Mathematical-based Approaches for the Semiconductor Capital Equipment Installation and Qualification Scheduling Problem

by

Junzilan Cheng

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved May 2015 by the
Graduate Supervisory Committee:

John Fowler, Co-Chair
Karl Kempf, Co-Chair
Scott Mason
Muhong Zhang

ARIZONA STATE UNIVERSITY

May 2015

ABSTRACT

Ramping up a semiconductor wafer fabrication facility is a challenging endeavor. One of the key components of this process is to schedule a large number of activities in installing and qualifying (Install/Qual) the capital intensive and sophisticated manufacturing equipment. Activities in the Install/Qual process share multiple types of expensive and scare resources and each activity might potentially have multiple processing options. In this dissertation, the semiconductor capital equipment Install/Qual scheduling problem is modeled as a multi-mode resource-constrained project scheduling problem (MRCPSP) with multiple special extensions. Three phases of research are carried out: the first phase studies the special problem characteristics of the Install/Qual process, including multiple activity processing options, time-varying resource availability levels, resource vacations, and activity splitting that does not allow preemption. A modified precedence tree-based branch-and-bound algorithm is proposed to solve small size academic problem instances to optimality. Heuristic-based methodologies are the main focus of phase 2. Modified priority rule-based simple heuristics and a modified random key-based genetic algorithm (RKGA) are proposed to search for Install/Qual schedules with short makespans but subject to resource constraints. Methodologies are tested on both small and large random academic problem instances and instances that are similar to the actual Install/Qual process of a major semiconductor manufacturer. In phase 3, a decision making framework is proposed to strategically plan

the Install/Qual capacity ramp. Product market demand, product market price, resource consumption cost, as well as the payment of capital equipment, are considered. A modified simulated annealing (SA) algorithm-based optimization module is integrated with a Monte Carlo simulation-based simulation module to search for good capacity ramping strategies under uncertain market information. The decision making framework can be used during the Install/Qual schedule planning phase as well as the Install/Qual schedule execution phase when there is a portion of equipment that has already been installed or qualified. Computational experiments demonstrate the effectiveness of the decision making framework.

I dedicate this work to my wife Mingjun Xia who supported and encouraged me in completing this work.

ACKNOWLEDGMENTS

First of all, I would like to express my sincere gratitude and respect to my Ph.D advisors, Dr. John Fowler and Dr. Karl Kempf. My Ph.D research work would not have been possible without your consistent support and invaluable guidance along the way. Thank you for being such excellent professional and personal models to me. I also greatly appreciate my committee members, Dr. Scott Mason, Dr. Muhong Zhang. Your guidance, suggestions and encouragement helped me finish this dissertation.

Additionally, I am thankful to teachers and friends at Arizona State University. Without you, my study and life experience at Tempe, Arizona would not have been that happy and fruitful. Just to name a few: Dr. Ronald Askin, Dr. Pitu Mirchandani, Dr. Esma Gel, Dr. Dan Shunk, Dr. Teresa Wu, Dr. Rong Pan, Dr. Daniel McCarville, Minyao Sun, Xiaotian Zhuang, Dr. Mengqi Hu, Lei Liu, Dr. Ning Wang, Zhuoyang Zhou, Min Zhang, Brint MacMillan.

Also, I would like to thank my golf buddies: Dr. John Fowler, Jerry Huff, Dr. Brian Stone, Ryan Panos, and Dr. Amit Shinde. Our time spent on various breathtaking golf courses and challenging driving ranges was one of the most memorable experiences I had during my study at Tempe, AZ. I am also thankful to the game of golf which taught me so many things: how important it is to focus on the next shot regardless of all previous shots; never give up on any shots and always enjoy the game!

Most importantly, I would like thank my wife – Mingjun Xia. Thanks for your unconditional love, faith and support along the way since we know each other years ago.

Last but not least, I would like to thank my research team members from Intel Corporation, Erik Hertzler and Daniel Peters, for offering your valuable insights and experience that inspired many ideas in this dissertation. I would also like to thank Intel Corporation for funding most of my Ph.D research.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

Table Page

Table Page

LIST OF FIGURES

Figure Page

Figure Page

CHAPTER 1 INTRODUCTION

1. Introduction and Motivation

The semiconductor manufacturing industry is a capital intensive industry. Nowadays, a state-of-the-art 300mm wafer fabrication (fab) facility with over one thousand pieces of major capital equipment costs at least $3 billion (Chien and Zheng (2012), Chasey and Pindukuri (2012)) and up to $10 billion (Ibrahim, Chik and Hashim, 2014) depending on fab capacity. **Error! Reference source not found.** shows the total annual capital investment from 2011 to 2013 of several semiconductor companies has been more than 45 billion dollars (source: http://www.icinsights.com/ March 26, 2013, Article: Intel and Samsung Forecast to Represent 42% of Semiconductor Capital Spending in 2013).

The capital equipment supply chain is the process of planning, procuring, transporting, installing and qualifying each piece of capital equipment to support production. The objective of capital equipment supply chain planning is to purchase the right amount of production capacity at the right time to reduce the mismatch of capacity and market demand. On one hand, the lack of capacity or bringing the right amount of capacity online at the wrong time can result in hundreds of million dollars of lost sales. On the other hand, excessive capacity means idle capital equipment, each of which can

cost millions of dollars, e.g. currently, a single photolithography stepper costs over 100 million dollars (Thoms, 2012).

Table 1: Top Semiconductor Capital Investment from 2011 to 2013

Company	2011 Actual ($M)	2012 Actual ($M)	2013 Forecast ($M)
Intel	10,764	11,000	13,000
Samsung	11,755	12,225	12,000
TSMC	7,333	8,324	9,000
GlobalFoundries	5,400	3,000	3,500
SK Hynix	3,165	3,655	3,200
Micron	2,913	1,773	2,225
Toshiba	1,935	1,637	1,600
UMC	1,585	1,723	1,500
SanDisk	1,368	988	1,000
Sony	1,805	1,100	775
Total	48,023	45,425	47,800

Within the entire supply chain, the equipment installation and qualification (Install/Qual) process consists of physically installing a piece of equipment (and necessary infrastructure, e.g. pipes for water, gas, wires, etc.) and qualifying the necessary equipment for a specific production requirement. Currently, the Install/Qual process faces several challenges (described below) and has the potential to be significantly improved.

First, the Install/Qual process spans 18 to 24 months and it consumes most of the supply chain lead time. Shortening the Install/Qual process can reduce the supply chain lead time and delay on capital investment planning decisions.

Second, there are over one thousand pieces of major production equipment that need to be installed and qualified. To install and qualify each one of them involves multiple activities, each of which might require multiple resources (labor, testing tool, etc.). The current Install/Qual scheduling approach in practice is mostly based on manual scheduling without a systematic way to search for better schedules that conform to certain resource limits. Thus, potentially better schedules may not be considered.

Third, the Install/Qualification process determines the timing of expenditures since equipment is generally paid for partially when it is received and partly after it is qualified. A better Install/Qual schedule can defer capital payment and reduce overall capital time-value of money.

Fourth, the Install/Qual process determines the capacity ramp-up strategy. A better Install/Qual schedule can potentially bring the right amount of capacity online at the right time to maximize revenue. Further, during schedule execution when there are market information changes, manual re-schedule is the current practice for the Install/Qual process. It is time-consuming and often results in sub-optimal solutions.

The Install/Qual process is extremely important and requires strategic decision making and careful planning. With such intensive capital involved, a small improvement can potentially bring millions of dollars of savings.

The objective of this dissertation is to provide effective scheduling and re-scheduling methodologies to support decision making in the Install/Qual process. This

dissertation research develops mathematical-based analytical modeling for the Install/Qual process and proposes various heuristic optimization methodologies to approach this challenging problem.

The three main phases of research are organized as follows. Phase 1 and phase 2 study different methodologies that search for Install/Qual schedules with short project makespans to reduce supply chain lead time. In Phase 3, ramp-up strategies that consider market price, market demand, resource consumption cost and the timing of capital equipment expenditures are investigated.

In phase 1, the Install/Qual scheduling problem is modeled as a multi-mode resource-constrained project scheduling problem. In this phase, the difference between preemption and activity splitting in the project scheduling literature is discussed. The semiconductor Install/Qual process represents a unique environment where activities can be split but not preempted. Other specialties of the Install/Qual process such as multiple processing options, time-varying resource constraints, and resource vacations are also modeled and discussed. A modified precedence tree-based branch-and-bound algorithm is proposed as an exact method to solve small size academic problem instances to optimality. Experiments demonstrate that allowing activity splitting results in major project makespan reductions compared to preemption. The higher the range of time-varying renewable resource limits and the tighter the renewable resource limits are, the

bigger the resulting makespan reductions can be. The computational complexity of the problem is observed since it still takes over hours to solve some small problem instances.

In phase 2, heuristic-based scheduling algorithms are proposed and studied for the Install/Qual scheduling problem. The first algorithm is based on priority rule-based simple heuristics and the second algorithm modifies the random key-based genetic algorithm (RKGA) to incorporate both mode assignments and relative priorities for activities. The third algorithm is based on ILOG-CPLEX to dynamically search for a good project horizon to reduce computational effort. Project decomposition is also proposed to integrate with meta-heuristics. A decomposition score is defined to measure whether an instance is better to be decomposed or not. Practical constraints of the Install/Qual process such as schedule infeasibility regarding non-renewable resources or time windows (ready time and due date) and backward scheduling approaches are studied and discussed. Computational experiments show that when the decomposition score is low, combining decomposition with other meta-heuristics is recommended. Decomposition works better when availability levels for non-renewable resources are high. Overall, the proposed RKGA outperforms simulated annealing, simple heuristics and modified CPLEX solutions, especially for large size problem instances. Some simple heuristic rules that consider problem characteristics are shown to work well when there are high resource levels. The Phase 3 focuses on how to ramp the right amount of capacity at the right time to maximize overall expected profit which includes revenue

generated by satisfying market demand, costs for consuming resources and the time value of money for capital investment. Uncertain market demand and market price are modeled using Geometric Brownian Motion (GBM) processes. This is true for cases where the final fab capacity is given as input and when it is a decision variable. A scheduling framework is proposed such that the Simulated Annealing algorithm is used as the optimization method to find better solutions along with Monte Carlo simulation as the solution evaluator to deal with uncertainty. Computational experiments show that a good threshold setup between optimization and simulation can achieve a good balance between computational effort and solution quality. Integrating both the optimization and simulation modules can find better solutions than only assuming static market demand and market price. The benefit of adding simulation increases as the demand uncertainty level increases. When demand uncertainty level is low, matching demand and capacity is recommended to achieve high expected profit. However, when the demand uncertainty level is high, over investing capacity is preferable since the cost of losing sales is higher than the cost of idle assets.

Table 2: Potential Research Application Areas of this Dissertation

Industries	Project scheduling problems
Semiconductor	This research
Construction	Brucker et al. (1999), Brucker and Knust (2006), Kim (2007), Pan et al. (2009)
Software Development	Brucker et al. (1999), Wang (2005), Buddhakulsomsiri and Kim (2006), Gonsalves et al. (2008)). Hapke et al. (1998)

Agricultural	Wang et al. (2005)
Steel Manufacturing	Voß and Witt (2007)
Movie shooting	Bomsdorf and Derigs (2008)

This dissertation demonstrates an example of applying mathematical-based approaches to analyze a real world challenging problem in the semiconductor industry. However, the basic methodologies can be adapted to other environments. **Error! Reference source not found.** shows other problem domains that can potentially use the research efforts in this dissertation.

CHAPTER 2 MULTI-MODE RESOURCE-CONSTRAINED PROJECT SCHEDULING PROBLEMS WITH NON-PREEMPTIVE ACTIVITY SPLITTING

1. Introduction

Equipment installation and qualification (Install/Qual) is the process of ramping up a wafer fabrication (fab) facility. During the Install/Qual process, each piece of equipment is first physically installed with necessary infrastructure (e.g. water and gas pipes) by trades (e.g. architects, electricians, mechanics, and plumbers). Supplier qualification is the next step and it consists of running experimental tests by the equipment supplier (supplier resource). Next, the semiconductor company's engineer (company resource) runs product test as the company qualification process before using the equipment for actual wafer production. A shorter Install/Qual process can make the next generation product available to customers sooner and delay capital investment decision-making since overestimation and underestimation can lead to millions of dollars wasted from either idle capital equipment or lost sales.

However, the complexity of the Install/Qual process makes the scheduling problem a challenging endeavor. Activities have precedence relationships such that physical installation is always followed by supplier qualification and company qualification, and support equipment needs to be installed before the production equipment. Availabilities on resources (human resources, project budget, factory floor

space, etc) need to be considered during execution of each activity which could have different options. For example, a piece of equipment can be installed by 3 senior and 1 junior technician with a total cost of $20K in 6 working days or 1 senior and 3 junior technicians with a total cost of $16K in 8 working days. Resources often have different working calendars. For example, trades may work 4 days/week and 10 hours/day while supplier resources may work 5 days/week and 8 hours/day and the company can deploy its own resources 7 days/week and 24 hours/day with 3 shifts. Thus, activities longer than four or five days may need to pause during weekends and resume the following week. However, during weekdays when an activity is processing, it cannot be interrupted. For instance, the equipment qualification process requires a combination of highly specialized personnel and precise tool configurations. Interrupting an ongoing qualification process with another process requires re-mapping these personnel and re-configuring tool settings, which in turn might bring an undesired complexity of progress tracking, significant process risk, and high operational costs.

Motivated by the semiconductor capital equipment Install/Qual scheduling problem, this research studies and extends the well-known resource-constrained project scheduling problem (RCPSP) framework and potentially can be applied to other production planning and scheduling problems (e.g. job shop, open shop) in the semiconductor industry. Examples of challenging semiconductor problems can be found in Mönch *et al.*, 2012.

In RCPSP, activities are represented by nodes and precedence relations are represented by directed arcs. Precedence constraints restrict an activity from starting until all of its predecessors are finished. Processing an activity requires either renewable resources with availability restrictions on each time period (e.g., the number of technicians per day, the number of tools per shift, etc.) and/or non-renewable resources with availability restrictions over the whole project horizon (e.g., project budget, raw materials, factory floor space, etc.). In a multi-mode RCPSP (MRCPSP), each activity can be processed in one of several possible ways, each of which is described by a combination of required resources and activity duration.

In classical RCPSPs and MRCPSPs, renewable resource limits are assumed constant over time. In practice, however, the total amount of resources available might not be constant for a number of reasons. Predictable reasons can be weekends and holidays for labor resources or scheduled maintenance for machines; unpredictable reasons can be personnel taking unexpected sick leave or unscheduled machine breakdowns. In the RCPSP with time-varying resource constraints, the assumption of constant resource limits is relaxed and a resource profile function is used to specify resource availability during each time period (Drexl and Grünewald, 1993, Hartmann, 1999 and Klein, 2000).

Even in the RCPSP with time-varying resource constraints and resource vacations, it is often assumed that activities cannot be split such that activities can only be

scheduled in consecutive time periods within which resources must be constantly available. These restrictions make this RCPSP model less than ideal for modeling some real-world cases. For example, if some labor resources only work five days per week (while others work seven days per week) on a project containing activities with durations longer than five days, no feasible solution can be found as there is no consecutive work period of sufficient length.

In fact, it may be feasible (even preferable) to interrupt some ongoing activities and replace them by other activities until a later time at which the interrupted activities are resumed. Consider the example problem instance in Figure 1, which is adapted from Ballestin *et al.,* (2008). Activities 2 through 6 have a serial precedence relationship while activity 1 is parallel to all of them. Activity 0 is a dummy start activity and activity 7 is a dummy finish activity. Each activity's resource requirement and duration is specified in Figure 1(a). Only one renewable resource is considered in the example and the resource profile is a constant two resource units (Figure 1b). Both solutions in Figure 2 are subject to the resource profile constraint. The optimal makespan when activity splitting is not allowed is C_{max} = 7 time units (Figure 2a), while the optimal makespan with activity splitting is C_{max} = 5 time units as activity 1 is split into three segments (Figure 2b).

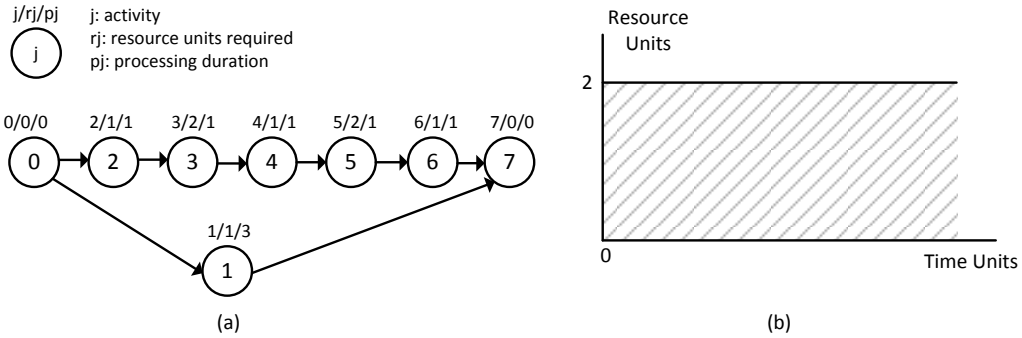

Figure 1: An example of a project network with a single resource

Several research efforts have modified the classical RCPSP to allow activity splitting. One such example is the Preemptive RCPSP (PRCPSP), which allows activities to be interrupted in any time period and resumed later at no additional cost. The idea of preemption is very popular in the machine scheduling literature as well. However, one drawback of the PRCPSP is that activities are allowed to be interrupted arbitrarily, not necessarily because of resource vacations or resource limits changes.

