\boldsymbol{F_2}$ i.e., the actual processing time = the ideal processing time / the value of $\boldsymbol{F_2}$.

Chuck efficiency factor ($\boldsymbol{F_3}$) represent an efficiency of chuck arm usage. Each lot has either an odd chuck or an even chuck. If previous lot and next lot have different chucks (e.g., an odd chuck for previous lot and an even chuck for next lot), there is no chuck arm loss. Otherwise, there is some fixed setup time before processing next lot. A value of this factor can be assigned 0 or 1. It is better to process first a lot with different chuck from the previous one for minimizing chuck arm loss.

Device priority factor ($\boldsymbol{F_4}$) indicates the priority of the product for processing. The product with highest value of this factor should be processed first. The value of this factor for each product can be assigned between 0 and 1 according to the priority of the product.

Moving target factor ($\boldsymbol{F_5}$) represents the proportion of the unfinished output number to the planned output number. Lot location factor ($\boldsymbol{F_6}$) indicates a distance from the lot to the machine.

Delay time factor ($\boldsymbol{F_7}$) and wait time factor ($\boldsymbol{F_8}$) indicate how long a given lot has been delayed and waited for processing, respectively. The value of these factors can be calculated by the

minimum value of delay (or wait) time / Threshold value for delay (or wait) and 1. It means the value of these factors cannot be exceed a value of 1.

Lot priority factor ($\boldsymbol{F_9}$) indicates a priority of a given lot for processing. Designate step factor ($\boldsymbol{F_{10}}$) indicates a priority of a given step for processing.

3.3.3 Calculating Weighted Sum of Control Factors

To calculate weighted sum of control factors, a normalized weight w_i should be defined for control factor F_i. For the total number of control factors p, each weight w_i has a value between 0 and 1. Also, the sum of all weights is equal to 1, i.e. $\sum_{i=1}^{p} w_i = 1$. For such normalized weights, the weighted sum of control factors is simply $\sum_{i=1}^{p} w_i F_i$.

When a machine is available, the job with the highest weighted sum of control factors is selected from waiting jobs for processing. For example, if there are three jobs with values of three control factors, $J_1(0.5,1,0.3), J_2(0.8,0.2,0.5), J_3(1,0,0.4)$, and the normalized weights are $(0.5,0.3,0.2)$, then the weighted sum of three jobs can be calculated as

$$\begin{bmatrix} 0.5 & 1 & 0.3 \\ 0.8 & 0.2 & 0.5 \\ 1 & 0 & 0.4 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.3 \\ 0.2 \end{bmatrix} = \begin{bmatrix} 0.61 \\ 0.56 \\ 0.58 \end{bmatrix}.$$

It means that J_1 has the highest priority for processing with the normalized weights$(0.5,0.3,0.2)$. This example shows that job priorities can be easily reassigned by changing the normalized weights.

3.3.4 Arena Simulation Model for Semiconductor Wafer Fabrication

The simulation model of semiconductor wafer fabrication is developed by using Arena simulation software that allows the user to build simulation models and perform experiments (Garrido, 2009). For the proposed simulation model, the various processing steps of all jobs fall into eight stations: photolithography (PHOTO), etching (ETCH), chemical-mechanical planarization (CMP), cleaning (CLN), chemical vapor deposition (CVD), ion implantation (IMP), diffusion (DIFF), and metal interconnect (METAL) as shown in Figure 3.2. Moreover, arrival dock and shop exit modules are constructed for arrival and departure of jobs, respectively.

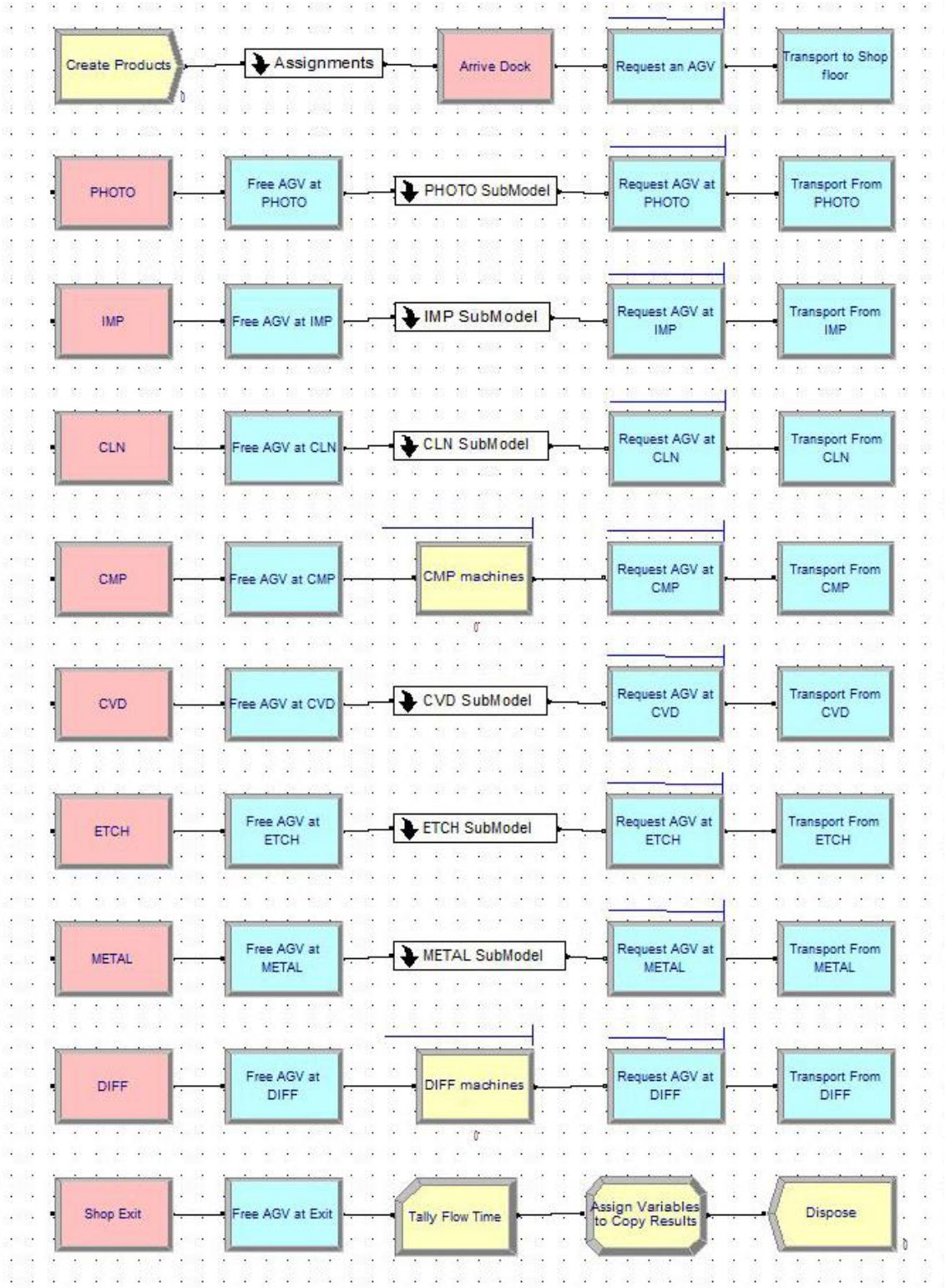


Figure 3.2 Arena simulation model for semiconductor wafer fabrication

3.4 Real Coded GA with Fitness Approximation

In this section, the real coded GA with fitness approximation to find near-optimal weights for control factors for the scheduling problem of semiconductor wafer fabrication is applied.

3.4.1 Flowchart of the Real Coded GA with Fitness Approximation

As shown in Figure 3.3, the flowchart of the real coded GA with fitness approximation starts with randomly generated input variables that represent weight values of control factors. The output variable is evaluated by using simulation model and represent performance measure that we want to improve.

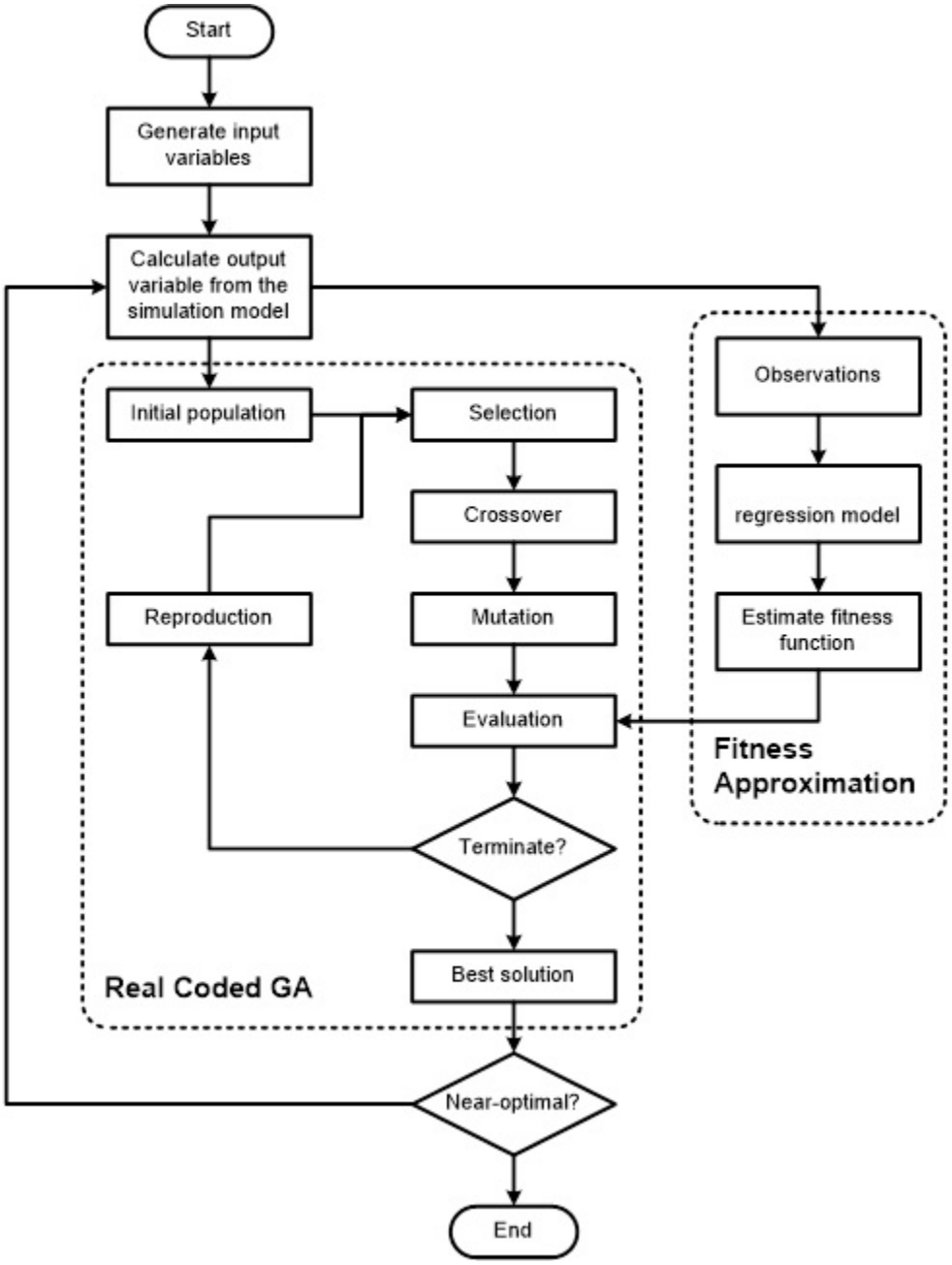


Figure 3.3 The flowchart of the real coded GA with fitness approximation

After generating a fixed number of input variables and output variable, these initial data can be used not only the initial population of real coded GA, but also the observations for finding polynomial regression coefficients.

The real coded GA initializes a population of solution and improves by repeatedly applying selection, crossover and mutation operators. Parents are chosen from the initial population. Then, offspring are produced from selected parents by using crossover and mutation operators. These offspring have to evaluate by using a fitness function in order to generate next population. When evaluation process is done, the real coded GA stores the best solution of this population and next population replaces the current population iteratively until termination condition is fulfilled.

The initial data from simulation model can be used as the observations for finding of regression coefficients to estimate the objective function of the RCGA because of the relationship between input variables and output variable is unknown. The estimated fitness function can be improved by adding new observation from the best solution of the real coded GA repeatedly.

3.4.2 Chromosome Representation

A chromosome consists of real values whose size is equal to the number of the weight values. Each element of the chromosome $x_1, x_2, \cdots, x_p$ corresponds to the real values for p weight values for control factors. The value of x_i is maintained between c_{min} and c_{max}. c_{min} and c_{max} is the minimum and maximum value of weights, respectively.

3.4.3 Selection

The selection procedure is applied to select parents from the current population to perform genetic operations. In this chapter, tournament selection method is applied for select parents. The

tournament selection includes a number of running tournaments of some individuals randomly selected from the current population. Then, the winners of each tournament are selected. The advantage of the tournament selection is that it can be easily adjusted by modifying the size of the tournament. If the tournament size is smaller, weak individuals have a larger chance to be selected.

3.4.4 Crossover Operator

To create a new generation, the operator of crossover is typically applied to a pair of individuals to generate two offspring (Goldberg, 1989). In this chapter, we use well-known single-point crossover operator. Let us assume that $P_1 = (c_1^1, c_2^1, \dots, c_n^1)$ and $P_2 = (c_1^2, c_2^2, \dots, c_n^2)$ are two parents. A position $i \in \{1,2, \dots, n-1\}$ is chosen randomly and two offspring are produced as $O_1 = (c_1^1, c_2^1, \dots, c_i^1, c_{i+1}^2, \dots, c_n^2)$ and $O_2 = (c_1^2, c_2^2, \dots, c_i^2, c_{i+1}^1, \dots, c_n^1)$.

3.4.5 Mutation Operator

In uniform mutation proposed by Michalewicz (1996), a gene is replaced with a random value between user-specified upper and lower bounds. In this chapter, the uniform mutation operator is applied. From a point $x = (c_1, c_2, \dots, c_n)$, the muted point $x^* = (c_1^*, c_2^*, \dots, c_n^*)$ is created as $c_i^* = c_i^l + r(c_i^u - c_i^l)$, where r is a random number between 0 and 1and is uniform distribution. c_i^l and c_i^u is lower bound and upper bound of the i-th gene, respectively.

3.4.6 Fitness Evaluation

The objective function of the optimization problem should be formulated into a fitness function used to measure the goodness of the solution (Mahmudy et al., 2013). However, due to features of the semiconductor manufacturing systems, the fitness function cannot be formulated using existing mathematical expressions for our problem. That is why we need to use an advanced regression model as a fitness function.

3.4.6.1 Polynomial Regression

In order to estimate the fitness function of the real coded GA, a third-order polynomial regression model is applied as follows:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \beta_{1,1} x_1^2 + \beta_{2,2} x_2^2 + \cdots + \beta_{p,p} x_p^2 + \beta_{1,2} x_1 x_2 + \beta_{1,3} x_1 x_3 + \cdots + \beta_{p-1,p} x_{p-1} x_p + \beta_{1,1,1} x_1^3 + \beta_{2,2,2} x_2^3 + \cdots + \beta_{p,p,p} x_p^3 + \beta_{1,1,2} x_1^2 x_2 + \beta_{1,1,3} x_1^2 x_3 + \cdots + \beta_{p,p,p-1} x_p^2 x_{p-1} + \beta_{1,2,3} x_1 x_2 x_3 + \beta_{1,2,4} x_1 x_2 x_4 + \cdots + \beta_{p-2,p-1,p} x_{p-2} x_{p-1} x_p + \varepsilon, \quad (3.5)$$

Where p is the number of input variables. The polynomial coefficients β's are calculated to minimizing the mean square error (MSE) of the observations.

3.4.6.2 Kernel Ridge Regression

As is shown in equation 3.4 kernel ridge regression has kernel trick that is distinguished from ordinary ridge regression. Kernel trick increases the improvement of computational efficiency.

Selecting ϕ that works to determine an accurate kernel function k_i (means choosing the appropriate kernel) is the important process to succeed of kernel trick. So, we present polynomial kernel to incorporate with ridge regression to estimate the fitness function of the real coded GA. The parameters of KRR are optimized to predict a high accuracy of regression model.

3.5 Experimental Studies

3.5.1 Problem Description

As shown in Figure 3.2, there is 8 stations and each station has fixed number of machines as shown in Table 3.3. As mentioned in Section 3.3.1, machines in each station are identical in nature. We assume that all physical distances between one station and another station are equal to 100 feet. There are 10 automated guided vehicles (AGVs) with same speed of 100 feet/min for transportation.