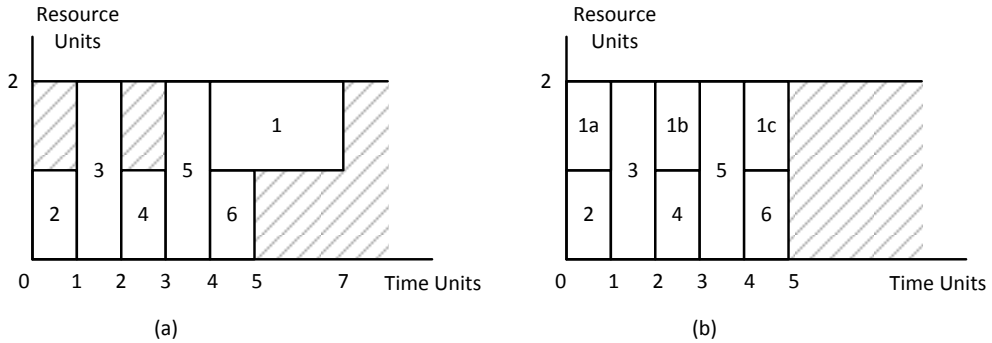

Figure 2: Optimal solutions for the example of an RCPSP without and with activity splitting

Another extension is the RCPSP with calendarization, which focuses on activity splitting resulting from workweek calendars and workday patterns. Unfortunately, most of these research efforts assume constant resource limits during workdays and only consider a single mode of activity processing. In this research, we examine a more general case of calendarization by allowing time-varying resource constraints and multiple processing modes.

In order to illustrate the differences between activity splitting and preemption, consider the project network in Figure 3 containing nine activities (1-9), dummy start (0), and completion nodes (10). One renewable resource $R1$ with the resource profile is provided as well. Resource $R1$ is not available from time unit 6 to time unit 8.

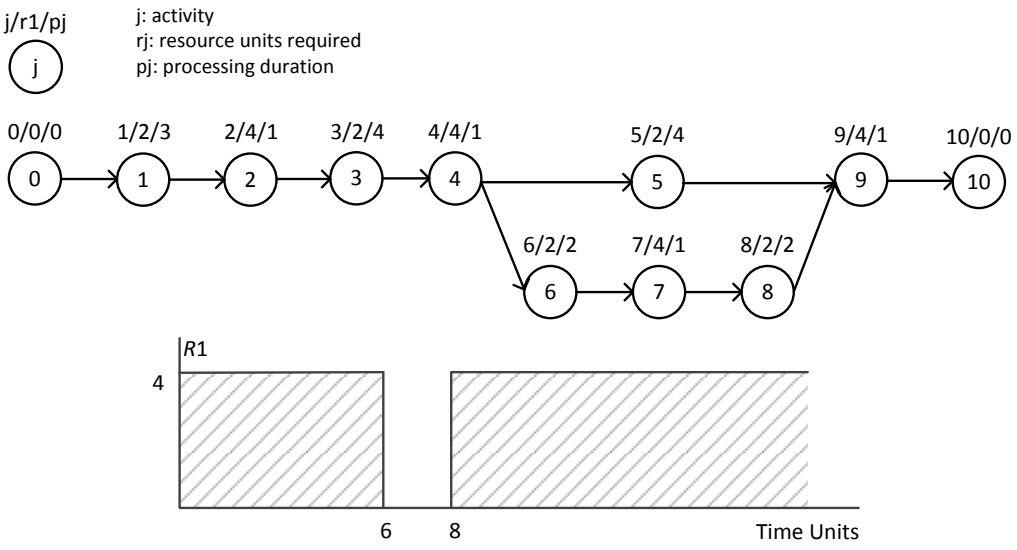

Figure 3: An example of a project network and resource profile

In a schedule illustrated in Figure 4, activity 3 is split from time 6 to time 8 due to

lack of resource and the activity resumes when resource becomes available at time 8; however activity 5 is split from time 13 to 14 even though the resource is available. In practice, these two types of activity splitting might need to be treated differently. To pause an ongoing activity because of resource unavailability and resume it later may have small financial or time impact. However, interrupting an ongoing activity by switching to another activity can result in a high penalty such as setup time lost, re-configuring complicated equipment settings, etc.

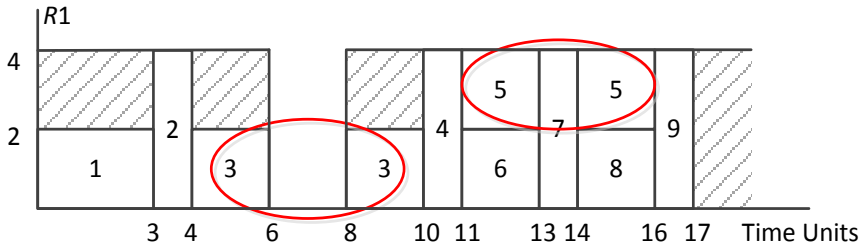

Figure 4: Example of different types of activity splitting

To specify the unique case wherein activities are only allowed to be split when resources are insufficient, we define preemption and activity splitting as follows: a *preempted activity* is an activity for which there is at least one time period after the start of the activity wherein the activity is eligible to be processed but is not being processed. Alternately, a *split activity* is an activity that is not processed in consecutive time periods.

In our previous definitions, a time period that is "eligible" for an activity should be both resource feasible for renewable resources and precedence feasible. According to

our definitions, one can easily see that a preempted activity is a split activity, but the converse is not necessarily true. The cases wherein activities are split may result from insufficient resources rather than by choice. The relationship between preempted and split activities is illustrated in Figure 5. Preemptive RCPSP is a more generalized assumption where activities can be interrupted at any integer time period. RCPSP without activity splitting is the basic assumption in the majority of existing RCPSP research efforts. We denote the special case wherein activity splitting is only allowed when there are insufficient resources as *non-preemptive activity splitting*.

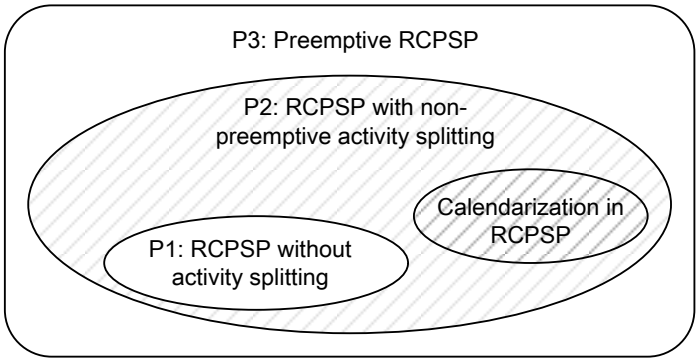

Figure 5: Activity splitting vs. preemption

In non-preemptive activity splitting, an activity that has started processing is allowed to pause only when resource levels are temporarily insufficient. However, it must be resumed at the next eligible processing time period until the activity is completed. Thus, we define three types of problems for our study:

- ***P*1**: RCPSP (or MRCPSP for the Multi-mode case) with no activity splitting

- ***P*2**: Non-preemptive activity splitting RCPSP (or MRCPSP)
- ***P*3**: Preemptive RCPSP (PRCPSP) (or PMRCPSP)

The remaining sections of this paper are organized as follows. In Section 2, we review the literature relevant to our research problem under study. Next, the RCPSP model and different activity splitting cases are described mathematically in Section 3. In Section 4, a modified branch-and-bound algorithm is proposed that is subsequently tested and analyzed in Section 5. Finally, conclusions and directions for future research are offered in Section 6.

2. Literature Review

Literature related to project scheduling dates back to the 1950s. The development of the Program Evaluation and Review Technique (PERT) (Malcolm *et al*., 1959) and the Critical Path Method (CPM) (Kelley, 1963) made it possible to find minimum duration schedules for projects when resource availability was not constrained. Since the availability of resources in real-world projects (e.g., humans, machines, financial budget, etc.) is typically a constraint, researchers started to consider project scheduling problems with resource constraints. Before Johnson (1967) first used the term "resource-constrained project scheduling problem," researchers used different descriptors, including resource allocation in project networks (Davis,1966 and Laue,1968), resource allocation in project planning (Petrović, 1968), project scheduling with resource

constraints (Balas, 1971), and projects with limited resources (Wiest, 1964 and 1967).

Several authors have reviewed the body of RCPSP research knowledge (Davis, 1973, Icmeli *et al*., 1993, Özdamar and Ulusoy, 1995, Herroelen *et al*., 1998, Brucker *et al*., 1999, Kolisch and Padman, 2001, Hartmann and Briskorn, 2010 and Węglarz *et al*., 2011). After years of research on the basic RCPSP, extensions started to attract research attention. According to the emphasis of different aspects, several major RCPSP extensions are summarized in Węglarz *et al.*, (2011):

- Multi-mode RCPSP (MRCPSP)
- Generalized RCPSP (GRCPSP)
- RCPSP with Generalized Precedence Constraints (RCPSP-GPC)
- RCPSP with time-varying resource constraints
- Bi-criteria and multi-criteria RCPSP
- RCPSP or PRCPSP with activity splitting
- Dynamic RCPSP

In the classical RCPSP model, resource limits are assumed constant and activities are not allowed to split. Preemptive RCPSP and RCPSP with activity splitting allow activities to be split. Demeulemeester and Herroelen (1996) show that allowing preemption has limited benefit on makespan reduction but increases computational efforts when resource limits are constant. However, as pointed out in Kolisch *et al*., (1995) and Ballestin *et al.*, (2008), the Patterson (1984) instance set tested in

Demeulemeester and Herroelen (1996) are not generated by controlled parameters and instances are not equally difficult to solve even with the same number of activities.

Buddhakulsomsiri and Kim (2006, 2007) show that with time-varying resource profiles and resource vacations, preemption can reduce project makespan significantly. Ballestin *et al.,* (2008, 2009) support the conclusion that preemption can decrease project length even with a limited number of preemptions allowed for each activity. But the "interruption" discussed in Buddhakulsomsiri and Kim (2006, 2007) and Ballestin *et al.*, (2008, 2009) is in fact activity splitting that includes both preemption and non-preemptive activity splitting.

In most existing RCPSP research, activity splitting and preemption are considered to be interchangeable. Tk̊indt and Billaut (2006) discuss both preemption and activity splitting in the scheduling context that can be specified as *prmp* and *split* in the activity characteristic β field, respectively, according to the classical $\alpha|\beta|\gamma$ classification of Graham *et al.,* (1979) and the RCPSP classification scheme of Brucker *et al.,* (1999) and Herroelen *et al.,* (2001). Other research that treated activity splitting and preemption as equals can be found in Buddhakulsomsiri and Kim (2006), Debels and Vanhoucke (2006), Buddhakulsomsiri and Kim (2007), Damay *et al.* (2007), Peteghem and Vanhoucke (2010), and Węglarz *et al.,* (2011).

Another branch of preemptive scheduling research effort is to assume a preemption penalty. Basically a delay on time or setup cost will be incurred to continue a

job when it is preempted. Several different assumptions are made to the concept of preemption. Preemption-restart assumes an interrupted job can only restart from the beginning of the job which means all current progress on this job will be lost. Researches on this can be found at Zheng *et al*., (2006), Fung *et al*., (2008) and Hoogeveen, Potts, and Woeginger (2000). It makes sense that only on-line scheduling problems need to include a preemption-restart case since an off-line version including the preemption-restart assumption is essentially non-preemption. Preemption-resume is the assumption that most researchers take such that an interrupted job can resume execution at the point where it is preempted with some penalty. Different discussions are also carried out on. Defining and quantifying preemption penalties is also a topic of interest. Several researchers have discussed time-related preemption penalties. For example, job-dependent setup times are assumed on preempted jobs in Zdrzałka (1994), Magazine and Hall (1997), Schuurman and Woeginger (1999), Liu and Cheng (2002) and Liu and Cheng (2004); batch setup times are assumed in Chen (1993) and Monma and Potts (1993). Financial preemption penalty is also studied by many researches. Zheng, Xu, and Zhang (2007) assumes preemption cost and maximizing profit is the scheduling objective; Fung (2008) assumes a penalty cost that related to the value of each preempted job.

Kaplan (1988) is one of the earliest researchers to consider single mode PRCPSP and to provide an exact algorithm for finding optimal solutions for small size problem instances. Demeulemeester and Herroelen (1996) prove that Kaplan's solution algorithm

is incorrect and propose a branch-and-bound procedure instead. Debels and Vanhoucke (2006) also consider PRCPSP but with setup times. Ballestin *et al.,* (2008) and Vanhoucke and Debels (2008) propose heuristics to solve PRCPSP while Damay *et al.*, (2007) provide a linear programming-based algorithm. To the best of our knowledge, there are five papers that have considered PMRCPSP: Nudtasomboon and Randhawa (1997), Prashant Reddy *et al.*, (2001), Buddhakulsomsiri and Kim (2006), Buddhakulsomsiri and Kim (2007), and Peteghem and Vanhoucke (2010). Węglarz *et al.*, (2011) review the first four papers and point out potential research directions in PMRCPSP, such as change of mode when a preempted activity is resumed.

However, existing research efforts on PRCPSP or PMRCPSP are focused on cases where activities can be interrupted at any integer time period. So solutions obtained from these research efforts are less than ideal for some real-world cases where activities allow splitting only when resources are insufficient.

Hallefjord and Wallace (1998) point out that work pattern and calendarization have been neglected in most academic research of project scheduling. The idea of work patterns is treated as the origin of the idea of calendarization in project scheduling. Work patterns are defined by assigning either “workday” or “holiday” to each time period when executing tasks in a project. As tasks can only be executed on “workdays” and not on “holidays,” the idea of work patterns focuses on when the resource is available instead of the amount of the available resource. Preemption is discussed, but only a finite

number of preemptions are allowed in executing each task. Each task might be able to be divided into sub-tasks that do not allow further interruption when executing.

Franck *et al.,* (2001) and Neumann *et al.,* (2003) represent the idea of calendarization by introducing a 0/1 binary function independent from resource constraints. General temporal constraints are given by minimum and maximum time lags. The minimum (maximum) time lags specify an activity can be started a certain unit of time after the start of another activity at the earliest (latest).The activity calendar specifies "work day" or "holiday" on activities instead of on resources. But in practice, if common resources are utilized by a large number of activities, it is easier to define a resource profile or a resource calendar than a large number of activity calendars. When a single mode is assumed, the activity completion time can be uniquely determined by the activity start time and the activity calendar.

Most exact approaches for RCPSP are based on branch-and-bound algorithms such as in Talbot (1982), Patterson *et al.*, (1989), Speranza and Vercellis (1993), Sprecher (1994), Sprecher *et al.*, (1997), Hartmann and Drexl (1998), and Sprecher and Drexl (1998). Hartmann and Drexl (1998) propose a precedence tree-based branch-and-bound algorithm and conclude that it outperforms others. The branch-and-cut-based algorithm proposed by Zhu *et al.,* (2006) is also very promising but requires a longer running time compared to the precedence tree-based branch-and-bound algorithm.

The precedence tree-based branch-and-bound algorithm is essentially an

enumeration scheme that evaluates all possible partial schedules. Efficient bounding rules such as data reduction rules, initial solution rules, and time window rules can improve performance (Hartmann and Drexl, 1998). Buddhakulsomsiri and Kim (2006) modify the precedence tree-based branch-and-bound algorithm to solve PRCPSP and point out 97% of the instances they explore can be solved optimally within one hour, but there are still several instances that required over 40 hours obtaining an optimal solution. As pointed out by Węglarz *et al.*, (2011), it is still computationally intractable to find optimal solutions for instances with more than 20 activities.

In summary, previous efforts in RCPSP usually treat preemption and activity splitting interchangeable. However, we identify a case where activities can only split when there are insufficient resources. Time-varying resource constraints and resource vacations are considered and each activity has multiple processing modes where mode switching is not allowed. Our research can be considered as an extension of RCPSP with calendarization to include time-varying resource constraints and multiple processing modes for each activity.