Table 3.3 The number of machines in each station

Station	Number of Machines	Station	Number of Machines
PHOTO	10	ETCH	4
IMP	3	METAL	4
CLN	12	CVD	3
DIFF	15	CMP	2

There are three types of jobs, J_1, J_2 and J_3, in the simulation model. The proportion of each type of jobs is 50% of J_1, 30% of J_2 and 20% of J_3. To generate step information for each type of jobs, 20 different layers are made as shown in Table 3.4. For each layer, there are step information with pairs of station name and processing time. Then, the sequence of layer IDs for the type of J_1 is

assigned from layer ID 1 to layer ID 20 in numerical order. The remaining sequences of layer IDs for the type of J_2 and J_3 are assigned randomly as shown in Table 3.5.

Table 3.4 Step information of 20 different layers

Layer ID	Step Information (Station name, Processing time (min))
1	(IMP,30) → (IMP,40) → (DIFF,300) → (PHOTO,60) → (DIFF,250) → (CLN,35) → (CLN,25)
2	(PHOTO,55) → (ETCH,40) → (CLN,25) → (CLN,30)
3	(PHOTO,62) → (IMP,45) → (CLN,25)
4	(PHOTO,62) → (METAL,50) → (CLN,25) → (CVD,30) → (DIFF,300)
5	(PHOTO,62) → (IMP,50) → (IMP,45)
6	(PHOTO,60) → (ETCH,50) → (CLN,35) → (CVD,35)
7	(PHOTO,55) → (ETCH,40) → (CLN,25) → (CLN,30) → (CMP,60) → (CLN,30)
8	(PHOTO,62) → (METAL,55) → (CLN,25) → (ETCH,40)
9	(PHOTO,62) → (ETCH,30) → (CLN,25) → (CVD,40)
10	(PHOTO,60) → (DIFF,200) → (DIFF,180) → (CLN,35) → (CLN,25) → (METAL,55)
11	(PHOTO,60) → (CVD,30) → (CLN,35) → (CLN,25)
12	(PHOTO,55) → (ETCH,40) → (ETCH,45) → (CLN,25) → (CLN,30) → (DIFF,250) → (CMP,55) → (CLN,30)
13	(PHOTO,62) → (METAL,45) → (CLN,25) → (METAL,50)
14	(PHOTO,60) → (DIFF,400) → (CLN,35) → (CLN,25)
15	(PHOTO,62) → (ETCH,35) → (METAL,45) → (CLN,25) → (CVD,35) → (CVD,40)
16	(PHOTO,60) → (CVD,30) → (CLN,35) → (CLN,25)
17	(DIFF,200) → (CLN,35) → (CLN,25)
18	(PHOTO,55) → (ETCH,40) → (CLN,25) → (CLN,30) → (CMP,60) → (CLN,30)
19	(PHOTO,62) → (METAL,50) → (CLN,25)
20	(PHOTO,55) → (ETCH,50) → (CLN,25) → (CLN,30) → (DIFF,400)

Table 3.5 Sequence of layer IDs for each type of jobs

Type of jobs	Sequence of layer IDs
J_1	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20
J_2	13,5,14,6,4,19,11,9,8,15,3,7,16,1,18,12,17,2,20,10
J_3	14,11,7,15,2,12,9,18,16,20,5,3,8,1,6,19,17,4,10,13

3.5.2 Experimental Setup

Simulation runs were conducted by executing the simulation model for 10 days of warm-up period to avoid startup bias and 30 days of simulation period. To get the observation data for fitness approximation, 50 different set of weight values are randomly generated for PHOTO station. For each simulation run with each set of weight values, the output variable is labeled by obtaining the average waiting time for all jobs processed on the PHOTO station in the simulation period. The waiting time of the job can be calculated as the difference between arrival time of the job and process start time of the job as shown in Figure 3.4.

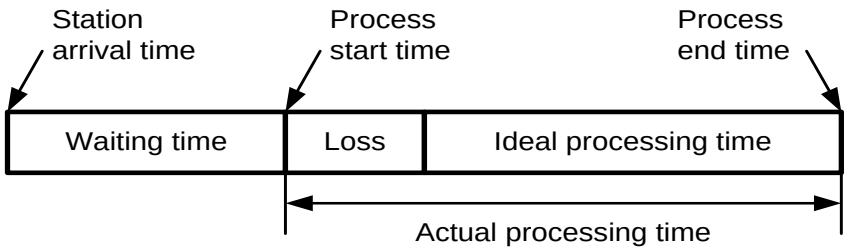


Figure 3.4 The waiting time of the job

After getting the observation data, regression coefficients are calculated by minimizing mean square error (MSE) of the observation for fitness approximation. Then, 10 of 50 observation data are randomly chosen for the initial population.

The population size, selection method, crossover and mutation operators, crossover and mutation rates, mutation probability, the number of individuals kept for the elitism, and the termination condition are fixed for a uniform testing environment as shown in Table 3.6.

Table 3.6 Testing environment for GAs

Parameter	Binary coded GA	Real coded GA
The population size	10	
Selection method	Tournament (size = 3)	
Crossover operator	Single-point crossover	
Mutation operator	Uniform mutation	
Crossover rate	0.6	
Mutation rate	0.3	
Mutation probability	0.02 (for each bit)	0.2 (for each gene)
Number of individuals kept for the elitism	1	
Termination condition	When the fitness value of best solution of current generation has reached a plateau (5 consecutive generations)	

3.5.3 Experimental Results

3.5.3.1 Comparison of the Existing Dispatching Rules, Fixed Weights and GAs without Fitness Approximation

The simulation results with the existing dispatching rules such as FIFO, SPT, EDD, and fixed weights (FW) are obtained to compare with the near-optimal solution of GAs, i.e. binary coded GA (BCGA) and real coded GA (RCGA), without fitness approximation (FA). GAs without FA

can be operated with the original fitness evaluation that means obtaining an output variable from the simulation model.

Table 3.7 Comparison of the average waiting times for the existing dispatching rules, fixed weights (FW) and GAs without fitness approximation (FA)

	Average Waiting Time (min)	Relative Improvement	
		FIFO	Fixed Weights
FIFO	52.65	-	10.08%
SPT	115.92	-120.17%	-97.98%
EDD	91.09	-73.01%	-55.58%
FW	58.55	-11.21%	-
BCGA	50.44	4.19%	13.85%
RCGA	48.97	6.99%	16.36%

As shown in Table 3.7, compared with FIFO, the average waiting time of BCGA and RCGA is reduced by 4.19% and 6.99%, respectively. Moreover, compared with fixed weights, the average waiting time of BCGA and RCGA is reduced by 13.85% and 16.36%, respectively.

The reason why the average waiting time of the FIFO is not the smallest is because the waiting time of other jobs which are waiting in the station becomes longer if the setup time, e.g. recipe change loss and chuck loss, of jobs currently being processed becomes longer. Moreover, the average waiting time of the SPT rule is the worst because the processing of jobs with a relatively large processing time is delayed due to the processing of jobs with a small processing time.

3.5.3.2 Comparison of GAs with and without Fitness Approximation Polynomial Regression (FAPR)

To compare performance of GAs with and without FAPR, the average waiting time and the number of simulation runs are obtained. GAs without FA can be operated with the original fitness evaluation that means obtaining an output variable from the simulation model. On the other hand, GAs with FA can be operate using same estimated fitness function consists of polynomial regression coefficients and input variables.

Table 3.8 Comparison of GAs with and without fitness approximation (FAPR)

	Average Waiting Time (min)			Number of Simulation Runs
	MEAN	STD	MIN	
BCGA	53.70	2.89	50.44	160
BCGA with FAPR	55.38	3.28	49.51	23
RCGA	53.45	3.01	48.97	120
RCGA with FAPR	53.02	2.54	48.71	28

As shown in Table 3.8, compared with BCGA without FA, the minimum value of the average waiting time of BCGA with FA is reduced by 1.84%. Moreover, the number of simulation runs of BCGA with FA is reduced by 85.63% compared with BCGA without FA. On the other hand, compared with RCGA without FA, the minimum value of the average waiting time of real coded GA with FA is reduced by 0.53%. Moreover, the number of simulation runs of real coded GA with FA is reduced by 76.67% compared with real coded GA without FA.

Until the termination condition of GA was satisfied, BCGA and RCGA required 160 and 120 simulation runs, respectively. In contrast, to get better results than the minimum of average waiting times of BCGA and RCGA, BCGA with FA and RCGA with FA required 23 and 28 simulation

runs, respectively. Therefore, the advantage of GAs with FA is that the number of simulation runs is significantly reduced. When our fitness approximation shows very good estimation of the original fitness function, i.e. obtaining an output variable from the simulation model, GAs with FA can get reduced the average waiting time with relatively small number of simulation runs compared with GAs without FA.

3.5.3.3 Comparison of GAs with and without Fitness Approximation Kernel Ridge Regression (FAKRR)

To compare performance of GAs with and without FAKRR, the average waiting time and the number of simulation runs are obtained. GAs without FA can be operated with the original fitness evaluation that means obtaining an output variable from the simulation model. On the other hand, GAs with FAKRR can be operated using same estimated fitness function consists of the KRR model and input variables.

Table 3.9 Comparison of GAs with and without fitness approximation (FAKRR)

	Average Waiting Time (min)			**Number of Simulation Runs**
	MEAN	**STD**	**MIN**	
BCGA	53.70	2.89	50.44	160
BCGA with FAKRR	51.88	1.68	49.32	27
RCGA	53.45	3.01	48.97	120
RCGA with FAKRR	49.82	1.83	46.50	29

As shown in Table 3.9, compared with BCGA without FAKRR, the minimum value of the average waiting time of BCGA with FA is reduced by 2.22%. Moreover, the number of simulation runs of BCGA with FA is reduced by 83.13% compared with BCGA without FA. On the other hand, compared with RCGA without FA, the minimum value of the average waiting time of real coded GA with FA is reduced by 5.04%. Moreover, the number of simulation runs of real coded GA with FA is reduced by 75.83% compared with real coded GA without FA.

Until the termination condition of GA was satisfied, BCGA and RCGA required 160 and 120 simulation runs, respectively. In contrast, to get better results than the minimum of average waiting times of BCGA and RCGA, BCGA with FA and RCGA with FA required 27 and 29 simulation runs, respectively. Therefore, the advantage of GAs with FA is that the number of simulation runs is significantly reduced. When our fitness approximation shows very good estimation of the original fitness function, i.e. obtaining an output variable from the simulation model, GAs with FA can get reduced the average waiting time with relatively small number of simulation runs compared with GAs without FA.

3.6 Conclusions

In this chapter, a new machine learning algorithm for scheduling of complex semiconductor manufacturing system is proposed. The efficient algorithms that consists of the adjustable dispatching rule (ADR) and the real coded genetic algorithm (GA) with fitness approximation has been proposed for scheduling problem of semiconductor wafer fabrication. For the ADR, we introduced the control factors and calculating weighted sum of control factors. Compared to the

existing dispatching rules, the advantage of the ADR is that priorities of waiting jobs can be reassigned easily by changing weight values of control factors. In order to find near-optimal weight values of the ADR, the real coded GA with fitness approximation is also proposed. For the fitness approximation, kernel ridge regression and polynomial regression are applied by using relatively small number of fitness evaluation. The performance of the real coded GA with fitness approximation is evaluated by using an extensive experiment with existing dispatching rules, fixed weights and GAs without fitness approximation. Compared with FIFO as one of the existing dispatching rule, the average waiting time of real coded GA without fitness approximation is reduced by 6.99%. Moreover, compared with fixed weights, the average waiting time of real coded GA without fitness approximation is reduced by 16.36%. The number of simulation runs of binary coded GA with fitness approximation polynomial regression (FAPR) and real coded GA with FAPR is reduced by 85.63% and 76.67%, respectively, compared with GAs without FA. The number of simulation runs of binary coded GA with fitness approximation kernel ridge regression (FAKRR) and real coded GA with FAKRR is reduced by 83.13% and 75.83%, respectively, compared with GAs without fitness approximation. The experiments show that the ADR and the real coded GA with both finesses approximation can find near-optimal solution of the scheduling problem for semiconductor wafer fabrication with relatively small number of fitness evaluation. It can play an important role when the cost of evaluating fitness is expensive.

CHAPTER 4
New Kernelized General Dominance Weight for Ranking of Scheduling Factors in Semiconductor Manufacturing System

4.1 Introduction

In semiconductor manufacturing, wafers are fabricated through complex multi processes. Wafer fab facilities typically consist of several general process modules: photolithography (PHOTO), etching (ETCH), chemical-mechanical planarization (CMP), cleaning (CLN), chemical vapor deposition (CVD), ion implantation (IMP), diffusion (DIFF), and metal interconnect (METAL). In a fab, several types of products are produced such as DRAM, SSD.

Due to their structures such as numerous product types, demand fluctuations, and hundreds of processing steps (Chen, 2010), the scheduling problem of the semiconductor wafer fabrication is well known NP-hard problem (Garey and Johnson, 1979). To resolve the scheduling problem, many dispatching rules such as shortest processing first (SPT), critical ratio (CR) and starvation avoidance (SA), and genetic algorithm (GA) based approaches are extensively used (Conway and Maxwell, 1962; Glassey and Resende, 1988; Harrath et al., 2002; Chien and Chen, 2007; Mahmudy et al., 2013). However, it is difficult to use these scheduling algorithms for real-life semiconductor wafer fabrication because of their dynamic features such as change of product demand, unpredictable machine breakdown, and shifting bottleneck. Depending on the type of products, wafer fabrications are carried out along the processes specified by the corresponding product recipes.

In Chapter 3, the adjustable dispatching rule (ADR) shows a successful scheduling result in semiconductor manufacturing processes with multiple types of jobs by calculating weighted sum of control factors for determining priorities of waiting lots. In particular, the ADR facilitates to dynamically reassign priorities of waiting lots in process by determining weights of control factors according to the conditions of fabrication processes.

In this chapter, we propose a novel variable ranking algorithm, kernelized general dominance weight (KGDW), to improve the scheduling performance with the ADR. The proposed model employs the relevance vector machine (RVM) regression model (Tipping, 2001) for the nonlinearity between inputs and output variables in the framework of the general dominance weight method (Budescu, 1993; Azen and Budescu, 2003,Kim et al., 2017) for the importance assessment of the inputs. With the proposed variable ranking model, the importance of control factors about wafer fabrication processes are scored in relative scales, and the resultant important scores are employed for the weights of control factors for scheduling with the ADR. By taking the importance scores as the weight parameters in the ADR scheduling.

This chapter is organized as follows. In Section 4.2, the RVM regression technique and the traditional variable ranking algorithms are briefly described. Then, we propose the kernelized GDW in Section 4.3. The performance of our proposed algorithm is evaluated with existing variable ranking algorithms using simulation results in Section 4.4. Section 4.5 finally presents the conclusions.

4.2 Background

4.2.1 Variable Ranking Algorithms

The key method for variable ranking methods is measuring the relative contribution of the predictor variables (X_i) in dependent variables (Y_i). In order to measure the variables importance, there are some variable ranking algorithms that are depending on regression techniques can do that.

Chao, et al. (2008) presented six variable ranking methods that depending on multiple linear regression. They are used for health studies. Gazzola, et al. (2018) presented a new technique that estimates the importance for each input variable within multiple steps process. Zhao, et al. (2017) extended the relative importance method from linear regression to additive models: a semi-parametric model type. A linear regression model implicitly assumes that the advertising channels contribute to revenue in a linear manner. As it is generally a restrictive assumption to expect the underlying data generation process of any real dataset to be linear, the resulting attribution values are very likely to be inaccurate. Nonlinear parametric models, such as logistic regression, might be viable alternatives. Nonetheless, it is still difficult to justify postulating any pre-assumed model structure. In other words, a good model should allow for far greater flexibility in unfolding the underlying relationship. A non-parametric component provides flexibility. As such, semi-parametric approaches have the potential to be an important auxiliary modeling paradigm for indirect attribution approaches(Zhao, et al. 2017).

4.2.1.1 Variable Importance in Projection (VIP)

VIP determines the importance percentage of independent variables depending on weights that contributes in value of Y by using Partial Least Squares (PLS) on (X, y) (Wold, et al. 2001).

Partial Least Squares Regression (PLS) is a recent method that creates and integrates features from primary structure and linear regression. This technique aims to estimate a group of dependent variables from of regressors or predictors. PLS uses latent variables to extract the prediction of dependent variables from the regressors by using orthogonal factors. It is preferred to use when we need to predict dependent variables in case there are a huge number of predictors variables. Partial Least Squares tries to discover the covariance between independent variables and dependent variables by finding latent variables. Original predictor variables can be explained as a linear combination of latent variables. A good prediction performance depends on a selection the number of latent variables (Hwang, et al. 2014).

4.2.1.2 Variable Permutation

This method studies how unexpected permutations of X_1 affect the residual sum of squares of a regression model, while conditioning on the values of $X_2, X_3, \ldots, X_p$ in order isolate the relative importance of X_1 from that of all other input variables. Its procedure consists of two loops and may be summarized by the following some steps(Strobl et al. 2008).

4.2.1.3 R-relief

Relief algorithm is successful and general predictor estimators. It can recognize conditional dependencies between predictors and dependent variables in regression and classification. In addition, their quality estimates have a natural interpretation. R-RELIEF relies on a particular, intuitively meaningful idea of what makes an input variable important. According to this idea, input variable X_1 is important if it satisfies two conditions: I) it separates observations with different output value; II) it does not separate observations with similar output value (Šikonja and Kononenko 2003).