3. Problem Statement

In a project network $G(N, A)$, the set of nodes N represents the activity set(= node set)N ($|N| = n$) and a set of directed arcs A represents the precedence relations among activities. While there are generalized precedence constraints in the existing literature

(see Sprecher, 1994 and Brucker and Knust, 2006), precedence relations considered in this research are *finish-to-start* with *zero time-lags* relations. For network completeness purposes, a dummy node 0 is added as the super source node and a dummy node $n + 1$ is added as the super sink node. Within this paper, we treat "activities", "tasks," and "jobs" interchangeably if not otherwise stated.

Table 3: Mathematical Notation

Symbols	Description
j	Activity/task/job
t	Time period
k	Resource type
m	Activity processing mode
Mod_j	Set of available processing modes for activity j
p_j^m	Processing duration for activity j under mode m
r_{jk}^m	Required amount for resource type k on activity j under mode m
U_k	Available upper bound for non-renewable resource type k
U_{kt}	Available upper bound for renewable resource type k at time t
N	Activities in the project $N = \{1, 2, \dots, n\}$
R^r	Set of renewable resources
R^n	Set of non-renewable resources
T	Maximum project planning horizon
rad_j	Ready time for activity j
due_j	Due date for activity j
$G(N, A)$	Network G with N represents nodes and A represents arcs
$arc(i, j)$	Directed arc connecting node i to node j
$pred(j)$	Set of predecessor activities of activity j; $pred(0) = \emptyset$
$succ(j)$	Set of successor activities of activity j; $succ(\lvert N \rvert + 1) = \emptyset$

Furthermore, both a set R^r of renewable resources and a set R^n of non-renewable resources are considered in this paper. In each time period t, the availability of a renewable resource k ($k \in R^r$) is restricted to be between the 0 and the upper resource

limit U_{kt}. The resource limit $[0, U_{kt}]$ is the "resource profile" function that specifies the availability of a particular resource over time. Throughout the entire project planning horizon $[0, T]$, the availability of a non-renewable resource k ($k \in R^n$) is limited between 0 and the upper resource limit U_k. Each activity j ($j \in N$) has a set of available processing modes Mod_j to choose from and each mode $m \in Mod_j$ has a corresponding activity duration p_j^m and consumes r_{jk}^m amount of resource k. The mathematical formulations for $\boldsymbol{P1}$, $\boldsymbol{P2}$, and $\boldsymbol{P3}$ use similar mathematical notation and variables (Table **3**) to the $\boldsymbol{P3}$ formulation in Buddhakulsomsiri and Kim (2006).

The primary decision variables are as follows: $y_j^m = 1$ if activity $j \in N$ is being processed in mode $m \in Mod_j$ and 0 otherwise; $x_{jt}^m = 1$ if activity $j \in N$ is being processed in mode $m \in Mod_j$ at time $t = 1, 2, \dots, T$ and 0 otherwise. In addition, variables S_j and C_j represent the start time and completion time of activity j and the start time of the dummy finish activity $S_{|N|+1}$ is essentially the project makespan. Data inputs are resource profiles $[0, U_{kt}]$ for renewable resources and $[0, U_k]$ for non-renewable resources. The PMRCPSP ($\boldsymbol{P3}$) formulation as given by Buddhakulsomsiri and Kim (2006) can be represented as follows:

$$\min\ S_{|N|+1} \tag{1}$$

subject to

$$\sum_{m \in Mod_j} y_j^m = 1,\ 1, \forall j \in N \tag{2}$$

$$\sum_{t=1}^{T} x_{jt}^m = p_j^m \cdot y_j^m, \quad \forall j \in N, m \in Mod_j \tag{3}$$

$$C_i \leq S_j - 1, \quad \forall (i,j) \in A \tag{4}$$

$$S_j \leq x_{jt}^m \cdot t + M\left(1 - x_{jt}^m\right), t \; \forall j \in N, m \in Mod_j, t = 1, 2, \dots, T \tag{5}$$

$$C_j \geq x_{jt}^m \cdot t, \quad \forall j \in N, m \in Mod_j, t = 1, 2, \dots, T \tag{6}$$

$$S_j \geq rad_j, \quad \forall j \in N \tag{7}$$

$$C_j \leq due_j, \quad \forall j \in N \tag{8}$$

$$\sum_{j \in N} \sum_{m \in Mod_j} r_{jk}^m \cdot x_{jt}^m \leq U_{kt}, t \; \forall k \in R^r, t = 1, 2, \dots, T \tag{9}$$

$$\sum_{t=1}^{T} \sum_{j \in N} \sum_{m \in Mod_j} r_{jk}^m \cdot x_{jt}^m \leq U_k, td \forall k \in R^n \tag{10}$$

$$y_j^m \in \{0, 1\}, \; 1 \; \forall j \in N, m \in Mod_j \tag{11}$$

$$x_{jt}^m \in \{0, 1\}, \quad \forall j \in N, m \in Mod_j, t = 1, 2, \dots, T \tag{12}$$

$$S_j \geq 0, \; \forall j \in N \tag{13}$$

$$C_j \geq 0, \; \forall j \in N \tag{14}$$

The objective function (1) minimizes the project makespan which can be represented by the starting time of the dummy finish activity $|N| + 1$. Constraint set (2) ensures exactly one mode is selected for each activity. Constraint set (3) ensures that if mode m is selected for activity j, the total processing time must equal the corresponding duration. Constraint sets (4) – (6) are precedence constraints and a big number M can be set as the maximum project planning horizon T. The "-1" in (4) removes strict inequality given integer time units (e.g. an arc (2, 3) and activity 3 starts on time unit 5, $S_3 = 5$, activity 2 has to complete before or on time unit 4 $C_2 \leq 5 - 1$). Activity ready times and due dates constraints are in (7) - (8). Constraint sets (9) – (10) specify resource

availability for both renewable resources and non-renewable resources, respectively. Constraint sets (11) – (14) are binary (11 and 12) and non-negativity (13 and14) constraints.

To modify the $\boldsymbol{P3}$ formulation for $\boldsymbol{P1}$, constraint set (15) is added to ensure that the duration from the activity start time to the completion time equals the activity duration. In other words, there is no activity splitting for any activity.

$$C_j - S_j = \sum_{m \in Mod_j} \sum_{t=1}^{T} x_{jt}^m - 1, \quad \forall j \in N \tag{15}$$

To modify the $\boldsymbol{P3}$ formulation for $\boldsymbol{P2}$, an indicator function is introduced to specify whether an activity j in mode m is feasible to process at a certain time period:

$$\gamma_{jkt}^m = 1_{[0,U_{kt}]}\left(r_{jk}^m\right) := \begin{cases} 1 & \text{if } r_{jk}^m \in [0, U_{kt}],\ \forall t \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

Additional decision variables o_{jt} and q_{jt} are defined to indicate whether a time period t is between the start time S_j and the completion time C_j of activity j.

$$o_{jt} = \begin{cases} 1 & \text{if } t \le C_j \\ 0 & \text{otherwise} \end{cases}, \quad \forall j \in N \tag{17}$$

$$q_{jt} = \begin{cases} 1 & \text{if } t \ge S_j \\ 0 & \text{otherwise} \end{cases}, \quad \forall j \in N \tag{18}$$

Additional constraint sets (19) – (22) are included to support the new decision variables o_{jt} and q_{jt}. As before, a big number M can be set as the maximum project planning horizon T. Constraint sets (23) – (24) restrict that an activity j cannot be preempted at time t if it is eligible. Constraint sets (25) – (26) are additional variable type constraints.

$$M \cdot o_{jt} \ge C_j - t + 1, \quad \forall j \in N, t = 1, 2, \ldots T \tag{19}$$

$$M \cdot \left(1 - o_{jt}\right) \geq t - C_j, \quad \forall j \in N, t = 1, 2, \dots T \tag{20}$$

$$M \cdot q_{jt} \geq t - S_j + 1, \ Eq \forall j \in N, t = 1, 2, \dots T \tag{21}$$

$$M \cdot \left(1 - q_{jt}\right) \geq S_j - t, \quad \forall j \in N, t = 1, 2, \dots T \tag{22}$$

$$x_{jt}^m \geq y_j^m + \gamma_{jkt}^m + o_{jt} + q_{jt} - 3, \ \forall j \in N, m \in Mod_j, k \in R^n, t = 1, 2, \dots T \tag{23}$$

$$4 \cdot x_{jt}^m \leq y_j^m + \gamma_{jkt}^m + o_{jt} + q_{jt}, t \ \forall j \in N, m \in Mod_j, k \in R^n, t = 1, 2, \dots T \tag{24}$$

$$o_{jt} \in \{0, 1\}, \ 1 \ \forall j \in N, t = 1, 2, \dots T \tag{25}$$

$$q_{jt} \in \{0, 1\}, \quad \forall j \in N, t = 1, 2, \dots T \tag{26}$$

4. Approach

The focus of this work is to examine the difference between problems $\boldsymbol{P1}$, $\boldsymbol{P2}$, and $\boldsymbol{P3}$ in terms of problem settings, mathematical formulation, and optimal solution properties. In this section, several observations for problems $\boldsymbol{P1}$, $\boldsymbol{P2}$, and $\boldsymbol{P3}$ are provided, along with an exact algorithm that is proposed to solve $\boldsymbol{P2}$ optimally for small size problem instances. Considering some basic scenarios for $\boldsymbol{P1}$, $\boldsymbol{P2}$, and $\boldsymbol{P3}$, the following two propositions follow:

Proposition 1. *All feasible solutions for* $\boldsymbol{P1}$ *are also feasible for* $\boldsymbol{P2}$*; all feasible solutions for* $\boldsymbol{P2}$ *are also feasible for* $\boldsymbol{P3}$*. Therefore, all feasible solutions for* $\boldsymbol{P1}$ *are also feasible for* $\boldsymbol{P3}$.

Solutions obtained from $\boldsymbol{P1}$ can be considered allowing activity splitting, but no activity has been split. The same argument can apply from $\boldsymbol{P1}$ to $\boldsymbol{P3}$ and from $\boldsymbol{P2}$ to $\boldsymbol{P3}$.

Proposition 2. *When there are constant resource limits and no resource vacations, the solution space of* $\boldsymbol{P1}$ *is the same as the solution space of* $\boldsymbol{P2}$**,** *and the optimal solution for* $\boldsymbol{P2}$ *is also feasible and optimal for* $\boldsymbol{P1}$.

The argument is straightforward since no activity will be split when there are constant resource limits and no resource vacations. Thus solutions obtained from $\boldsymbol{P2}$ are essentially the same as those for $\boldsymbol{P1}$.

Table 4: Notation for precedence tree-based branch-and-bound algorithm

Notation	Description
g	Precedence tree level
j_g	Activity j selected at level g of the precedence tree
m_{j_g}	Selected mode for activity j_g
Mod_{j_g}	Set of available modes for activity j_g
S_{j_g}	Start time of activity j_g
C_{j_g}	Completion time of activity j_g
$p_{j_g}^{m_{j_g}}$	Duration of activity j_g at mode m_{j_g}
SJ_g	Set of already scheduled activities at level g
AJ_g	Set of active activities at level g(active activity: An activity that has not been scheduled but all of its predecessors are completely scheduled.)
EST_j	Earliest precedence feasible start time of activity j

Beyond these simple scenarios, a modified precedence tree-based branch-and-bound algorithm is proposed to solve $\boldsymbol{P2}$. Table 4 provides basic notation for the algorithm.

4.1 Precedence tree-based branch mechanism

The precedence tree-based branch-and-bound algorithm is essentially an enumeration scheme that evaluates all possible partial schedules. The pseudo code below illustrates the steps in the algorithm.

Algorithm: Precedence tree-based branch-and-bound algorithm for $\boldsymbol{P2}$:

Step 1: Initialization

Set the precedence tree level $g = 1$;

Schedule the dummy start activity $j = 0$ at time zero: $j_1 = 0, m_{j_1} = 1, S_{j_1} = 0$;

Set the already scheduled activity $SJ_1 = \emptyset$.

Step 2: Update the set of active activities

Increase the precedence tree level $g = g + 1$

Update the set of already scheduled activities $SJ_g = SJ_{g-1} \cup \{j_{g-1}\}$

Compute the set of active activities $AJ_g = \{j \in N \backslash SJ_g | pred(j) \subseteq SJ_g\}$

If the last activity (dummy completion) is active, i.e., $n + 1 \in AJ_g$, then store the current solution and go to step 5. Else, go to step 3.

Step 3: Select the next activity from AJ_g to be scheduled

If there is no untested activity left in AJ_g, then go to step 5

Else, randomly select an untested activity $j_g \in AJ_g$

Step 4: Select a mode for the selected job and schedule the activity

If there is no untested mode left in $\{1, \dots, Mod_{j_g}\}$, then go to step 3;

Else, randomly select an untested mode $m_{j_g} \in \{1, \dots, Mod_{j_g}\}$.

Compute the earliest precedence feasible start time, $EST_{j_g} = \max\{C_{j_g} | i \in pred(j_g)\} + 1$

Compute the start time S_{j_g} and completion time C_{j_g} based on the following constraints to satisfy non-preemptive activity splitting:

1. $S_{j_g} \geq EST_{j_g}$
2. $p_{j_g}^{m_{j_g}} = \sum_{t=S_{j_g}}^{C_{j_g}-1} (1_{[0,U_{kt}]})$
3. $\forall t \in [S_{j_g}, C_{j_g} - 1]$, if $1_{[0,U_{kt}]} = 1$, then $x_{jt}^{m} = 1$

Go to step 2.

Step 5: Backtracking

Decrease the precedence tree level by 1, $g = g - 1$

If the precedence tree level is 1, then STOP;
Else to go step 4.

4.2 Bounding Rules

We adopt basic bounding rules in Hartmann and Drexl (1998) and in Buddhakulsomsiri and Kim (2006) including the time window rule (latest completion time), data reduction rules, and precedence tree-specific rules. Since non-renewable resources are not considered in Buddhakulsomsiri and Kim (2006), the mode infeasibility rule regarding non-renewable resources is adopted from Hartmann and Drexl (1998). The ***P*1** version of this branch-and-bound algorithm can be found in both Hartmann and Drexl (1998) and Buddhakulsomsiri and Kim (2006), while the ***P*3** version can be found in Buddhakulsomsiri and Kim (2006).

For every activity j, the lower bound for the activity duration p_j^{LB} is the shortest duration among all modes $p_j^{LB} = \min\{p_j^m\}$; the lower bounds for resource requirements r_{jk}^{LB} are the shortest resource usages among all modes for all resource types $r_{jk}^{LB} = \min\{r_{jk}^m\}$. It is worth mentioning that the lowest duration and lowest resource usages often belong to different modes.

4.3 Modified Time Window Rule

The traditional time window rule uses the critical path approach to determine the EST (earliest start time), EFT (earliest finish time), LST (latest start time) and LFT

(latest finish time) time window without resource consideration. However since for RCPSP where resource availability plays an important role in determining a schedule, adding resource constraints can tighten the scheduling time window for each activity. Given resource profiles for renewable resources, the EST and LFT not only need to satisfy precedence constraints, but also need to satisfy resource constraints assuming the lower bound resource requirements for each activity. For $\boldsymbol{P1}$ where activity splitting is not allowed and $\boldsymbol{P2}$ where preemption is not allowed, additional activity splitting constraints also need to be considered. A detailed description can be found below.

Time Windows Determination

Forward Pass

Initialize the dummy start activity into the active activity set $AJ = \{0\}$, initialize the flagged activity set $FG = \emptyset$

Determine the EST and EFT for the dummy start activity: $EST_o = EFT_0 = 0$, add dummy start activity into $FG = \{0\}$

Exam each activity j, if each predecessor activity l of activity j belongs to flagged activity: $\forall l \in pred(j)$, $l \in FG$, determine the EST_j and EFT_j for activity j such that $EST_j = \min\{t^*\}$, $EFT_j = \min\{t^{**}\}$:

$$t^* \geq \max_{l \in pred(j)}\{EFT_l\} + 1$$

$$\forall t \in [t^*, t^{**}], \sum_{t=t^*}^{t=t^{**}} \gamma_{jkt}^{LB} \geq p_j^{LB} \text{ for } \forall k \in R^r$$

If no such time t^* and t^{**} can be found, return infeasible. For $\boldsymbol{P1}$ where activity splitting is not allowed, these single unit time periods need to be continuous: $t^{**} = t^* + p_j^{LB} - 1$; for $\boldsymbol{P2}$ where activity splitting is allowed but preemption is not allowed: $\forall t \in [t^*, t^{**}]$, $\gamma_{jkt}^{LB} = 1$; for $\boldsymbol{P3}$, these single unit time periods do not need to be continuous.

If the dummy finish activity N is flagged, set $EST_N = EFT_N = LST_N = LFT_N$ and start backward pass.

Backward Pass

Backward pass follows the similar logic as the forward pass until the dummy start activity is flagged. To determine the LFT_j and LST_j for activity j such that $LFT_j = \max\{t^*\}$, $LST_j = \max\{t^{**}\}$:

$$t^* \leq \min_{l \in succ(j)}\{LST_l\} - 1$$

$\forall t \in [t^{**}, t^{*}]$, $\sum_{t=t^{**}}^{t=t^{*}} \gamma_{jkt}^{LB} \geq p_j^{LB}$ for $\forall k \in R^r$

If no such time t^* and t^{**}can be found, return infeasible.