4.2.1.4 Relative Weights

The Relative Weights method divides the coefficient of determination R^2 of multiple linear regression to a group of orthonormal components that be as a result from independent variables. It makes a group of new independent variables which have the relationship with the primary variables and are not correlate to each other. The respond variables can be predicted from a new group of predictors variables that produces regression coefficients.

4.2.1.5 General Dominance Weights

This method divides the coefficient of determination R^2 from multiple linear regression by repeated consecutive aggregate of squares on X_1. It defines importance in a unique way by comparing pairs of predictors (from a selected model) across all subset models (subsets of the

predictors) to make the determination of relative importance or dominance. Dominance analysis is an axiomatic and engaging procedure for determining predictor importance that requires only a measure of model fit (e.g., R^2) to determine the additional contribution of any given predictor to any specific subset model. In linear regression, whether model fit is defined as the ratio of variance in the response accounted for by the predictors or the squared correlation between the observed and the predicted responses, the same R^2 measure is obtained (Azen and Budescu 2003).

4.2.2 Relevance Vector Machine (RVM) Regression

The RVM proposed by Tipping (2001) is a kernel-based machine learning algorithm that uses Bayesian inference to get sparse solutions for classification and regression problems. It can be used as an alternative to the support vector machine (SVM). Compared to the SVM, the sparsity of solutions is the most recognizable advantage of the RVM. It means the number of support vectors of the SVM is bigger than the number of relevance vectors of RVM. So, the RVM is significantly more than SVM and the posterior distributions of most weights in the RVM are rapidly peaked around zero (Samui et al., 2011; Hwang et al., 2014; Hwang and Jeong, 2018).

In supervised learning, there are training observations$\{\mathbf{x}_i, y_i\}_{i=1}^{N}$. The RVM regression technique may be explained by the form:

$$y = f(\mathbf{x}) + \epsilon, \tag{4.1}$$

where $\epsilon \sim N(0, \beta^{-1})$ and the first term of (1) can be denoted by $f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})$ with a weight vector $\mathbf{w}$ and a vector of basis functions $\boldsymbol{\phi}(\mathbf{x}) = [\phi_1(\mathbf{x}), \ldots, \phi_N(\mathbf{x})]^{\mathrm{T}}$.

The conditional probability of a weight vector $\mathbf{w}$ given the diagonal matrix $\mathbf{A}$ as the hyperparameter can be explained as $p(\mathbf{w}|\mathbf{A}) \sim N(\mathbf{w}|\mathbf{0}, \mathbf{A^{-1}}) = \prod_{i=1}^{N} N(\mathrm{w}_i|0, A_{ii}^{-1})$ because an automatic relevance determination (ARD) prior. Also, a kernel function $K(\mathbf{x}, \mathbf{x}')$ can be used instead of a basis function $\phi_i(\mathbf{x}) = K(\mathbf{x}, \mathbf{x_i})$.

Due to obtain the noise precision β and the prior precision $\mathbf{A}$, the maximum likelihood estimation can be applied with the likelihood function as follows:

$$p(\mathbf{y}|\mathbf{X}, \mathbf{A}, \beta) = \int p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \beta) p(\mathbf{w}|\mathbf{A}) d\mathbf{w}$$

$$\sim N(\mathbf{0}, \boldsymbol{\phi}\mathbf{A^{-1}}\boldsymbol{\phi}^{\mathbf{T}} + \beta^{-1}\mathbf{I}) \qquad (4.2)$$

where $\boldsymbol{\phi}$ is a M x M matrix and i-th row is $\phi_i(\mathbf{x})^{\mathrm{T}}$ (Son and Lee, 2016).

4.3 Kernelized General Dominance Weight

We present the kernelized GDW technique to rank the importance of each control factor to improve the system performance of semiconductor wafer fabrication. The GDW is proposed by Budescu (1993) and refined by Azen and Budescu (2003) to obtain variable importance using variance contributions of all pairs of variables.

The kernelized GDW is based on the framework of the GDW adopting the RVM regression technique for estimating the effect of a variable of interest on the predictive performance in consideration of non-linear relationships between the inputs and output. For the simplicity of the following description, the RVM regression model can be written as

$$y = f(\mathbf{x}) + \varepsilon, \tag{4.3}$$

where f is the predictive function of the regression model, and $\mathbf{x}$ is an input feature vector.

This method decomposes the R^2 of model (4.3) via iterative sequential sums of squares on X_j. Its procedure may be summarized by the next steps:

Step 1:

- The procedure of the proposed method for the importance of variable X_j for $j = 1, \dots, p$ is as follows. Let S_t for the indexes $t = 1, \dots, 2^p - 1$ be the subsets of $S = \{X_1, X_2, \dots, X_p\} \setminus \{X_j\}$ and $\mathbf{x}^{(S_t)}$ be the input vector of the variables in S_t.
- For each subset S_t for $t = 1, \dots, 2^p - 1$, repeat the steps 2-4.

Step 2:

- Using the variables in S_t, train the model in (4.3) with the training dataset and compute the pseudo coefficient of determination with the testing dataset, $R^2_{S_t}$, as

$$R_{S_t}^2 = 1 - \frac{SSE_{S_t}}{SST}$$

- $SSE_{S_t} = \sum_i \left(y_i - f\left(\mathbf{x}_i^{(S_t)}\right)\right)^2$ is sum of squared errors with the testing dataset
- $SST = \sum_i (y_i - \bar{y})^2$ is sum of squared total with the testing dataset

Step 3:

- Using the variables in $S_t' = S_t \cup \{X_j\}$, train the model in (4.3) with the training dataset and compute the pseudo coefficient of determination with the testing dataset $R_{S_t'}^2$.

$$R_{S_t \cup \{X_j\}}^2 = 1 - \frac{SSE_{S_t'}}{SST}$$

- $SSE_{S_t'} = \sum_i \left(y_i - f\left(\mathbf{x}_i^{(S_t')}\right)\right)^2$ is sum of squared errors with the testing dataset

Step 4:

- Compute the difference of the testing performance between the models using the subsets with X_j and without X_j as

$$\Delta_{S_t} = R_{S_t'}^2 - R_{S_t}^2. \quad (4.4)$$

In (4.4), we employ the pseudo coefficient of determination with the testing dataset, $R_{S_t}^2$ of the models instead of the coefficient of determination, R^2, used in the original GDW, wherein R^2 can be limited to precisely quantify the importance of variables in terms of prediction with nonlinear

predictive models. Although the values of Δ_{S_t} are expected to increase from zero as much as the focal variable improves the model performance, (4.4) can result in negative values in cases of low importance. That is, the model performance can be degenerated by taking the focal variable as model overfitting or increase of noise (unnecessary inputs).

Step 5:

- Calculate the variable importance of X_j averaging out the difference of the testing performance Δ_{S_t} with weights over all the subsets as

$$RI_j = \frac{1}{p}\sum_{k=0}^{p-1}\left(\frac{1}{\binom{p-1}{k}}\sum_{t:|S_t|=k}\Delta_{S_t}\right) \tag{4.5}$$

A first "inner" average (within parentheses) considers all possible RVM regression models on a given number of input variables. To be specific, the term within parenthesis contains a sum that runs over all subsets S_t whose cardinality (number of input variables) is equal to k; there are $\binom{p-1}{k}$ such subsets, and so they represents the increase in pseudo coefficient of determination yielded by the addition of X_1 to the input variables of a RVM regression model, averaged out over all RVM regression models on k input variables (Grömping, 2009). An outer average (outside the parentheses) averages out all inner averages across all possible numbers of input variables. The sum outside the parentheses runs over all possible k values, from 0 (no input variable) to $p-1$ (all input variables other than X_1). Since there are a total of p such values, the ratio $\frac{1}{p}$ normalizes the sum to an average.

4.4 Simulation Studies

4.4.1 Simulation Model Description

In this section, the simulation model developed using Arena software is briefly introduced for getting observations to evaluate performance of the kernelized GDW algorithm. The simulation model generates reports immediately when a simulation run is done (Garrido, 2009). Due to its features such as supporting hierarchical architecture and object-oriented programming, the Arena is a powerful modeling simulation tool for diverse areas including manufacturing, logistics, transportation and data communication (Hammann and Markovitch, 1995).

Based on sample data, processing steps obtained from a semiconductor manufacturing company

Then, the precedence constraints between two modules can be found using also sample data of processing steps.

As shown in Figure 4.1, the eight modules and moving sequences between two modules are depicted as rounded rectangles and arrows, respectively. Using the simulation model is developed with eight modules: PHOTO, ETCH, CMP, CLN, CVD, IMP, DIFF and METAL.

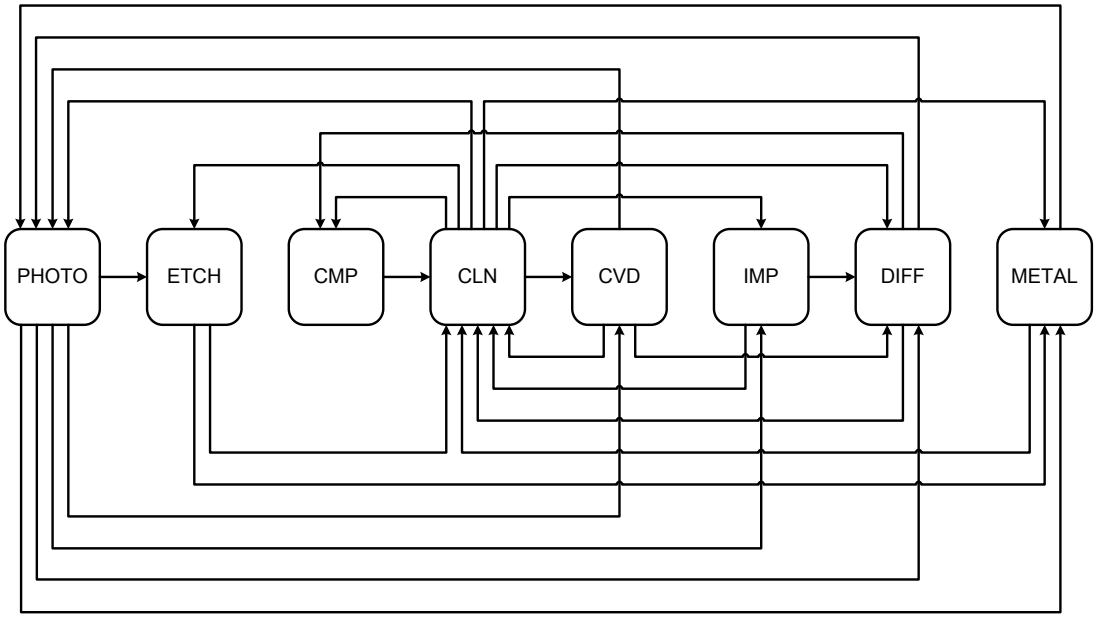


Figure 4.1 Semiconductor wafer fab modules and moving sequences

As shown in Table 4.1, the number of identical machines for each module are carefully determined to achieve a reasonable load balance of the simulation model. For the transportation between two different modules, we also assume that all distances between two different modules are equal to 100 feet, and 10 automated guided vehicles (AGVs) are randomly located.

Table 4.1 The number of machines in each module

Module	Number of Machines	Module	Number of Machines
PHOTO	10	ETCH	4
IMP	3	METAL	4
CLN	12	CVD	3
DIFF	15	CMP	2

For the entities of the simulation model, i.e., wafer lots, there are three types of products P_1, P_2 and P_3. The proportion of each product type is assigned 50% of P_1, 30% of P_2, and 20% of P_3. It means that if the average inter-arrival time of all products is 60 min, the average inter-arrival time of P_1, P_2 and P_3 is 120 min, 200 min, and 300 min, respectively.

As shown in Table 4.2, the sample data of processing steps obtained from a semiconductor manufacturing company consists of 20 layers and 94 processing steps with pairs of the name of module and ideal processing time. The ideal processing time is the processing time for a given step without setup time.

Table 4.2 The sample data of the processing steps

Layer ID	No of Steps	Processing Steps (Module name, Ideal processing time (min))
1	7	(IMP,30) → (IMP,40) → (DIFF,300) → (PHOTO,60) → (DIFF,250) → (CLN,35) → (CLN,25)
2	4	(PHOTO,55) → (ETCH,40) → (CLN,25) → (CLN,30)
3	3	(PHOTO,62) → (IMP,45) → (CLN,25)
4	5	(PHOTO,62) → (METAL,50) → (CLN,25) → (CVD,30) → (DIFF,300)
5	3	(PHOTO,62) → (IMP,50) → (IMP,45)
6	4	(PHOTO,60) → (ETCH,50) → (CLN,35) → (CVD,35)
7	6	(PHOTO,55) → (ETCH,40) → (CLN,25) → (CLN,30) → (CMP,60) → (CLN,30)
8	4	(PHOTO,62) → (METAL,55) → (CLN,25) → (ETCH,40)
9	4	(PHOTO,62) → (ETCH,30) → (CLN,25) → (CVD,40)
10	6	(PHOTO,60) → (DIFF,200) → (DIFF,180) → (CLN,35) → (CLN,25) → (METAL,55)
11	4	(PHOTO,60) → (CVD,30) → (CLN,35) → (CLN,25)
12	8	(PHOTO,55) → (ETCH,40) → (ETCH,45) → (CLN,25) → (CLN,30) → (DIFF,250) → (CMP,55) → (CLN,30)
13	4	(PHOTO,62) → (METAL,45) → (CLN,25) → (METAL,50)
14	4	(PHOTO,60) → (DIFF,400) → (CLN,35) → (CLN,25)
15	6	(PHOTO,62) → (ETCH,35) → (METAL,45) → (CLN,25) → (CVD,35) → (CVD,40)
16	4	(PHOTO,60) → (CVD,30) → (CLN,35) → (CLN,25)
17	3	(DIFF,200) → (CLN,35) → (CLN,25)
18	6	(PHOTO,55) → (ETCH,40) → (CLN,25) → (CLN,30) → (CMP,60) → (CLN,30)
19	3	(PHOTO,62) → (METAL,50) → (CLN,25)
20	5	(PHOTO,55) → (ETCH,50) → (CLN,25) → (CLN,30) → (DIFF,400)

In order to assign the processing steps for P_1, layer ID are chosen from 1 to 20 in numerical order. Then, the processing steps for P_2 and P_3 are assigned by shuffling the sequence of P_1 as shown in Table 4.3.

Table 4.3 Sequence of layer IDs for each product type

Product Type	Sequence of Layer IDs
$\mathbf{P_1}$	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20
$\mathbf{P_2}$	13,5,14,6,4,19,11,9,8,15,3,7,16,1,18,12,17,2,20,10
$\mathbf{P_3}$	14,11,7,15,2,12,9,18,16,20,5,3,8,1,6,19,17,4,10,13

4.4.2 Experimental Setup

In order to get training observations for the kernelized GDW algorithm, the description of the experimental setup is as follows. Each simulation run is executed 30 days of simulation period with 10 days of warm-up to prevent startup bias. First, 100 different set of weight values for 10 control factors, i.e. recipe change loss ($\boldsymbol{F_1}$), chamber availability ($\boldsymbol{F_2}$), chuck efficiency ($\boldsymbol{F_3}$), device priority ($\boldsymbol{F_4}$), moving target ($\boldsymbol{F_5}$), lot location ($\boldsymbol{F_6}$), delay time ($\boldsymbol{F_7}$), wait time ($\boldsymbol{F_8}$), lot priority ($\boldsymbol{F_9}$), and designate step factor ($\boldsymbol{F_{10}}$), are randomly generated as input variables. The more detailed description for control factors, i.e. the purpose of each control factor and how to calculate the value of control factors, can be found in Chapter 3. The output variable, average loss, is obtained from the result of each simulation run. The average loss means the difference average of time between actual processing time and ideal processing time for all lots processed in the simulation period as shown in Figure 4.2.

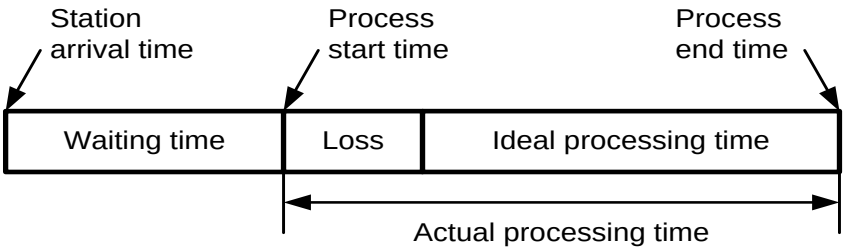


Figure 4.2 The Loss time for all processed lots

4.4.3 Experimental Results

In this experiment, we focus on the two control factors that are related with the efficiency of the machines in PHOTO module. First, the recipe change loss factor indicates a loss of efficiency cause by various amount of setup time for the change of recipe that requires when both previous and current processes have different recipe on a given machine. Calculating the recipe change loss factor value for a given lot on a given machine can be calculate as the ratio of (maximum loss – current loss) to the maximum loss. Second, the chuck efficiency factor indicates a loss of efficiency cause by the setup time for moving chuck arms. Each lot should be processed using either an odd chuck arm or an even chuck arm. When previous and current lots should be processed using same chuck arm, a fixed amount of setup time for moving chuck arms is required.