For $\boldsymbol{P1}$, these single unit time periods need to be continuous: $t^* = t^{**} + p_j^{LB} - 1$; for $\boldsymbol{P2}$ where activity splitting is allowed but preemption is not allowed: $\forall t \in [t^{**}, t^{*}]$, $\gamma_{jkt}^{LB} = 1$; for $\boldsymbol{P3}$, these single unit time periods do not need to be continuous.

If the dummy start activity is flagged, stop the algorithm.

4.4 Data Reduction Rules

Before scheduling a project, initial data screening is conducted to remove infeasible or dominated modes. A mode m for activity j is infeasible or dominated if one of the following conditions holds:

- Infeasible regarding to a non-renewable resource such that even if all other activities choose the LB resource requirement, the total non-renewable resource required is more than available limit:

$$r_{jk}^{m} + \sum_{l \neq j, l \in N} r_{lk-LB}^{m} > U_k, \forall k \in R^n$$

- Infeasible regarding to a renewable resource and activity duration if that during the $[\,EST_j\,,\,LFT_j\,]$ time window, there are not enough time periods that satisfy the renewable resource requirement and activity duration. Mathematically, $\nexists t^*, t^{**} \in \left[EST_j, LFT_j\right]$, such that $\forall t \in [t^*, t^{**}]$, $\sum_{t=t^*}^{t=t^{**}} \gamma_{jkt}^{m} \geq p_j^{m}$ for $\forall k \in R^r$. For $\boldsymbol{P1}$ where activity splitting is not allowed, these single unit time periods need to be continuous: $t^{**} = t^* + p_j^{m}$; for $\boldsymbol{P2}$ where activity splitting is allowed but preemption is not allowed: $\forall t \in [t^*, t^{**}]$, $\gamma_{jkt}^{m} = 1$; for $\boldsymbol{P3}$, these single unit time periods do not need to be continuous.

- A mode j^m is dominated by mode j^{m^*} for activity j if it requires at least as resources and has a duration that is at least as long:

$$p_j^m \geq p_j^{m^*}, r_{jk}^m \geq r_{jk}^{m^*}, \forall k \in R$$

4.5 Initial Solutions

One main modification of the branch-and-bound algorithm is to add a better initial solution as bounding rules. We use multiple priority rule-based simple heuristics and return the best solution found by several simple heuristics as the initial solution.

To solve MRCPSP, priority rule-based heuristics combine mode selection rules which determine mode assignments for each activity and activity priority rules which specify the activity loading sequence. A combination of a mode selection and an activity selecting rule uniquely determines a schedule (however the reverse does not hold true since different rules might reach the same schedule). A schedule generation scheme (SGS) is necessary to determine the transforming mechanism from a heuristic rule to a schedule. Detail description of the SGS can be found in Cheng *et al.*, (2013). Other discussions of SGSs can be found at references like Sprecher *et al.*, (1995) and Kolisch and Hartmann (1999). Table 5 and Table 6 summarize most commonly studied mode selection and activity priority rules and are all included in the initial solution generation. SDM (shortest duration mode) selects mode with shortest processing duration among all available processing modes for an activity; LTRU_R (least total renewable resource

usage) rule chooses the mode that utilizes the least amount of renewable resource while LTRU_N (least total non-renewable resource usage) prioritizes the mode that requires the least amount of non-renewable resource.

Table 5: Mode Selection Rules

Priority Rules	Mathematical Formula	Selected Reference
SDM (shortest duration mode)	$\{m \in Mod_j \mid p_j^m = min_{\forall l \in Mod_j} p_j^l\}$	Boctor (1996), Buddhakulsomsiri and Kim (2007)
LTRU_R (least total renewable resource usage)	$\left\{m \in Mod_j \mid \sum_{k \in R^r} \left(r_{jk}^m \cdot p_j^m\right) = \min_{\forall l \in Mod_j} \sum_{k \in R^r} \left(r_{jk}^l \cdot p_j^l\right)\right\}$	Boctor (1996), Buddhakulsomsiri and Kim (2007)
LTRU_N (least total non-renewable resource usage)	$\left\{m \in Mod_j \mid \sum_{k \in R^n} r_{jk}^m = \min_{\forall l \in Mod_j} \sum_{k \in R^n} r_{jk}^l\right\}$	Boctor (1996), Buddhakulsomsiri and Kim (2007)

Table 6: Activity Priority Rules

Priority Rules	Mathematical Formula	Selected Reference
SPT (shortest processing time)	$\left\{j \in N \mid p_j^m = \min_{l \in N} p_l^m\right\}$	Alvarez-Valdes and Tamarit (1989), Lova *et al.*, (2006)
LPT (longest processing time)	$\left\{j \in N \mid p_j^m = \max_{l \in N} p_l^m\right\}$	Alvarez-Valdes and Tamarit (1989), Lova *et al.*, (2006)
ERT (earliest ready time)	$\left\{j \in N \mid rad_j = \max_{l \in N} rad_l\right\}$	This research
EDD (earliest due date)	$\left\{j \in N \mid due_j = \min_{l \in N} due_l\right\}$	This research
MSLK (minimum slackness)	$\left\{j \in N \mid LST_j - EST_j = \min_{l \in N}(LST_l - EST_l)\right\}$	Davis and Patterson (1975) Buddhakulsomsiri and Kim (2007)
MLST (minimum latest start time)	$\left\{j \in N \mid LST_j = \min_{l \in N} LST_l\right\}$	Alvarez-Valdes and Tamarit (1989), Kolisch (1995)
MLFT (minimum latest finish time)	$\left\{j \in N \mid LFT_j = \min_{l \in N} LFT_l\right\}$	Davis and Patterson (1975)

MTS (maximum total successors)	$\left\{j \in N \mid \lvert succ(j)\rvert = \max_{l \in N} \lvert succ(l)\rvert\right\}$	Alvarez-Valdes and Tamarit (1989)
GRPW (greatest rank positional weight)	$\left\{j \in N \mid p_j^m + \sum_{i \in ALL\_succ(j)} p_j^m = \max_{l \in N}\left(p_j^m + \sum_{i \in ALL\_succ(j)} p_j^m\right)\right\}$	Helgeson and Birnie (1961), Buddhakulsomsiri and Kim (2007)

Activity priority rules determine how to select the next activity to be scheduled along with precedence constraints. The SPT (shortest processing time) rule chooses the activity with the shortest activity duration. The LPT (longest processing time) rule selects an activity with the longest duration and is the basic scheduling method applied in the current practice of the Install/Qual process. The ERT (earliest ready time) rule selects the activity with the earliest ready time and the EDD (earliest due date) rule chooses the activity with the earliest due date. MSLK (minimum slackness) is the rule that the activity with the minimum slackness has the highest priority to be scheduled first. Slackness is obtained from the difference between LST (latest start time) and EFT (earliest finish time) of an activity. LFT (latest finish time) and EST (earliest state time) are calculated by backward recursion method in (Pinedo, 2008) with consideration of the due date and ready time of each activity (a LFT cannot be later than the due date, and an EST cannot be earlier than its ready time). MMSLK (modified minimum slackness) modifies the MSLK rule by including activity durations. MLST (minimum latest start time) is the rule that prioritizes the activity with the smaller LST. MLFT (minimum latest finish time) is the rule that gives priority to the activity with the smallest LFT. MTS

(maximum total successors) rule prioritizes the activity with the maximum number of immediate successors. GRPW (greatest rank positional weight) is a widely used rule originally proposed by Helgeson and Birnie (1961) as a line balancing method in the machine scheduling literature. For single mode RCPSP, this rule prioritizes the activity with the highest cumulative sum of the individual processing times and the processing time of all successors of that activity. The sum is the so-called "positional weight" of that activity (see Alvarez-Valdes and Tamarit, 1989).

5. Computational Experiments

5.1 Experimental Design

As discussed in Buddhakulsomsiri and Kim (2006), allowing preemption can reduce the makespan when resource constraints are time-varying. In this section, computational experiments are conducted to identify factors and problem characteristics that lead to makespan reduction in problems $\boldsymbol{P2}$ and $\boldsymbol{P3}$ as compared to $\boldsymbol{P1}$.

The tested problem instances are from a well-known online library PSPLIB (Kolisch and Sprecher, 1997) and generated by a project generator ProGen (Kolisch *et al.*, 1995). However, the benchmark problem instances from PSPLIB and ProGen do not consider time-varying resource profiles, resource vacations or activity ready times and due dates. In this research, time-varying resource constraints are generated by introducing randomness around the constant resource limits. Other approaches for

generating time-varying resource constraints can be found in Klein (2000) and Böttcher *et al.,* (1999). Resource vacations are generated by adding vacation patterns. Activity ready times and due dates are randomly generated based on the entire project horizon T. Details of our instance modification procedure can be found in the appendix. Six major factors considered in this experiment are briefly discussed in the following paragraphs. Network complexity (NC), resource factor (RF), resource strength (RS) are adopted from PSPLIB and ProGen. Resource range (RR) and), resource vacation (RV) and ready time/due date (RD) are new proposed parameters.

5.2 Network Complexity ($\boldsymbol{NC}$)

Network complexity is measured as the average number of non-redundant arcs per node including the dummy start and completion nodes. An arc (h, j) is redundant if there are arcs (i_0, i_1), …, (i_{s-1}, i_s) in the network with $i_0 = h$, $i_s = j$ and $s \geq 2$ (Kolisch *et al.*, 1995). A detailed description on how to construct a network for a given NC level can be found in Kolisch *et al.,* (1995) and Kolisch and Sprecher (1997). Hartmann and Kolisch (2000) and Buddhakulsomsiri and Kim (2006) point out that the NC factor does not significantly affect the makespan difference between $\boldsymbol{P1}$ vs. $\boldsymbol{P3}$. In this experiment, we set network complexity factor constant since the activity precedence network in the practical semiconductor problem is not a decision point (precedence relation among activities are pre-determined and will not be impacted).

5.3 Resource Factor (***RF***)

Resource factor measures the average percentage of resource types required per activity. $RF = 1$ means each activity requires all types of resources while $RF = 0$ means no activity requires any resource (scheduling without resource constraints). In the experiment, RF has two levels: $RF = 0.5$ and $RF = 1$.

5.4 Resource Strength (***RS***)

Resource strength measures the "tightness (richness)" of a resource and it is normalized on a 0 – 1 scale. $RS = 0$ (tightest) means the minimum resource level such that there is a feasible schedule; $RS = 1$ (richest) indicates all resource are available enough so that all activities can be scheduled at their earliest start time. In the experiment, two levels of RS are selected, $RS = 0.2$ and $RS = 0.7$, since they are the basic levels in PSPLIB.

5.5 Resource Range (***RR***)

Resource range is a percentage value that measures the maximum width of resource limits over time. The resource limit U_{kt} is generated randomly using a uniform distribution:

$$U_{kt} \sim \text{uniform}(U_{k0}^{'}(1 - RR), U_{k0}^{'}(1 + RR))$$

in which $U_{k0}^{'}$ is assumed to be the "baseline" level of renewable resource k. In

Figure 6, the resource limit in (a) is constant ($RR = 0$). Figure 6(b) has a time-varying resource profile at a relative low level ($RR = 0.25$). The dashed lines around the constant resource level indicate the upper and lower bound of the uniformly distributed random numbers. The time-varying resource limit is at a medium level when $RR = 0.5$ (6c) and at a high level when $RR = 0.75$ (6d). $RR = 1.0$ is not considered since the resource level could be zero when $RR = 1.0$.

5.6 Resource Vacation ($\boldsymbol{RV}$)

Resource vacation is a binary factor that captures the fact that it is very possible that a resource is totally unavailable in some time periods. For example, human resources in many practical cases are not available during weekends and holidays. Second, unavailable resource time periods usually follow some kind of pattern. If a resource is only available five days per week, the resource is not available two days during every seven-day period.

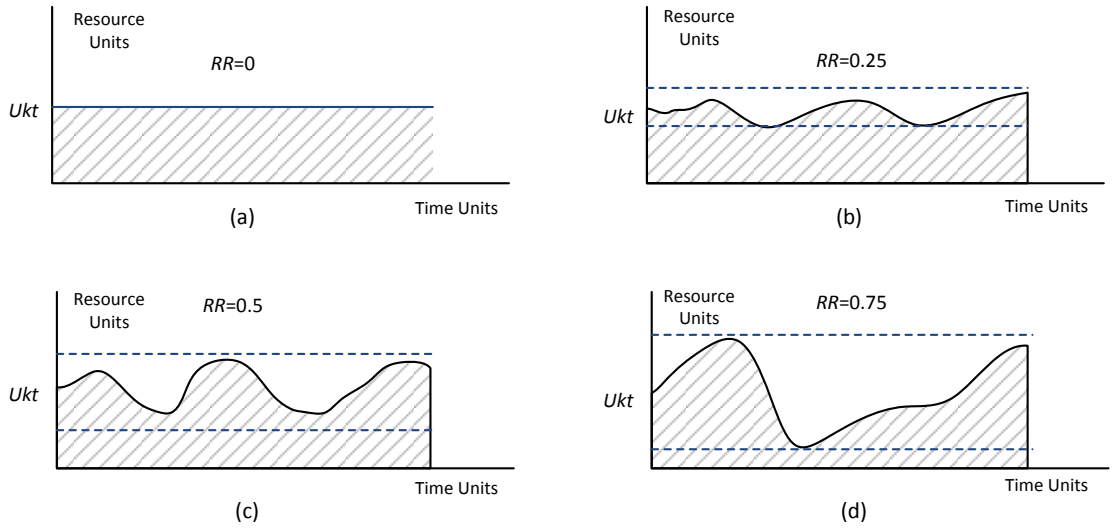

Figure 6: Resource range levels

If $RV = 0$ (false), no resource vacations are considered in the resource profile, while for $RV = 1$ (true), resource vacations are included. To apply the idea of resource vacations, the pattern of vacations should be defined first. Since the resource vacation patterns are highly dependent on the actual problem, we do not provide a general formulation for the resource pattern. In this experiment, $RV = 1$ indicates that there is one day with no resource available every 14 days. The pattern of 2 weeks = 14 days is selected since the activity durations from PSPLIB and ProGen are from 1 to 10, if the vacation is every seven days, there might not be any feasible solution when there is activity longer than seven days for $P1$. Figure 7(a) is the resource profile for $RR = 0$ and $RV = 0$, while Figure 7(b) represents $RR = 0$ and $RV = 1$; Figure 7(c) is a resource profile generated by setting $RR = 0.75$ and $RV = 0$, and Figure 7(d) is a resource profile generated by setting $RR = 0.75$ and $RV = 1$.

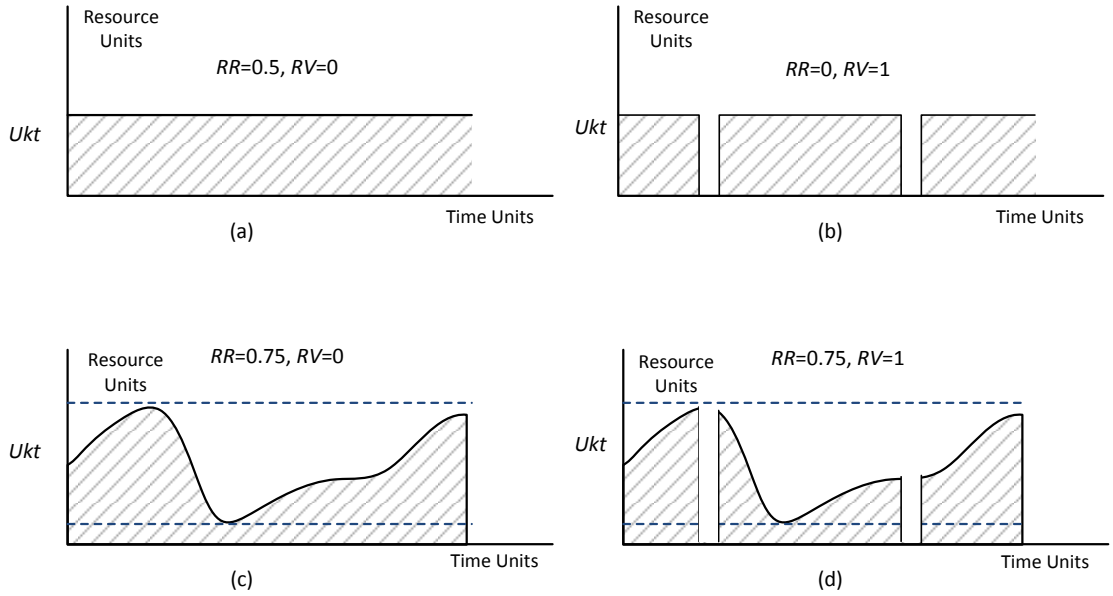

Figure 7: Resource range and resource vacation factors

One thing worth mentioning is that the differences between $\boldsymbol{P1}$, $\boldsymbol{P2}$, and $\boldsymbol{P3}$ are related to renewable resources since activity splitting and preemption are time-related and thus only apply to renewable resources. Even though non-renewable resources are considered in our research, they are not included as possible factors that might distinguish $\boldsymbol{P1}$, $\boldsymbol{P2}$, and $\boldsymbol{P3}$. Thus, RSR and RFR are resource strength and resource factor for renewable resources, respectively.