The advantages of this method are that processing priorities of waiting lots could be reassigned dynamically by determining weight values for each control factors at the right moment according to the conditions of semiconductor wafer fabrication. To develop and improve the scheduling performance with ADR, it is useful to determine the weight values for each control factors.

Variable ranking with relative importance (RI) scores of the kernelized GDW is evaluated with the existing variable ranking methods such as general dominance weight (GDW), relative weights (RW), variable importance in projection (VIP), variable permutation (VP), tree method (TM) and r-relief method (RM) as shown in Table 4.4.

Table 4.4 Relative importance (RI) scores and ranking of control factors

	KGDW		GDW		VIP		TM		VP		RM		RW	
	RI	**Rank**	**RI**	**Rank**	**RI**	**Rank**	**RI**	**Rank**	**RI**	**Rank**	**RI**	**Rank**	**RI**	**Rank**
F_1	0.435	1	0.345	1	0.096	8	0.233	1	0.270	1	0.225	2	0.105	4
F_2	0.022	9	0.034	8	0.095	9	0.114	4	0.055	8	0.023	9	0.107	3
F_3	0.270	2	0.218	2	0.093	10	0.127	2	0.182	2	0.040	7	0.089	8
F_4	0.022	8	0.020	9	0.102	3	0.057	9	0.000	10	0.078	5	0.096	6
F_5	0.008	10	0.015	10	0.104	2	0.076	7	0.062	7	0.042	6	0.120	1
F_6	0.061	3	0.075	5	0.101	5	0.028	10	0.107	4	0.145	4	0.088	10
F_7	0.038	7	0.077	4	0.106	1	0.121	3	0.109	3	0.039	8	0.102	5
F_8	0.044	5	0.063	6	0.099	6	0.088	5	0.079	6	0.232	1	0.110	2
F_9	0.040	6	0.091	3	0.102	4	0.084	6	0.046	9	0.000	10	0.089	7
F_{10}	0.055	4	0.059	7	0.099	7	0.068	8	0.086	5	0.172	3	0.088	9

Desirable ranking of weight factors from field experiences are as follows: F_1 is the most important one because it is related with the recipe change loss. The recipe change loss can be increased to maximum of 20 min according to the recipe for the lot, and the relationship between the machine and the lot. F_3 is the second important one because it is related with the chuck loss, but relatively less important than F_1 because chuck loss can be increased to maximum of 5 min (i.e., it is significantly less than recipe change loss). F_2 should be the least one because it is related with the

chamber availability but, in the PHOTO station, all lots have values of F_2 are equal to 0. F_4, F_6, F_9 and F_{10} are the second least important group with the same level. Other remaining factorsF_5, F_7 and F_8 are not important either, but a little better than the above four factors. These remaining factors may have similar importance values.

According to the field experiences, we expect that F_1 and F_3 should be rank 1 and rank 2, respectively. The results show that KGDW, GDW, TM and VP may be good candidates of variable ranking method for our simulation model. However, we also expect that the RI scores of F_1 and F_3 should be significantly larger than others, and the RI score of F_1 should be larger than the RI score of F_3. In order to verify which variable ranking method is the best from the candidates, two simple arithmetic calculations are performed as shown in Table 4.5. The results show that KGDW is the best variable ranking method of the candidates.

Table 4.5 Arithmetic calculations of relative importance (RI) scores for $\boldsymbol{F_1}$ and $\boldsymbol{F_3}$

	KGDW	**GDW**	**TM**	**VP**
$\boldsymbol{F_1 + F_3}$	0.7054	0.5633	0.3606	0.4528
$\boldsymbol{F_1 - F_3}$	0.165	0.127	0.106	0.088

4.5 Conclusion

In this chapter, a new variable ranking algorithm called the kernelized general dominance weight (GDW) is proposed to improve the scheduling performance by using relevance vector machine (RVM) regression technique for semiconductor wafer fabrication. The relative importance (RI) scores of each control factor and each system performance are calculated by using kernelized GDW. The RVM regression technique is applied to build kernel version of GDW, and to get the best fit model. To assess the achievement of presented algorithm, the simulation model is developed and the observations are generated by running of the simulation model. Experimental results show that relative importance (RI) scores and variable ranking of the kernelized GDW are reasonable than the results of existing variable ranking algorithms.

CHAPTER 5 Conclusion Remarks

In this dissertation, we have proposed and developed several methodologies for finding best scheduling of different manufacturing systems such as traditional manufacturing systems and semiconductor manufacturing systems. In chapter 2, a dynamic job shop scheduling algorithm based on regression technique is proposed to overcome on drawbacks in the existing dynamic job shop scheduling based on classification approaches. Kernel ridge regression is presented because it has ability to discover the unknown relations between predictor variables and response variables.

The experimental results are promising and strongly present that the proposed approach in this project can be applied to give the best values for the performance measures.

In Chapter 3, the efficient algorithm that consists of the adjustable dispatching rule (ADR) and real coded genetic algorithm (RCGA) with fitness approximation has been proposed. Kernel ridge regression and polynomial regression techniques as fitness function and computer simulation for scheduling problem of semiconductor manufacturing system has been presented. RCGA generates a new population and then it works to improve it by repeating the main operations such as selection, crossover and mutation operators. By the empirical results state that the ADR and the real coded GA with fitness approximation can find near-optimal solution of the scheduling problem for semiconductor wafer fabrication with relatively small number of fitness evaluation. It can play an important role when the cost of evaluating fitness is expensive.

In chapter 4, a new ranking algorithm called kernelized general dominance weight (KGDW) is proposed to determine the importance of each weight factors for improving scheduling performance of semiconductor wafer fabrication. In order to get high quality of prediction model,

we have modified relevance vector machine (RVM) regression to work with GDW. To assess the achievement of presented algorithm, the simulation model is developed and the observations are generated by running of the simulation model. Experimental results show that relative importance (RI) scores and variable ranking of the kernelized GDW are reasonable than the results of existing variable ranking algorithms.

References

Azen, R., and Budescu, D. V. (2003). The dominance analysis approach for comparing predictors in multiple regression. *Psychological Methods*, 8(2), 129-148.

Baker, K. R. (1984). Sequencing rules and due-date assignments in a job shop. *Management Science*, 30(9), 1093-1104.

Budescu, D. V. (1993). Dominance analysis: A new approach to the problem of relative importance of predictors in multiple regression. *Psychological Bulletin*, 114(3), 542-551.

Chao, E. Y., Zhao, Y., Kupper, L. L. and Nylander-French, L. A. (2008). Quantifying the relative importance of predictors in multiple regression analyses for public health studies. *Journal of Occupational and Environmental Hygiene*, 5(8), 519-529.

Chen, T. (2010). An optimized tailored nonlinear fluctuation smoothing rule for scheduling a semiconductor manufacturing factory. *Computers and Industrial Engineering*, 58(2), 317-325.

Chien, C. F., and Chen, C. H. (2007). A novel timetabling algorithm for a furnace process for semiconductor fabrication with constrained waiting and frequency-based setups. *OR Spectrum*, 29(3), 391-419.

Cho, H. W., Kim, K. J., and Jeong, M. K. (2008). Determination of influential factors and diagnostics using multivariate statistical relationships between variables and faults. *Expert Systems with Applications*, 35(1), 30-40.

Conway, R. W., and Maxwell, W. L. (1962). Network dispatching by the shortest-operation discipline. *Operations Research*, 10(1), 51-73.

Dabbas, R. M., and Fowler, J. W. (2003). A new scheduling approach using combined dispatching criteria in wafer fabs. *IEEE Transactions on Semiconductor Manufacturing*, 16(3), 501-510.

Exterkate, P., Groenen, P. J., Heij, C., and van Dijk, D. (2016). Nonlinear forecasting with many predictors using kernel ridge regression. *International Journal of Forecasting*, 32(3), 736-753.

Fenner, J.S., Jeong, M.K., and Lu, J.C. (2005). Optimal automatic control of multistage production processes. *IEEE Transactions on Semiconductor Manufacturing*, 18(1), 94–103.

Garey, M. R., and Johnson, D. S. (1979). A guide to the theory of NP-completeness. A Series of Books in the Mathematical Sciences. *W. H. Freeman and Company*, San Francisco.

Garrido, J. M. (2009). Object oriented simulation: A modeling and programming perspective. *Springer*, Boston.