5.7 Ready Time and Due Date ($\boldsymbol{RD}$)

Activity ready times and due dates are well-studied concepts in the machine scheduling literature (Pinedo, 2008). In the project scheduling literature, generalized

precedence constraints can be indirectly used to model activity ready times and deadlines (De Reyck and Herroelen, 1999). In this research, two levels (loose, tight) of activity ready times and due dates settings are generated based on the project horizon as follows. If $RD = \text{loose}$: $rad_j \sim \text{uniform}(0, 5\% \cdot T)$, $due_j \sim \text{uniform}(95\% \cdot T, T)$; while if $RD = \text{tight}$: $rad_j \sim \text{uniform}(0, 10\% \cdot T)$, $due_j \sim \text{uniform}(90\% \cdot T, T)$. The determining of using 5% as loose and 10% for tight ready times and due dates levels is based on preliminary test runs.

The computational experiment contains two parts in order to answer to different questions. The first part is to show whether there is a significant makespan difference between $\boldsymbol{P1}$, $\boldsymbol{P2}$, and $\boldsymbol{P3}$ while the second part of the experiment is to identify significant factors that can distinguish $\boldsymbol{P1}$, $\boldsymbol{P2}$, and $\boldsymbol{P3}$. Each tested project instance has three alternative processing modes. The activity duration is generated from a discrete uniform distribution from $[1, 10]$ time units. There are two types of renewable resources and two types of non-renewable resources. Table 7 summarizes the major factors levels that are studied.

The experiment is a full factorial design with 8 replicates to balance the computational time and accuracy of the experiment. Theoretically there would be $2 \cdot 2 \cdot 2 \cdot 4 \cdot 2 \cdot 2 \cdot 2 \cdot 8 = 2048$ problem instances but only a total of 1538 instances are generated and tested since certain parameter settings (e.g. $RSR = 0.25$) are not able to construct an instance.

Table 7: Factor levels overview

Factors	N	RFR	RSR	RR	RV	RD	Initial Solution
Levels	12	0.5	0.25	0	0	Loose	No
	16	1	0.75	0.25	1	Tight	Yes
				0.5			
				0.75			

The precedence tree-based branch-and-bound algorithm is programmed in C++ (Microsoft Visual Studio 2008 version: http://www.microsoft.com/visualstudio), and the experiment was conducted on a desktop with an Intel® 2 Quad Core™ CPU Q9400 @ 2.66GHz, 4.00 GB installed memory, and the Windows 7 Enterprise 64-bit Operating System. The computational time for most problem instances is less than one minute, but there are a few instances that require a few hours to solve.

5.8 Experiment part 1: Binary response

For each problem instance tested in experiment part 1, the response is either a 1 (makespan difference) or a 0 (no makespan difference) between $\boldsymbol{P1}$ vs. $\boldsymbol{P3}$, $\boldsymbol{P1}$ vs. $\boldsymbol{P2}$, and $\boldsymbol{P2}$ vs. $\boldsymbol{P3}$. The percentage of instances that have makespan improvement is calculated through equation (27).

$$\%\ \textbf{instances with improvement} = \frac{\textbf{\# improved instances}}{\textbf{\# total instances}} \qquad (27)$$

Table 8: Results summary for part 1

Scenarios	% instances with improvement	
	Total	$RR = 0, RV = 0$
$\boldsymbol{P1}$ vs. $\boldsymbol{P3}$	83.1%	4.6%
$\boldsymbol{P1}$ vs. $\boldsymbol{P2}$	69.3%	0
$\boldsymbol{P2}$ vs. $\boldsymbol{P3}$	61.7%	4.6%

Results for part 1 are summarized in Table 8. In total, the majority of problem instances have makespan improvement: 83.1% for $\boldsymbol{P1}$ vs. $\boldsymbol{P3}$, 69.3% for $\boldsymbol{P1}$ vs. $\boldsymbol{P2}$, and 61.7% for $\boldsymbol{P2}$ vs. $\boldsymbol{P3}$. However when resource limits are constant (Table 8, $RR = 0$ and $RV = 0$), only 4.6% instances have a makespan difference for $\boldsymbol{P1}$ vs. $\boldsymbol{P3}$ and $\boldsymbol{P2}$ vs. $\boldsymbol{P3}$. No instance has makespan improvement from $\boldsymbol{P1}$ to $\boldsymbol{P2}$ since non-preemptive activity splitting does not exist when resource limits are constant (Proposition 2). Instances have makespan improvement from $\boldsymbol{P1}$ to $\boldsymbol{P3}$ and $\boldsymbol{P2}$ to $\boldsymbol{P3}$ because of preemption.

5.9 Experiment part 2: Magnitude of makespan improvement

In part 2, we are interested in finding out the magnitude of makespan improvement and significant factors between different problem settings. Equations (28) and (29) define the quantity of makespan improvement between $\boldsymbol{P1}$ to $\boldsymbol{P3}$ and $\boldsymbol{P1}$ to $\boldsymbol{P2}$, respectively. These definitions follow the makespan improvement definition in Buddhakulsomsiri and Kim (2006).

$$\%\ \mathbf{improvement}_{\boldsymbol{P1-P3}} = \frac{(\mathbf{makespan}_{\boldsymbol{P1}} - \mathbf{makespan}_{\boldsymbol{P3}})}{\mathbf{makespan}_{\boldsymbol{P1}}} \qquad (28)$$

$$\%\ \mathbf{improvement}_{\boldsymbol{P1-P2}} = \frac{(\mathbf{makespan}_{\boldsymbol{P1}} - \mathbf{makespan}_{\boldsymbol{P2}})}{\mathbf{makespan}_{\boldsymbol{P1}}} \qquad (29)$$

For the comparison between $\boldsymbol{P2}$ and $\boldsymbol{P3}$, there are two possible measurement criteria. Equation (30) uses $\boldsymbol{P1}$ as the denominator in order to match the magnitude of the other two as well as satisfying equation (32). The other criteria (31) use $\boldsymbol{P2}$ as the denominator to measure the quantity of makespan improvement between $\boldsymbol{P2}$ and $\boldsymbol{P3}$.

$$\%\ \mathbf{improvement}_{P2-P3(1)} = \frac{(\mathbf{makespan}_{P2} - \mathbf{makespan}_{P3})}{\mathbf{makespan}_{P1}} \quad (30)$$

$$\%\ \mathbf{improvement}_{P2-P3(2)} = \frac{(\mathbf{makespan}_{P2} - \mathbf{makespan}_{P3})}{\mathbf{makespan}_{P2}} \quad (31)$$

$$\%\ \mathbf{improvement}_{P1-P3} = \%\ \mathbf{improvement}_{P1-P2} + \%\ \mathbf{improvement}_{P2-P3(1)} \quad (32)$$

Table 9 provides the overall quantity of makespan improvement. There is a 18.8% makespan improvement from $\boldsymbol{P1}$ to $\boldsymbol{P3}$ and a large portion (14.0% / 18.8% = 74.3%) of these improvements are because of non-preemptive activity splitting from $\boldsymbol{P1}$ to $\boldsymbol{P2}$, while a relatively much smaller portion (4.8% / 18.8% = 25.7%) is because of preemption from $\boldsymbol{P2}$ to $\boldsymbol{P3}$. However, in the case of constant resource limits and no resource vacations (at Table 9, $RR = 0$ and $RV = 0$), the makespan improvement is very limited. There is only 0.4% makespan improvement from $\boldsymbol{P1}$ to $\boldsymbol{P3}$ and $\boldsymbol{P2}$ to $\boldsymbol{P3}$ (1). Compared to the $\boldsymbol{P1}$ vs. $\boldsymbol{P3}$ with $RR = 0$ and $RV = 0$ experiment conducted in Patterson (1984), they have slightly higher makespan improvement (0.7%). This indicates that preemption provides limited benefits for makespan improvement when resource limits are constant and there are no resource vacations. Based on Proposition 2, makespan improvement from $\boldsymbol{P1}$ to $\boldsymbol{P2}$ is 0 when resource limits are constant.

Table 9: Results summary for part 2

Scenarios	% improvement	
	Total	$RR = 0, RV = 0$
$\boldsymbol{P1}$ vs. $\boldsymbol{P3}$	18.8%	0.4%
$\boldsymbol{P1}$ vs. $\boldsymbol{P2}$	14.0%	0
$\boldsymbol{P2}$ vs. $\boldsymbol{P3}$ (1)	4.8%	0.4%

Figure 8 provides a histogram for the magnitude of makespan improvement. The X-axis is the percentage of improvement from 0.0% to 100.0% while the Y-axis is the number of instances. For most of the 1538 tested instances, the magnitudes of makespan improvement are below 40%. There are limited instances for $\%\,\text{improvement}_{P1-P3}$ higher than 40% and they are all because of non-preemptive activity splitting from $\boldsymbol{P1}$ to $\boldsymbol{P2}$. The magnitude of preemption-caused makespan improvement from $\boldsymbol{P2}$ to $\boldsymbol{P3}$ is low since no instance has higher than 40% makespan improvement.

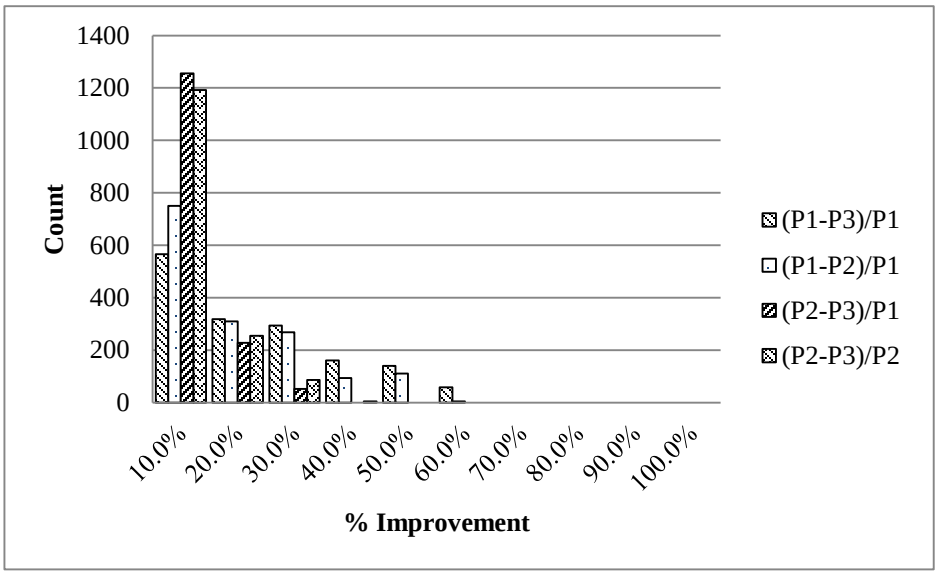

Figure 8: Histogram for the magnitude of makespan improvement

In order to study the relative difficulty of different problem parameter settings and performance of heuristics-based initial solutions, CPU time for each instance run is captured. Table 10 provides the average CPU time comparison between runs with or without heuristics-based initial solutions. It is observed that heuristic-based initial

solutions can help reduce CPU time for all three types of problem settings.

Table 10: CPU study for heuristic performance

CPU (sec.)	no heuristics	with heuristics
***P*1**	34.4	33.3
***P*2**	41.3	40.0
***P*3**	705.5	702.0

Figure 9 provides a histogram view of how initial solutions impact each test run. On average, initial solutions reduce CPU time by about 1-2 seconds. The majority of test runs do not show significant CPU time reduction (run with or without initial solutions only impact CPU time by -5 seconds to + 5 seconds). However, there are still a noticeable number of test runs that initial solutions can help in reducing CPU time by more than 10 seconds, 20 seconds, or more than 600 seconds.

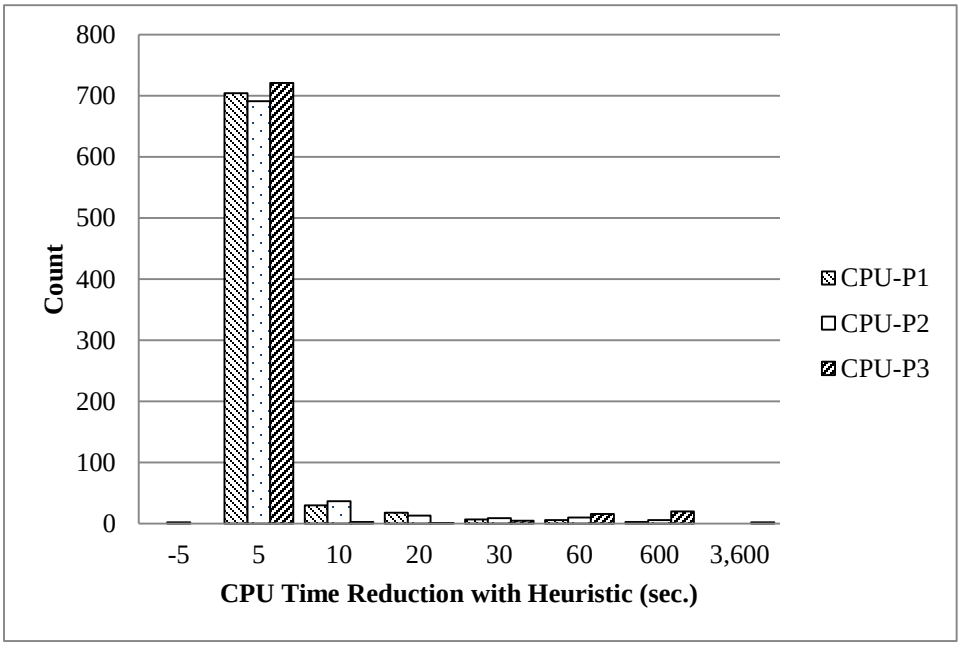

Figure 9: Histogram for CPU time reduction with heuristics

Possible explanations are: first, generating heuristic-based initial solutions is fast (less than 1-2 second), so running with initial solutions will not penalize the overall CPU time; second, the precedence tree-based branch-and-bound algorithm is a depth-first search enumerating scheme and returns feasible solutions as soon as the algorithm finds them. If these feasible solutions are close to the initial solutions found by heuristics, runs without initial solutions will perform similarly runs with initial solutions. However in some scenarios where branch-and-bound cannot find a competing solution fast enough, the initial solution can help in bounding a number of precedence tree branches.

In the majority of scenarios, generating heuristics-based initial solutions will not increase nor decrease the overall CPU time. However, their use can significantly reduce the overall solution time for some problem instances.

Table 11: Effect tests of factors

Prob>$\|t\|$	$\|N\|$	RSR	RFR	RR	RV	RD	$RSR \cdot RR$	$RFR \cdot RR$	$RR \cdot RV$
$\boldsymbol{P1}$ vs. $\boldsymbol{P3}$	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001
$\boldsymbol{P1}$ vs. $\boldsymbol{P2}$	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001
$\boldsymbol{P2}$ vs. $\boldsymbol{P3}$ (2)	<.001	<.001	<.001	<.001	**0.094**	<.001	<.001	<.001	<.001

Regression analysis is conducted to specify what factors are significant in affecting the makespan difference between $\boldsymbol{P1}$ to $\boldsymbol{P3}$, $\boldsymbol{P1}$ to $\boldsymbol{P2}$, and $\boldsymbol{P2}$ to $\boldsymbol{P3}$ (2). Basic statistical assumptions for regression analysis such as normality and homogeneity of variance are checked and satisfied.

$|N|$ is strongly significant for all three comparisons. Since makespan differences are measured on relative portion basis instead of absolute number of time units, it is more likely to find a high "percent" of makespan difference between $\boldsymbol{P1}$, $\boldsymbol{P2}$ and $\boldsymbol{P3}$ at problem instances with less activities than more activities. For example, instances with 10 time units makespan difference between different problem settings out of total makespan of 100 time units are more likely to be found than instances with 100 time units makespan difference for total makespan of 1000 time units. However, since problem instances tested in the research are from the benchmark problem instance generator ProGen, the observation here could be just reflecting how ProGen is setup instead of other practical project scheduling instances. Future research can focus on verify if $|N|$ is still significant in identifying makespan difference among three problem settings on practical problem instances.

Factors that relate to resource requirements (RSR, RFR, RR, $RSR * RR$, $RFR * RR$) are strongly significant for all three comparisons. In other words, resource tightness, resource requirements and time-varying resource constraints all significantly affect activity splitting, non-preemptive activity splitting, and preemption. The $RSR * RR$ interaction term is strongly significant since when renewable resources are tight and resource constraints are time-varying, it is more likely to expect large differences between $\boldsymbol{P1}$ and $\boldsymbol{P3}$, $\boldsymbol{P1}$ and $\boldsymbol{P2}$, or $\boldsymbol{P2}$ and $\boldsymbol{P3}$.