Gazzola. G., Choi, J., Kwak, D., Kim, B., Kim, D., Tong, S., and Jeong, M. K. (2018). Integrated variable importance assessment in multi-stage production processes. *IEEE Transactions on Semiconductor Manufacturing*, 31(3), 343-355.

Glassey, C. R., and Resende, M. G. (1988). Closed-loop job release control for VLSI circuit manufacturing. *IEEE Transactions on Semiconductor Manufacturing*, 1(1), 36-46.

Goldberg, D. E. (1989). Genetic algorithms in search, optimization, and machine learning. *Addison-Wesley*, Boston.

Grömping, U. (2009). Variable importance assessment in regression: linear regression versus random forest. *The American Statistician*, 63(4), 308-319.

Guh, R. S., Shiue, Y. R., and Tseng, T. Y. (2011). The study of real time scheduling by an intelligent multi-controller approach. *International Journal of Production Research*, 49(10), 2977-2997.

Hammann, J. E., and Markovitch, N. A. (1995). Introduction to arena [simulation software]. *Simulation Conference Proceedings*, pp. 519-523.

Harrath, Y., Chebel-Morello, B., and Zerhouni, N. (2002). A genetic algorithm and data mining based meta-heuristic for job shop scheduling problem. *IEEE Symposium on Emerging Technologies and Factory Automation* 2(1), 727-728.

Helland I.S. (1990). PLS regression and statistical models. *Scandinavian Journal of Statistics*, 17(2), 97–114.

Herrera, F., Lozano, M., and Sánchez, A. M. (2003). A taxonomy for the crossover operator for real-coded genetic algorithms: An experimental study. *International Journal of Intelligent Systems*, 18(3), 309-338.

Holthaus, O., and Rajendran, C. (1997). Efficient dispatching rules for scheduling in a job shop. *International Journal of Production Economics*, 48(1), 87-105.

Hwang, S., Jeong, M. K., and Yum, B. J. (2014). Robust relevance vector machine with variational inference for improving virtual metrology accuracy. *IEEE Transactions on Semiconductor Manufacturing*, 27(1), 83-94.

Hwang, S., and Jeong, M. K. (2018). Robust relevance vector machine for classification with variational inference. *Annals of Operations Research*, 263(1), 21-43

Jeong, M. K., Mat, P., and Chen, Z. (2005). Throughput gain with parallel flow in automated flow lines. *IEEE Transactions on Automation Science and Engineering*, 2(1), 84-86.

Kim, J., Jeong, M. K. and Elsayed, E. A. (2017). Monitoring multistage processes with autocorrelated observations. *International Journal of Production Research*, 55(8), 2385-2396.

Li, X., and Olafsson, S. (2005). Discovering dispatching rules using data mining. *Journal of Scheduling*, 8(6), 515-527.

Mahmudy, W., Marian, R. M., and Luong, L. H. (2013). Modeling and optimization of part type selection and loading problem in flexible manufacturing system using real coded genetic algorithms. *International Journal of Electrical, Computer, Electronics and Communication Engineering*, 7(4), 251-260.

Metan, G., and Sabuncuoglu, I. (2005). A simulation based learning mechanism for scheduling systems with continuous control and update structure. *IEEE of the 37th Conference on Simulation*, pp. 2148-2156.

Metan, G., Sabuncuoglu, I., and Pierreval, H. (2010). Real time selection of scheduling rules and knowledge extraction via dynamically controlled data mining. *International Journal of Production Research*, 48(23), 6909-6938.

Michalewicz, Z. (1996). Genetic algorithms + data structures = evolution programs. *Springer*, New York.

Min, H. S., and Yih, Y. (2003). Selection of dispatching rules on multiple dispatching decision points in real-time scheduling of a semiconductor wafer fabrication system. *International Journal of Production Research*, 41(16), 3921-3941.

Mönch, L., Fowler, J. W., Dauzère-Pérès, S., Mason, S. J., and Rose, O. (2011). A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations. *Journal of Scheduling*, 14(6), 583-599.

Montazeri, M., and Van Wassenhove, L. N. (1990). Analysis of scheduling rules for an FMS. *International Journal of Production Research*, 28(4), 785-802.

Murugan, M. S., Suresh, S., Ganguli, R., and Mani, V. (2007). Target vector optimization of composite box beam using real-coded genetic algorithm: a decomposition approach. *Structural and Multidisciplinary Optimization*, 33(2), 131-146.

Olafsson, S., and Li, X. (2010). Learning effective new single machine dispatching rules from optimal scheduling data. *International Journal of Production Economics*, 128(1), 118-126.

Pinedo, M. L. (2016). Scheduling: theory, algorithms, and systems. *Springer*, New York.

Priore, P., Gómez, A., Pino, R., and Rosillo, R. (2014). Dynamic scheduling of manufacturing systems using machine learning: An updated review. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 28(1), 83-97.

Quinlan, J. R. (1993). C4.5: Programs for machine learning. *Morgan Kaufmann*, San Francisco.

Renders, J. M., and Flasse, S. P. (1996). Hybrid methods using genetic algorithms for global optimization. *IEEE Transactions on Systems, Man, and Cybernetics*, 26(2), 243-258.

Samui, P., Lansivaara, T., and Kim, D. (2011). Utilization relevance vector machine for slope reliability analysis. *Applied Soft Computing*, 11(5), 4036-4040.

Shakhlevich, N., Hoogeveen, H., and Pinedo, M. (1998). Minimizing total weighted completion time in a proportionate flow shop. *Journal of Scheduling*, 1(3), 157-168.

Shaw, M. J., Park, S., and Raman, N. (1992). Intelligent scheduling with machine learning capabilities: the induction of scheduling knowledge. *IIE Transactions*, 24(2), 156-168.

Shiue, Y. R. (2009). Data-mining-based dynamic dispatching rule selection mechanism for shop floor control systems using a support vector machine approach. *International Journal of Production Research*, 47(13), 3669-3690.

Šikonja, M., and Kononenko, I. (2003). Theoretical and empirical analysis of reliefF and rreliefF. *Machine Learning*, 53 (1), 23–69.

Sivakumar, A. I., and Gupta, A. K. (2006). Online multi-objective pareto optimal dynamic scheduling of semiconductor back-end using conjunctive simulated scheduling. *IEEE Transactions on Electronics Packaging Manufacturing*, 29(2), 99-109.

Son, Y., and Lee, J. (2016). Active learning using transductive sparse bayesian regression. *Information Sciences*, 374(1), 240-254.

Sotskov, Y. N., and Shakhlevich, N. V. (1995). NP-hardness of shop-scheduling problems with three jobs. *Discrete Applied Mathematics*, 59(3), 237-266.

Strobl, C., Boulesteix, A., Kneib, T., Augustin, T., and Zeileis, A. (2008). Conditional variable importance for random forests. *BMC Bioinformatics*, 9(1), 307.

Su, C. T., and Shiue, Y. R. (2003). Intelligent scheduling controller for shop floor control systems: A hybrid genetic algorithm/decision tree learning approach. *International Journal of Production Research*, 41(12), 2619-2641.

Thiesse, F., and Fleisch, E. (2008). On the value of location information to lot scheduling in complex manufacturing processes. *International Journal of Production Economics*, 112(2), 532-547.

Tipping, M. E. (2001). Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1(1), 211-244.

Tyan, J. C., Du, T. C., Chen, J. C., and Chang, I. H. (2004). Multiple response optimization in a fully automated FAB: An integrated tool and vehicle dispatching strategy. *Computers and Industrial Engineering*, 46(1), 121-139.

Uzsoy, R., Church, L. K., Ovacik, I. M., and Hinchman, J. (1992). Dispatching rules for semiconductor testing operations: A computational study. *In Electronics Manufacturing Technology Symposium, Thirteenth IEEE/CHMT International*, pp. 272-276.

Wold, S.; Sjöström, M.; Eriksson, L. (2001) PLS-regression: a basic tool of chemometrics. *Chemometrics and Intelligent Laboratory Systems*, 58(2), 109-130.

Yugma, C., Blue, J., Dauzère-Pérès, S., and Obeid, A. (2015). Integration of scheduling and advanced process control in semiconductor manufacturing: Review and outlook. *Journal of Scheduling*, 18(2), 195-205.

Zhang, H., Jiang, Z., and Guo, C. (2009). An optimised dynamic bottleneck dispatching policy for semiconductor wafer fabrication. *International Journal of Production Research*, 47(12), 3333-3343.

Zhao, K., Mahboobi, S., & Bagheri, S. (2017). Revenue-based attribution modeling for online advertising. *IEEE Transactions on Knowledge and Data Engineering*, 1(1), 1-16.