RV is significant for the first two experiments since both $\boldsymbol{P2}$ and $\boldsymbol{P3}$ can split

activities when there are resource vacations; however, RV is not significant for $\boldsymbol{P2}$ vs. $\boldsymbol{P3}$ since both $\boldsymbol{P2}$ and $\boldsymbol{P3}$ have "equal" advantage on makespan reduction regarding resource vacations.

RD factor is significant for all three experiments since the tightness of ready times and due dates determines when each activity can be scheduled and thus impacts the makespan of tested instances. Also, for tight RD factor, more $\boldsymbol{P1}$ instances are not able to find feasible solutions.

6. Conclusion and Future Research

In this research, we distinguished the differences between preemption, activity splitting, and non-preemptive activity splitting in project scheduling. A new type of problem $\boldsymbol{P2}$ (RCPSP with non-preemptive activity splitting) was identified to model real-world project scheduling challenges where resource limits are time-varying and there are also resource vacations. Comparison experiments were conducted in this research to study what parameter factors impact the makespan difference from $\boldsymbol{P1}$ to $\boldsymbol{P3}$, $\boldsymbol{P1}$ to $\boldsymbol{P2}$, and from $\boldsymbol{P2}$ to $\boldsymbol{P3}$. With resource vacations and time-varying resource constraints, there is a significant makespan improvement when comparing $\boldsymbol{P1}$ to $\boldsymbol{P3}$ - most of the makespan reduction occurs during the transition from $\boldsymbol{P1}$to $\boldsymbol{P2}$. The tighter resource limits and higher time-varying resource limits become, and tighter activity ready times and due dates become, the larger the makespan difference is between $\boldsymbol{P1}$ vs. $\boldsymbol{P3}$,

$\boldsymbol{P1}$ vs. $\boldsymbol{P2}$, and $\boldsymbol{P2}$ vs. $\boldsymbol{P3}$. However, resource vacations do not generally lead to significant makespan improvements between $\boldsymbol{P2}$ and $\boldsymbol{P3}$.

Even though our problem instances only have 12 or 16 jobs, two renewable resources, two non-renewable resources and three alternative processing modes, many tested instances cannot find optimal solution within the 1 hour CPU limit time. Thus, the natural next step of our research agenda is to study simple heuristics and meta-heuristics for solving for medium (10-50 activities), large (50-100), and practical (>500 activities) size problem instances. The proposing of priority rule-based simple heuristics already shows great advantage in computational time (< 1 sec.). An on-going research effort is underway that focuses on simple heuristics and meta-heuristics.

Appendix. Modified ProGen

Step 1: Set the NC, RSR, RFR, RSN and RFN levels.

Step 2: Generate problem instances from ProGen. A detailed description of ProGen can be found in Kolisch *et al.*, (1995) and Kolisch (1996). The ProGen generator can be downloaded from the PSPLIB site: http://129.187.106.231/psplib/.

Step 3: Retrieve the resource limit U'_{k0} for each renewable resource k in the generated instance.

Step 4: Specify the RR level.

Step 5: Randomly generate a time-varying resource profile based on the following:

$$U_{kt} \sim \text{uniform}\left(U'_{k0}(1-RR), U'_{k0}(1+RR)\right)$$

Even with the same RR level ($RR \neq 0$) of the original instance from PSPLIB and ProGen, different resource profiles can be generated. In order to evaluate the variance in resource profile generation, we generate two resource profiles for each original problem instance as duplicate measurements (Montgomery, 2008).

Step 6: Specify the RV level.

- if no resource vacation is considered, set $RV = 0$ and stop.
- if resource vacation is considered, set $RV = 1$and go to step 7.

Step 7: Generate a random number rn between $[0, 1)$ and use the mode function (MOD) to specify whether a time period is weekend (resource vacation).

- If $rn < 0.5$, set $\{U_{kt} = 0 | MOD(t, 14) = 0, \forall t = 1,2, \dots, T\}$
- if $rn \geq 0.5$, set $\{U_{kt} = 0 | MOD(t, 14) = 7, \forall t = 1,2, \dots, T\}$

Step 8: Generate activity ready times (rad_j) and due dates (due_j) for each activity $j$as follows.

Loose ready times and due dates:

$$rad_j \sim \text{uniform}(0, 5\% \cdot T)$$

$$due_j \sim \text{uniform}(95\% \cdot T, T)$$

Tight ready times and due dates:

$$rad_j \sim \text{uniform}(0, 10\% \cdot T)$$

$$due_j \sim \text{uniform}(90\% \cdot T, T)$$

CHAPTER 3 HEURISTIC-BASED SCHEDULING ALGORITHMS WITH DECOMPOSITION FOR PRACTICAL PROJECT SCHEDULING PROBLEMS IN SEMICONDUCTOR MANUFACTURING

1. Introduction

The process of "ramping up" a semiconductor wafer fabrication facility is a challenging endeavor. Depending on capacity, a state-of-the-art 300mm wafer fab can costs from $3 billion USD (Chien and Zheng (2012), Chasey and Pindukuri (2012)) to $10 billion USD (Ibrahim, Chik and Hashim, 2014). The vast majority of this investment procures over 1,000 pieces of capital equipment that need to be installed and qualified ("Install/Qual") for wafer production. The timing of the Install/Qual process is critical since it represents the time period between equipment delivery and product release-to-market. Shortening the Install/Qual process can defer capacity decisions to lower the risk of demand-capacity mismatch.

Practical limitations in the Install/Qual process make the project scheduling problem nontrivial. First, both renewable resources (e.g. technicians, testing equipment) and non-renewable resources (e.g. project budget, floor space) are constrained. Secondly, working calendars can differ for different types of renewable resources (e.g. 4 days per week @10 hours per day vs. 5 days per week @8 hours per day). Even for a given renewable resource, the total available resources per working day can vary, as workers

take vacations and/or testing machines break down. Next, each activity may have multiple alternative processing modes. For example, a piece of equipment could be installed by three senior and one junior technician for a total cost of $20,000 in six working days. Alternately, one senior and three junior technicians can complete the same installation in eight working days for a total cost of $16,000.

While each activity is allowed to pause when resources are temporarily not available, the activity cannot be preempted by other activities. The size of this practically-motivated Install/Qual process containing over 1,000 pieces of equipment and multiple types of resources is much bigger than typical project scheduling instances studied in the literature. Currently, simple rules based on historical data ("tribal knowledge") are used to solve the Install/Qual scheduling problem in practice. Our goal is to determine the latest start time of the Install/Qual process subject to resource constraints, precedence relationships, and activity due dates so that capacity planning decisions can be made as late as possible.

The main contribution of this paper is to propose and compare heuristic-based methodologies to solve the Install/Qual scheduling problem in a reasonable amount of computation time. The methodologies under study include 1) a modified exact method via the use of the CPLEX solver, 2) priority rule-based simple heuristics, 3) simulated annealing, and 4) a modified random key-based genetic algorithm (modified RKGA). A project decomposition mechanism is studied for practical size problem instances. The

remaining sections of this paper are organized as follows. In Section 2, the Install/Qual scheduling problem is briefly described and formally modeled. In Section 3, related research efforts are reviewed and our problem solving methodologies are discussed in detail. An overview of our computational experiments is presented in Section 4, followed by an analysis of the results in Section 5. Finally, research conclusions and suggestions for future research directions are presented in Section 6.

2. Problem Statement

In Cheng et al. (2014), the Install/Qual scheduling problem is formulated as a multi-mode resource-constrained project scheduling problem (MRCPSP). A project network $G(N, A)$ contains a set of nodes N representing the activity set $\mathbb{N}$ and a set of directed arcs A representing the precedence relations among activities. Dummy nodes 0 and $|\mathbb{N}| + 1$ are added as super source and super sink nodes, respectively, to start and complete the project network. Both renewable resources $\mathbb{R}^r$ and non-renewable resources $\mathbb{R}^n$ are considered. At each time period t, the availability of a renewable resource k is restricted to be in the range $[0, U_{kt}]$ ("resource profile"). The availability level of a non-renewable resource k is limited by the upper bound U_k throughout the entire project horizon $[0, T]$. Each activity j can be processed in multiple modes such that each mode $m \in Mod_j$ specifies duration p_j^m and r_{jk}^m amount of resource. An activity needs to be scheduled between the ready time rad_j and its due date due_j. EST_j (EFT_j), LST_j (LFT_j)

represent the earliest start (finish) time, latest start (finish) time of activity *j*, respectively. Activity splitting is only allowed when resources are not sufficient which is non-preemptive activity splitting according to the classification scheme introduced in Cheng et al. (2014).

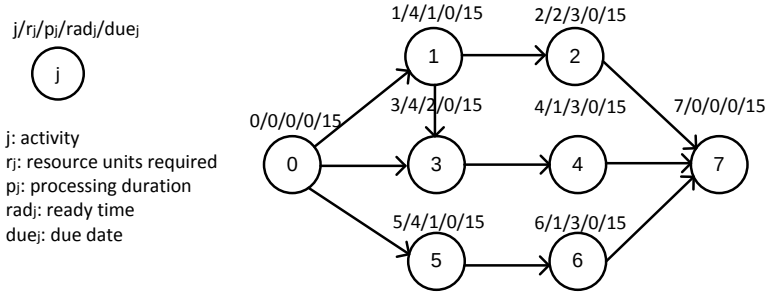

Figure 10: A Project Network Example

When resource limits are constant for all renewable resources and resource vacations are not allowed, forward and backward scheduling approaches are interchangeable by simply "sliding" the entire schedule in time. However, in the Install/Qual process as well as other similar situations where resource limits for renewable resources are time-varying and resource vacations are included, forward and backward schedules are different. An example instance is shown in Figure 10 with the objective to minimize project makespan. When the resource limit is constant at 4 resource units (Figure 11 a, c, e), schedule (a) is an optimal schedule found by using the backward scheduling approach (activities are scheduled as late in time as possible). While schedule (c) is an optimal schedule found by the forward scheduling approach

(activities are scheduled as early in time as possible). By simply sliding to its end date, forward schedule (c) is converted into an alternate optimal backward schedule (e) with only the forward scheduling approach and the sliding mechanism. However, when the resource limit is time-varying as in (b), an optimal forward schedule (d) cannot use the "slide" mechanism to become an optimal backward schedule. In fact, the slide backward schedule (f) is much worse compared to the backward schedule (b).

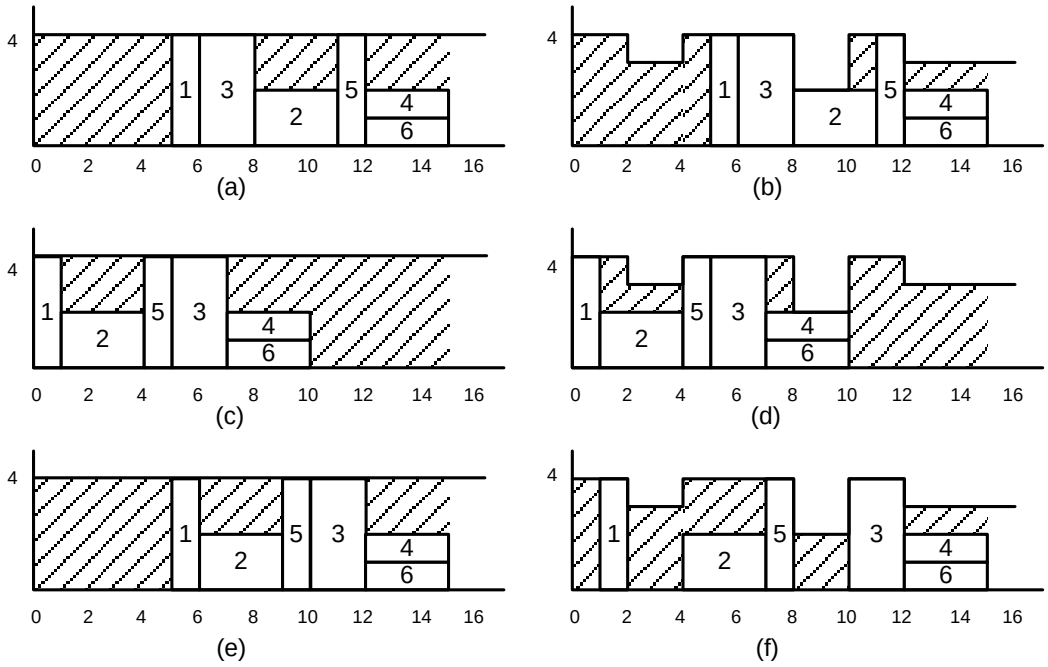

Figure 11: Forward vs. Backward Schedules

To implement the backward scheduling approach, the original project network $G(N, A)$ is converted to a "backward" network $G'(N', A')$ by reversing the direction of each arc. Furthermore, for each activity in the project, the ready time and due date are

replaced by each other. To modify the mathematical formulation in Cheng et al. (2014) to handle backward scheduling, the objective function needs to be changed from minimizing the start time of the dummy finish activity (which is equivalent to the completion time of the last actual activity) to maximizing the completion time of the dummy start activity (which is equivalent to the start time of the first actual activity).

3. Literature Review and Methodology

This research work is the follow on related to Cheng et al. (2014) where a detailed literature review on resource-constrained project scheduling problem (RCPSP) and various extensions can be found. The main extensions include (1) both renewable and non-renewable resources, (2) multiple activity processing modes, (3) time-varying resource constraints, (4) non-preemptive activity splitting and (5) activity ready times and due dates. To formulate the Install/Qual scheduling problem, Cheng et al. (2014) extend the basic RCPSP model to include (1), (2), (3) and (4). Reviews on extensions (1), (2) and (3) can be found at Kolisch and Padman (2001), Hartmann and Briskorn (2010), and Węglarz et al. (2011). Extension (4) is first discussed in Cheng et al. (2014) where activities are only allowed to split when there are insufficient resources available but activity preemption is prohibited. Extension (5) is included in this work to handle activity ready times and due dates which is a common concept in the machine scheduling literature (Pinedo, 2008). In the project scheduling literature, generalized precedence

constraints can be indirectly used to model activity ready times and deadlines (De Reyck and Herroelen, 1999).

A modified precedence tree-based branch-and-bound algorithm is proposed in Cheng et al. (2014) as an exact approach. However, computational limitations are observed even for small size academic problem instances. Problem instances with 10 activities, 2 renewable resources, 2 non-renewable resources and 3 alternative modes might take more than 10 hours to find and prove optimal solutions. As pointed out by Węglarz et al. (2011), it is still computationally intractable to solve MRCPSP instances with more than 20 activities. Thus, this research focus on heuristic-based algorithms with four categories of heuristics (Table 12) that are reviewed and discussed in the chapter.

Table 12: Summary of Heuristic Methodologies

Category	Methodology
Simple heuristics	Modified priority rule-based simple heuristics
Meta heuristics	Modified random key based genetic algorithm, simulated annealing
Exact solution-based	Modified ILOG-CPLEX approach
Decomposition	Time window-based project decomposition

Before discussing various heuristics approaches, solution representation and schedule generation schemes need to be defined. Kolisch (1999) and Alcaraz and Maroto (2001) summarize four different types of solution representations: activity list (AL), priority rule (PR), random key (RK) and shift vector (SV) and point out that the AL and the RK representations are the most widely used and most efficient in many situations.

Debels et al. (2006) illustrate that the RK representation can lead to promising result if the topological ordering (TO) (Valls et al., 1999) is applied. Further, the RK representation is relatively easy to implement in most cases and facilitates maintaining solution feasibility when the crossover operation is used in a genetic algorithm (Mendes et al., 2009). RK encoding is selected in this work such that a schedule can be coded as $SOL = \{MOD, RK\}$ in which $RK = \{RK_j | RK_j \in (0,1), j \in \mathbb{N}\}$ is a vector of random keys and $MOD = \{m_j | m_j \in Mod_j, j \in \mathbb{N}\}$ is a vector of mode assignments.

A Schedule Generation Scheme (SGS) is the process of constructing a schedule from an algorithm or a heuristic and the most widely studied SGSs are parallel and serial SGSs. Parallel SGSs schedule multiple activities at a time and increment time while serial SGSs schedule one activity at a time. As shown in Sprecher et al. (1995) and Kolisch (1996a), the search space of a parallel SGS might not always include the optimal solution and thus the optimal solution cannot be found in some cases.

In practice, forward SGSs are more common where activities try to be scheduled early in time and the objective is to finish the entire project as early as possible. On the contrary, to schedule a project from some pre-defined "finish time" to "start time" with the decrease of time is the backward SGS approach. In order to apply a backward SGS, the ready time and due date for each activity in the original problem instance are swapped and a new precedence network $G'(N', A')$ is created by reversing the original network $G(N, A)$ by setting $rad'_j = due_j$, $due'_j = rad_j$, $N' = N$, and replace $arc(i, j) \in$

A with $arc(j,i) \in A'$. Li and Willis (1992) propose a local search scheduling technique that iteratively schedules activities forwardly and backwardly in time to reduce schedule makespan. As shown in Hartmann and Kolisch (2000), the computational effort of one forward-backward run is the same with executing one SGS which is $O(n^2K)$ where n and K represent the number of non-dummy activities and the number of renewable resources, respectively. As pointed out by Özdamar and Ulusoy (1996) and Valls et al. (2005), the forward-backward iterative scheduling technique generally terminates after three or four consecutive passes. For this reason and the fact that rule-based heuristics are fast scheduling methods, it is better to combine the iterative scheduling approach with a meta-heuristic such as genetic algorithm (GA), simulated annealing (SA) or Tabu search (TS). Successful examples of combining GA with the iterative forward-backward approach can be found in Özdamar (1999), Alcaraz and Maroto (2001), Alcaraz et al. (2003), Debels and Vanhoucke (2005), Debels et al. (2006), Lova et al. (2009), Peteghem and Vanhoucke (2010) and Zamani (2011).

3.1 Priority Rule-Based Simple Heuristics

In the MRCPSP literature, priority rule-based heuristics represent a type of simple heuristic that combine mode and activity selection rules with SGSs to generate schedules. Mode selection rules determine the processing mode among multiple modes for each activity and activity selection rules specify the relative priority for each activity when

being selected to process. The most commonly adopted mode selection rules include shortest duration mode (SDM), least total renewable resource usage (LTRU_R), and least total non-renewable resource usage (LTRU_N). Applications of these rules can be found in Boctor (1996) and Buddhakulsomsiri and Kim (2007). This research integrates duration and non-renewable resource to propose a shortest duration and least non-renewable resource usage rule (SD-LTRU_N) $\left\{m \in Mod_j \middle| \sum_{k\in\mathbb{R}^n}\left(r_{jk}^m * p_j^m\right) = \min_{\forall l\in Mod_j} \sum_{k\in\mathbb{R}^n}\left(r_{jk}^l * p_j^l\right)\right\}$. Common activity selection rules are shortest processing time (SPT), longest processing time (LPT), minimum slackness (MSLK), minimum latest start time (MLST), minimum latest finish time (MLFT), maximum total successors (MTS) and greatest rank positional weight (GRPW). Application of these rules can be found in Helgeson and Birnie (1961), Davis and Patterson (1975), Alvarez-Valdes and Tamarit (1989), Kolisch et al. (1995), Kolisch (1996b), Lova et al. (2006) and Buddhakulsomsiri and Kim (2007). This research adds earliest ready time (ERT) $\left\{j \in \mathbb{N} | rad_j = \max_{l\in\mathbb{N}} rad_l\right\}$, earliest due date (EDD) $\left\{j \in \mathbb{N} | due_j = \min_{l\in\mathbb{N}} due_l\right\}$, and a modified minimum slack (MMSLK) rule $\left\{j \in \mathbb{N} \middle| \left(LST_j - EST_j\right)/p_j^m = \min_{l\in\mathbb{N}}\left((LST_l - EST_l)/p_l^m\right)\right\}$ into the comparison. Cheng et al. (2014) discussed how to calculate these values when resources are considered. Overall there are a total of 40 priority rule-based simple heuristics each of which is a combination of one mode selection rule (SDM, LTRU_R, LTRU_N, SD-LTRU_N) and one activity priority rule (SPT, LPT, ERT, EDD, MSLK, MLST, MLFT, MTS, GRPW, MMSLK). The first algorithm that we propose and

examine – “Best Simple” algorithm combines all 40 simple heuristics and returns the best solution as the overall solution of the algorithm. Also, they are used as the starting solutions for other heuristics which are discussed in the next several subsections.

3.2 Genetic Algorithm

The genetic algorithm (GA) is a well-studied meta-heuristic first proposed by Holland (1975). The application of GA has later been shown to be efficient among various meta-heuristic solution techniques for NP-hard combinatorial optimization problems (see Gen and Cheng (2000); Gen et al. (2008)). GA maintains a solution population with a number of candidate individuals (chromosomes) over many generations and the fitness value of each individual chromosome is evaluated and fitter individuals are more likely to be selected to produce offspring for the next generation. Proposed by Norman and Bean (1995), the random key-based genetic algorithm (RKGA) has been shown to be easy to implement with powerful search capability. RKGA for RCPSP can be found in research by Debels and Vanhoucke (2007) and Mendes et al. (2009). Both of these are single mode RCPSPs and non-renewable resources are not included. For MRCPSP, Okada et al. (2010) consider multiple modes and applied the idea of using a separate random key vector to represent the processing mode for each activity. However, activity splitting is not allowed and renewable resources have constant resource profiles in Okada et al. (2010).

In this research, initial solutions for GA are generated from both randomly and uses the priority rule-based simple heuristics to define the initial population with size Pop_Size. Therefore, our GA can benefit from good initial starting points but also maintain the diversity of initial solutions through randomly generated solutions. As pointed out by Kolisch and Drexl (1997) and in many other research efforts, the problem of finding a feasible mode assignment for MRCPSP with more than one type of non-renewable resource is NP-complete since it is essentially a knapsack problem. Thus, it is non-trivial to find an efficient way guaranteed to modify an infeasible mode assignment to a feasible one. Based on a local search procedure in Hartmann (2001), a *mode repair operation* is developed to improve an infeasible mode toward a feasible direction until it reaches a feasible mode assignment or remains infeasible after a certain number of searches. A penalty value is introduced to measure the level of infeasibility for infeasible solutions in GA. The penalty value PE_i^{NR} for a solution i with regard to non-renewable resources is defined as $PE_i^{NR} = \mathbb{C}_i^{NR} \cdot NF^{NR}$, where $\mathbb{C}_i^{NR}$ is the per unit cost for non-feasibility value NF^{NR} for non-renewable resources. There can also be infeasibility due to the activity ready times and due dates. In the backward scheduling approach, each activity is scheduled in a backward manner from its due date. It is an infeasible solution if it violates the ready time. A penalty value is calculated as $PE_i^{RD} = \mathbb{C}_i^{RD} \cdot NF^{RD}$ where $\mathbb{C}_i^{RD}$ represents the per unit cost for ready time and due date value NF^{RD} for each activity. The fitness value for a solution i in the modified RKGA is calculated through a fitness

function that includes the completion time of the dummy start time (C_{0_i}), a penalty value regarding non-renewable resources (PE_i^{NR}) and a penalty value regarding ready time and due date (PE_i^{RD}) as $FV_i = C_{0_i} + PE_i^{NR} + PE_i^{RD}$. Compared to the priority rule-based simple heuristics, GA can start from solutions that are still infeasible after the mode repair operation. With the help of the penalty function, the GA ranks these infeasible solutions along with the feasible solutions. Constant values $\mathbb{C}_i^{NR}$ and $\mathbb{C}_i^{RD}$ are set so that they are much larger than C_{0_i} value ($\mathbb{C}_i^{NR} \gg T$, $\mathbb{C}_i^{RD} \gg T$) to penalize and eventually avoid infeasible solutions.

The elitist reproduction process is accomplished by maintaining a portion of the best individuals into the next generation to make sure the genetic algorithm almost monotonically improves solution quality. A parameter $P_{ER} \in [0, 1]$ is defined as the portion of the elitist solutions in the population. For the selection mechanism, two chromosomes are selected randomly with replacement from the previous generation's population as parents for the crossover and mutation operations. The worst portion of the previous generation is included since these solutions may be "bad" because of infeasibility but still be "good" candidates in terms of project makespan. These solutions can potentially lead to very promising schedules. The two parent chromosomes and two child chromosomes are evaluated based on the fitness function. Two of the best chromosomes enter the next generation population. This process is repeated multiple times until they reach the candidate number chromosomes for the next generation

population $(1 - P_{ER}) * Pop\_Size$. These candidates are sorted and the last portion of them is replaced by randomly generated immigrants.

Crossover is a basic GA operator that selects and combines two chromosome members (parents) to produce new chromosomes with the hope that new chromosomes can inherit good attributes from their parents and hence be better solutions for the next generation population. Traditional single point and two point crossovers randomly select one or two crossover points within a chromosome and interchange a segment of genes on the two parent chromosomes. Examples of those can be found in Debels and Vanhoucke (2005), Debels et al. (2006) and Debels and Vanhoucke (2007). Uniform crossover generalizes the point crossover and essentially makes every gene a potential crossover point so that it adds flexibility on building chromosomes on the gene level rather than chromosome segment. However, the additional flexibility in uniform crossover suffers the possibility of destroying a good solution structure (Sivanandam and Deepa, 2007). Norman and Bean (1995) discuss the Bernoulli crossover which is also called the parameterized uniform crossover in Spears et al. (1993). Bernoulli crossover has one parameter that controls the amount of disruption during recombination without having bias towards the length of the representation used (Norman and Bean, 1995). Since the random key representation includes both the mode assignment vector and the activity priority vector, both the Bernoulli crossover and the two-point crossover are adopted. Bernoulli crossover is applied on the mode assignment chromosome and two-point

crossover is applied on the activity priority chromosome. The use of the Bernoulli crossover for mode assignment chromosome enables parent chromosomes to contribute to the individual gene level, because mode assignments of activities are less dependent on each other (with the exception of the total available non-renewable resources). However, for the activity priority vector where the priority value of each activity depends on priority values of the other activities, it makes more sense to maintain a chromosome segment level by using the two-point crossover.

The mutation operation is implemented to avoid premature convergence. After a gene is randomly selected for mutation, a new mode assignment (not equal to the original one) is randomly selected to replace the current mode assignment and the activity priority key is replaced by 1 minus the original value. These can improve the effectiveness of the mutation process since the new solution randomly generated from a new mode and activity key has a high probability of converging to essentially the optimal schedule. For instance, if an activity has 2 alternative modes and the current mode is mode 1, mode 1 can be selected again with 50% of probability if we just randomly select a mode for this activity.

The purpose of diversification in a GA is to escape from a premature convergence and avoid homogeneous offspring solutions. In this work, diversification is accomplished by introducing an immigration operator (IO). The IO randomly generates a number of new solutions as immigrants and replaces the worst portion of the candidate solutions

obtained from crossover and mutation. The IO operator can introduce new (and possibly better) search directions into the population at each generation. A parameter $P_{IO} \in [0, 1]$ is defined to control the number of new immigrants. Based on the definition, the worst $P_{IO} * Pop\_Size$ solutions of new candidate offspring are replaced by immigrants for the next generation population. The modified RKGA in this work stops after *Num_Iter* number of generations. Terminating the modified RKGA after a pre-determined number of iterations makes it easy to control the computational effort. It is very helpful to provide a fair computational comparison. The number of iterations *Num_Iter* is set by preliminary experiments to avoid early or late termination.

3.3 Dynamic T Approach

To evaluate the performance of proposed heuristics, it would be ideal to know the optimal solutions for a set of test problem instances. However, due to the known NP-hardness of the problem, exact methods for medium to large size problem instances might take hours or possibly even days of computational time. In this research, we rely on the ILOG-CPLEX solver but with a modified MIP formulation and call this method the Dynamic T (Dyn T) approach. The idea is as follows. In the MIP formulation, the number of decision variables x_{jt}^{m} (whether activity j is scheduled at time t with mode m) and the number of constraints with x_{jt}^{m} or time index t depend on the project horizon T. The smallest value T^* is the optimal solution which is also the shortest makespan. While

$T \ll T^*$, the problem instance is infeasible and it is relatively faster for CPLEX to prove infeasibility; however when $T \gg T^*$, the problem instance is feasible but requires a long computational time since the number of constraints and decision variables are large. So the algorithm start from a relatively small lower bound value for T that it can prove infeasibility of the instance and then iteratively increase the T value with a dynamic step size until a feasible solutions is found. The pseudo code for this algorithm is as follows.

Dyn T Algorithm

Step 1: Solve the LP relaxation and obtain the LP relaxed solution: LP_{relax}, and also define an upper bound for project horizon as T_{max}. At iteration k, the T value can be calculated as

$$T_k = T_{k-1} + stepsize_k$$

Where the step size is calculated as: $stepsize_k = \min\{stepsize^0, (T_{max} - LP_{relax}) * sizefactor\}$, in which $stepsize^0$ is the minimum stepsize and $sizefactor$ is a value between 0 and 1 that determines the speed of $stepsize$ changes. In this research, $stepsize^0 = 2$ and $sizefactor = 0.8$ based on preliminary tests.

Step 2: Solve the MIP formulation using T_k. If the MIP instance is infeasible, update $k = k + 1$ and go back to the previous step to update T_k, else return the best found value as the solution.

The overall summary of related literature can be found in Table 13. Acronyms in the table include: Priority Rule-based heuristics (PR-H), Scatter search (SS), Genetic algorithm (GA), Bi-population GA (BP-GA), Hybrid GA (H-GA), Branch & Bound (B&B), Hybrid Scatter Search (H-SS), Particle Swarm Optimization / Particle Swarm (PS), Linear programming (LP), Branch & Cut (B&C), Random sampling (RS), Multi-pass heuristics (MP-H), Population-based heuristics (PB-H), Critical activity reordering (CAR), Activity list (AL), Random key (RK).

Table 13: Literature Review Summary

Citation	Resource	R/NR	Mode	Preem	Method	Direction	Representation	SGS	Dataset	Max Size
Li and Willis (1992)	const/vary	R/NR	single	P1	MP-H	F/B	AL	-	Own	25
Hartmann (1998)	const	R	single	P1	GA + PR-H	F/B	RK	serial	PSPLIB Patterson	60
Özdamar (1999)	const	R/NR	multi	P1	H-GA	F/B	-	parallel	PSPLIB Own	90
Alcaraz and Maroto (2001)	const	R	single	P1	GA	F/B	AL	serial	PSPLIB	120
Tormos and Lova (2001)	const	R	single	P1	Hybrid MP-H	F/B	-	serial parallel	PSPLIB	120
Nonobe and Ibaraki (2001)	vary	R/NR	multi	P1	TS	F	-	-	PSPLIB	30
Józefowska et al. (2001)	const	R/NR	multi	P1	SA	F	AL	serial	PSPLIB	30
Hartmann (2001)	const	R/NR	multi	P1	GA	F	AL	serial	PSPLIB	30
Bouleimen and Lecocq (2003)	const	R/NR	multi	P1	SA	F	AL	serial	PSPLIB Patterson	30
Alcaraz et al. (2003)	const	R/NR	multi	P1	GA	F/B	AL	serial	PSPLIB Boctor	100
Valls et al. (2003)	const	R	single	P1	CAR	F/B	-	serial	PSPLIB	120
Tormos and Lova (2003)	const	R	single	P1	MP-H + RS	F/B	AL	serial parallel	PSPLIB	120

Valls et al. (2004)	const	R	single	P1	PB-H	F	-	serial parallel	PSPLIB	120
Valls et al. (2005)	const	R	single	P1	-	F	-	serial parallel	PSPLIB	120
Debels and Vanhoucke (2005)	const	R	single	P1	BP-GA	F/B	AL	serial	PSPLIB	120
Buddhakulsomsiri and Kim (2006)	vary	R	multi	P1, P3	B&B	F	-	-	PSPLIB	13
Debels et al. (2006)	const	R	single	P1	SS	F	RK	serial	RanGen PSPLIB	120
Zhang et al. (2006)	const	R/NR	multi	P1	PS	F	Particle	serial	PSPLIB	20
Zhu et al. (2006)	vary	R/NR	multi	P1	BC	F	-	-	PSPLIB	30
Buddhakulsomsiri and Kim (2007)	vary	R	multi	P1, P3	PR-H	F	-	serial	PSPLIB	90
Debels and Vanhoucke (2007)	const	R	single	P1	BP-GA	F/B	RK	serial	PSPLIB own	120
Damay et al. (2007)	const	R	single	P1, P3	LP	F	-	-	PSPLIB	60
Jarboui et al. (2008)	const	R/NR	multi	P1	PS	F	Particle	-	PSPLIB	30
Ranjbar et al. (2008)	const	R/NR	multi	P1	H-SS	F	AL	serial	PSPLIB	20
Vanhoucke and Debels (2008)	const	R	single	P1, P3	B&B-H	F	-	serial	RanGen	20
Ballestin et al. (2008)	const	R	single	P1, P3	-	F	AL	serial	PSPLIB	120
Mendes et al. (2009)	const	R	single	P1	GA	F	RK	-	PSPLIB	120
Lova et al. (2009)	const	R/NR	multi	P1	H-GA	F/B	RK	serial parallel	PSPLIB	100

Peteghem and Vanhoucke (2010)	const	R/ NR	multi	P1, P3	BP-GA	F/B	RK	serial	RanGen PSPLIB	30
Okada et al. (2010)	const	R/ NR	multi	P1	GA	F	RK	serial	-	-
Gonçalves et al. (2010)	const	R	single	P1	GA	F/B	RK	serial	PSPLIB	120
Zamani (2011)	const	R	single	P1	H-GA	F/B	-	serial	PSPLIB	120
This research	**vary**	**R/ NR**	**multi**	**P2**	**GA + PR-H**	**B**	**RK**	**serial**	**PSPLIB own**	**1000**

3.4 Project Decomposition

Besides simple heuristics and the GA that are reviewed previously, project decomposition is also a popular heuristic approach, especially for large size problem instances. Payne (1995) and Lova and Tormos (2001) show that 80% ~ 90% of real world projects are multi-project problems that are either constrained by some common sharing resources or precedence relations. This motivates resource constrained ***multi-project*** scheduling problem (RCMPSP) as a branch of the project scheduling literature. Compared to RCPSP, RCMPSP is not as thoroughly studied due to the fact that generally RCMPSP can be solved using the single-project approach by merging all subprojects into a mega-project with one super-source node and one super-sink node. The single-project approach is easy to understand but suffers major drawbacks. One of the most obvious ones is that aggregating multiple projects yields very large problem instances which make the already difficult RCPSP even more difficult to solve (Chiu and Tsai, 1993).

Also, using the single-project approach may lose different emphasis (e.g. tardiness, cost) for each subproject and make independent analysis on each subproject difficult (Chiu and Tsai, 1993). In contrast, research like Serafini and Speranza (1991), Sprecher (2002) and Debels and Vanhoucke (2007) treat subprojects separately in RCMPSP (multi-project approach) or decompose a single project in RCPSP into subprojects. The multi-project approach is necessary in RCMPSP when each subproject has to be handled separately (e.g. each subproject has a different objective). Meanwhile, the decomposition approach for RCPSP instances is considered as a heuristic approach to break large size problem instances into smaller ones and then solved by exact methods, simple heuristics or meta-heuristics. As shown by Deckro et al. (1991), decomposition methods that rely on problem characteristics generally offer the most promising solution.

RCMPSP often assumes precedence constraints are defined only within jobs in each subproject (Krüger and Scholl, 2009) and only global linkage connects each subproject together (e.g. only connects to the dummy source or sink node at each project). Thus, it is intuitive to think for RCPSP with similar network structures such that the original network can be isolated into multiple sub networks and they are connected with some "inter-network" links can be solved using a decomposition approach with less impact of losing better solutions. The network complexity factor utilized in RCPSP benchmark problem instance generator ProGen (Kolisch et al. 1995) is measured by the average non-redundant arcs per node including dummy start and completion nodes. But

the definition does not imply the special “decomposable” network structure discussed before. The network decomposition approach in Sprecher (2002) determines the subproject first and then randomly generates precedence feasible sequences to assign activities into each subproject. Zamani (2004) use the simulated annealing technique to find a starting schedule and defines subprojects as activities in time windows which are defined as the time horizon between a randomly generated starting point and a time window length into the Gantt chart of a project. Palpant et al. (2004) combines large neighborhood search with project decomposition such that at each step of the algorithm, a sub component of the base solution is fixed while the others define a subproblem that is solved with a heuristic or an exact solution method. In Zamani (2011), initial solutions are generated by random sampling and decomposed into subprojects. Then subprojects are scheduled through exact methods and further refined by a genetic algorithm. It worth mentioning that all studies of network decomposition methods in Sprecher (2002), Zamani Reza (2004), Palpant et al. (2004) and Zamani (2011) are applied on single mode project networks without any non-renewable resources and none of these studies discuss what kind of network structure is better for decomposition.

Activities in a project scheduling instance are often constrained by some commonly shared resources. When generating the RCPSP benchmark problem instances with ProGen, Kolisch et al. (1995) uses a resource factor (RS) parameter applies to both renewable resources (RSR) and non-renewable resources (RSN). The RS parameter

represents the percent of resource types that each activity utilizes on average. For example a project network with total 2 types of renewable resources and 2 types of non-renewable resources. RS = 0.5 if on average each activities only requires one type of renewable resource and one type of non-renewable resource. Intuitively, a higher RS means activities are closely resource-connected and should be solved using the single-project approach. In contrast, projects with a lower RS can be solved using a decomposition approach with activities that share the same types of resources in the same subproject. However, since RS is an average value for all activities, it is not necessary that two different projects can be decomposed the same way even if their RS values are the same. Similarly, RCMPSP often assumes some common resources among subprojects. Confessore et al. (2007) consider multiple projects where each subproject has its own resources and they share one common resource. Krüger and Scholl (2009) assumes there are higher resource transfer penalty costs (e.g. setup time) when resources are being utilized by activities in different subprojects. More (less) common resources that are being shared by subprojects usually lead to closer (looser) relations among subprojects and intuitively a single- (multi-) project approach is more preferable.

Other than precedence and resource availability constraints, activity ready times and due dates also restrict when an activity can start or complete. Activities with overlap windows from the ready time to the due date should be scheduled at the same time since they are most likely to compete for limited resources. In contrast, activities with less time

window overlaps can be decomposed into separate subprojects without losing better solutions. Pritsker et al. (1969) add due dates and deadlines for the sink activities of each project in RCMPSP and Franck et al. (1997) consider a network of multiple projects with minimal and maximal time lags.

In summary, previous research on project decomposition in RCPSP and RCMPSP have focused on the project decomposition mechanisms but not on whether a project instance is better to be composed or not based on project characteristics such as project network, resources, ready time and due date and so on. Therefore, this research aims to propose a decomposability score (*decom_score*) based on project characteristics to guide researchers and practitioners when decomposition is recommended for a given RCPSP or RCMPSP instance. A Euclidean distance measurement is proposed such that the distance is measured from the modified earliest start time (EST) and latest finish time (LFT) window approach of Cheng et al. (2014). As an analogy if each activity is considered as a geographical location, the EST and LFT for the activity are considered as the latitude and longitude of that location. Then the *decom_score* is defined as the ratio of average distance among subnetworks and average distance among all activities. A detailed definition can be found in the algorithm description later in this paper.

In RCPSP with decomposition, the number of subprojects needs to balance the optimality of projects and computational efforts. Too few subprojects will not make a difference compared to solving the entire problem as one single project since each

subproject is still difficult to solve, while too many subprojects may restrict the solution space to be much smaller than the original problem and optimality will likely suffer. In this work, we rely on the solvability of each subproject and problem characteristic to decide the size of subprojects. After the size of subprojects is determined, each activity needs to be assigned to one subproject. This work tries to create a series of subprojects that all inter-subproject links only go one direction (precedence feasible sequence). Therefore, subprojects can be scheduled serially from the one that contains the dummy start activity to the last subproject with the dummy complete activity. The determination of which activity to select is based on the distances for potential activities from existing activities in a subproject. Palpant et al. (2004) compare several activity selection rules such as higher priority for activities on the critical path, immediate predecessors, contiguous predecessors and found out the best performing rule is the "block" rule that selects contiguous or parallel activities with existing activities in the subproject. This is similar to the method in this work that tries to select activities with smaller "distance", in other words, in the same "block".

Since each subproject is solved separately, the amount of non-renewable resources for each subproject needs to be determined. The more subprojects after decomposition, the more likely that non-renewable resources allocated for a subproject become too restrictive to keep potential better solutions. Since it is already NP-complete to find a feasible mode assignment for MRCPSP instances with more than one type of

non-renewable resource, it is not easy to determine the optimal non-renewable availability level for each subproject. This work proportionally allocates non-renewable resources to each subproject based on resources required for all activities in a subproject. Detailed pseudo code for the algorithm is as follows.

Project Decomposition

Step 1: Initialize a list of clusters $cl_k = \emptyset,\ k = 1, 2, \dots K$, where K is the maximum number of subprojects, $unassigned = \{0, 1, \dots, N\}$ as the set of nodes that have not been assigned to any cluster yet, pick the dummy source node 0, and cluster cl_1 as the current cluster, add node 0 to cluster: $cl_1 = cl_1 \cup \{0\}$, remove node from unassigned set: $unassigned = unassigned \backslash \{0\}$

Define the distance between two points i and j in clusters cl_k:

$$\text{Distance}(i,j) = \sqrt{\left(EST_i - EST_j\right)^2 + \left(LFT_i - LFT_j\right)^2}, \forall i, j \in cl_k$$

Define the *decom_score* as the average distance between all clusters cl_k

$$decom\_score = \frac{\text{average}\left\{distance\left(\bar{k}, \bar{l}\right)\right\}}{\text{average}\left\{distance(i,j)\right\}}, \forall k, l = 1, 2, \dots K, \forall i, j \in \{0, 1, \dots N\}$$

Step 2: Calculate the center of gravity point $\bar{k}$ $(\overline{x_k}, \overline{y_k})$ of the current cluster cl_k with coordinates:

$\overline{x_k} = \sum_{\forall i \in cl_k} x_i / |x_i|$, $\overline{y_k} = \sum_{\forall i \in cl_k} y_i / |y_i|$

Step 3: For any node $i \in cl_k$, for any predecessor node $j \in pred(i)$ that is unassigned, add node j into the cluster $cl_k = cl_k \cup \{j\}, \forall j \in pred(i), j \in unassigned$. Update the center of gravity point. Repeat step 3 until all predecessor nodes for each node in the current cluster are assigned, else, go to step 4

Step 4: When no predecessor nodes can be added to the current cluster: if $\forall i \in cl_k, \forall j \in pred(i)$, then $j \in cl_k$, check the stopping rule to see if need to explore other activities in $unassigned$. If no, stop the current cluster, add a new dummy finish node for the cluster and move to the next cluster $k = k + 1$, add a new dummy start node to the new cluster. If yes, go to step 5

Step 5: Pick a connected node i $(\exists m \in cl_k, (m, i) \in A)$ that belongs to unassigned set $(i \in unassigned)$ such that $dist\left(\bar{k}, i\right) \leq dist\left(\bar{k}, j\right), \forall j \in unassigned, j \neq i$ (break tie by using the node with the smallest node index). Add node i into the current cluster: $cl_k = cl_k \cup \{i\}$, update the center of gravity point with step 2, go back to step 3

4. Computational Experiments

To study the performance of simple heuristics, modified RKGA, Dynamic (Dyn T) and the project decomposition algorithm, three computational experiments are described in this section. In the first experiment, the proposed modified RKGA is compared with simple heuristics, basic simulated annealing and Dyn T algorithms on small size academic problem instances where optimal solutions are known. The decomposition versions of these heuristics other than simple heuristics are also examined. Since simple heuristics are fast enough to solve large size problem instances quickly, there is no need to integrate them with decomposition. In the second experiment, these heuristics are tested on large academic size problem instances where optimal solutions are unknown. Problem instances for these two experiments are generated and modified from ProGen (Kolisch et al. 1995) which is a well-known benchmark instance generator for academic research. Time-varying resource constraints and resource vacations are added since instances generated by ProGen assume constant resource profiles for renewable resources. Details of this procedure can be found at the Modified ProGen in the appendix of Cheng et al. (2014). In the third experiment, two study cases are generated based on the size and parameter settings of the practical Install/Qual scheduling problem. Due to confidentiality agreements, the actual Install/Qual data is not used in this work. In all three experiments, activity ready times and due dates are considered as data input. Network complexity (NC), resource factor (RF) and resource strength (RS) are

parameters from ProGen while resource range (RR) and resource vacation (RV) are parameters adopted from Cheng et al. (2014).

The network complexity (NC) factor measures the average number of non-redundant arcs per node including the dummy start and finish nodes. Network complexity level is set at 1.5 for all tested instances since it is the recommended setting in ProGen for low network complexity and the Install/Qual process has low network complexity. The resource factor (RF) measures the average ratio of the number of resource types required over the total available resource types for all activities. In the Install/Qual process, activities require two renewable resources on average (e.g. mechanics and plumbers) and two non-renewable resources (e.g. floor space and budget) at the same time. Thus in all three experiments where there are two types of renewable resources and two types of non-renewable resources, a resource factor $RF = 1$ is selected. In ProGen, resource strength (RS) is a normalized parameter to measure the "tightness" of a type of resource. $RS = 0$ means the resource level is very tight and there are very few feasible schedules with that resource level. $RS = 1$ indicates all resources are the least tight and their availability levels are high enough so that all activities can be scheduled at the earliest start time. In all three experiments, resource strength for renewable resource (RSR) varies in 10 different levels from 0 to 1. Instances are solved at each RSR level to understand the trade-offs between resource level and project makespan. For non-renewable resources, $RSN = 0.25$ represents the low level and

$RSN = 0.75$ the high level. These two values are selected based on standard ProGen settings and practical Install/Qual scheduling problems. The resource range factor (RR) in Cheng et al. (2014) aims to introduce randomness of resource limits of renewable resources. RR is a percentage value that measures the width of a resource limit range that is used to generate uniformly distributed random numbers. In all three experiments, $RR = 0.25$ is selected since the practical Install/Qual scheduling has a low level of resource fluctuation, especially for human resources. In Cheng et al. (2014), the resource vacation factor (RV) is a binary parameter to indicate whether resource vacations (e.g. weekends, holidays, etc.) are considered. In this research, resource vacations are included so $RV = 1$ for all three experiments. In summary, fixed parameter values are $NC = 1.5$, $RF = 1$, $RR = 0.25$ and $RV = 1$; while controllable parameter values are: $RSN = 0.25$ or 0.75 and RSR varies from 0 to 1. A summary of experiment parameter settings can be found in Table 14.

Table 14: Basic Parameter Settings for Tested Instances

Parameter	$\lvert\mathbb{R}^r\rvert$	$\lvert\mathbb{R}^n\rvert$	NC	RF	RSR	RSN	RR	RV
Value	2	2	1.5	1	0~1	0.25 or 0.75	0.25	1

In Experiment I, each tested instance has 20 activities since that is considered a medium level academic size problem instance and solving it to optimality is possible within a reasonable amount of computational time. Experiment II studies instances with 100 activities for large academic size problem instances. For all tested instances, there

are three alternative processing modes for each activity, two types of renewable resources and two types of non-renewable resources. In total, 200 instances each were tested for Experiments I and II. In Experiment III, ten instances are specifically designed based on the actual Install/Qual scheduling problem with half of them having a high *decom_socre* and the other half a low *decom_score*. Each instance has 1000 activities since the Install/Qual scheduling problem has about 1000 major activities, each representing a unique piece of capital equipment. The RSN parameter is limited to 0.75 which is similar to the current non-renewable resource level for the practical Install/Qual process. The main difference between these two instances is that one has a higher *decom_score* and the other one has a lower *decom_score*. A comparison summary of the three experiments is provided in Table 15.

Table 15: The Values of Basic Parameter Settings for the Three Experiments

Parameter	RSN	RSR	$\lvert\mathbb{N}\rvert$	$\lvert\mathbb{R}^r\rvert$	$\lvert\mathbb{R}^n\rvert$	$\lvert Mod_j\rvert$	# Tested instances
Experiment I	{0.25, 0.75}	(0, 1)	20	2	2	3	200
Experiment II	{0.25, 0.75}	(0, 1)	100	2	2	3	200
Experiment III	{0.75}	{0.75}	1000	2	2	3	10

ILOG-CPLEX is used to solve for optimal solutions. The priority rule-based simple heuristics, SA, Dyn T, modified RKGA and the project decomposition method are programmed in C++ using Microsoft Visual Studio 2010. All three experiments are conducted on a laptop with an Intel® Core ™ 2 Duo CPU P8400 @ 2.26GHz, 2 GB installed memory, and the Windows 7 Enterprise 32-bit Operating System.

5. Experimental Results and Discussion

Figure 12 and Figure 13 report the percentage of instances that are returned as "feasible" by a mode selection rule or SA, modified RKGA, Dyn T or the exact method for Experiment I and II, respectively. The infeasibility of an instance can result from non-renewable resources, the maximum project horizon or activity ready time and due date. In the two experiments, the feasibility levels between "RSN-high" and "RSN-low" are quite different. It shows that the majority of the infeasibility of tested instances come from processing mode selection related to non-renewable resources. Even with the mode repair operation, the SDM rule has the lowest percentage of feasible instances in both experiments. The reason is that the SDM rule, regardless of activity priority rule, selects the mode with the shortest duration which in most cases is the mode with highest resource usage. The same reason applies for the LTRU_R rule since it selects the rule with the least amount of renewable resources, not non-renewable resources. The LTRU_N rule minimizes non-renewable resources, thus, most of the cases returned feasible solutions. SD-LTRU_N tries to balance the activity duration and non-renewable resource usage, thus the feasibility level of instances sits between the SDM rule and the LTRU_N rule. Both SA and modified RKGA have an infeasibility penalty function to improve infeasible schedules towards feasibility. When decomposition is used for instances where non-renewable resource levels are low, the number of instances that cannot find a feasible solution increases. In Experiment II, when there are a larger

number of decomposed subprojects, the portion of instances that can find a feasible solution when RSN is low is even smaller than for the decomposition algorithm.

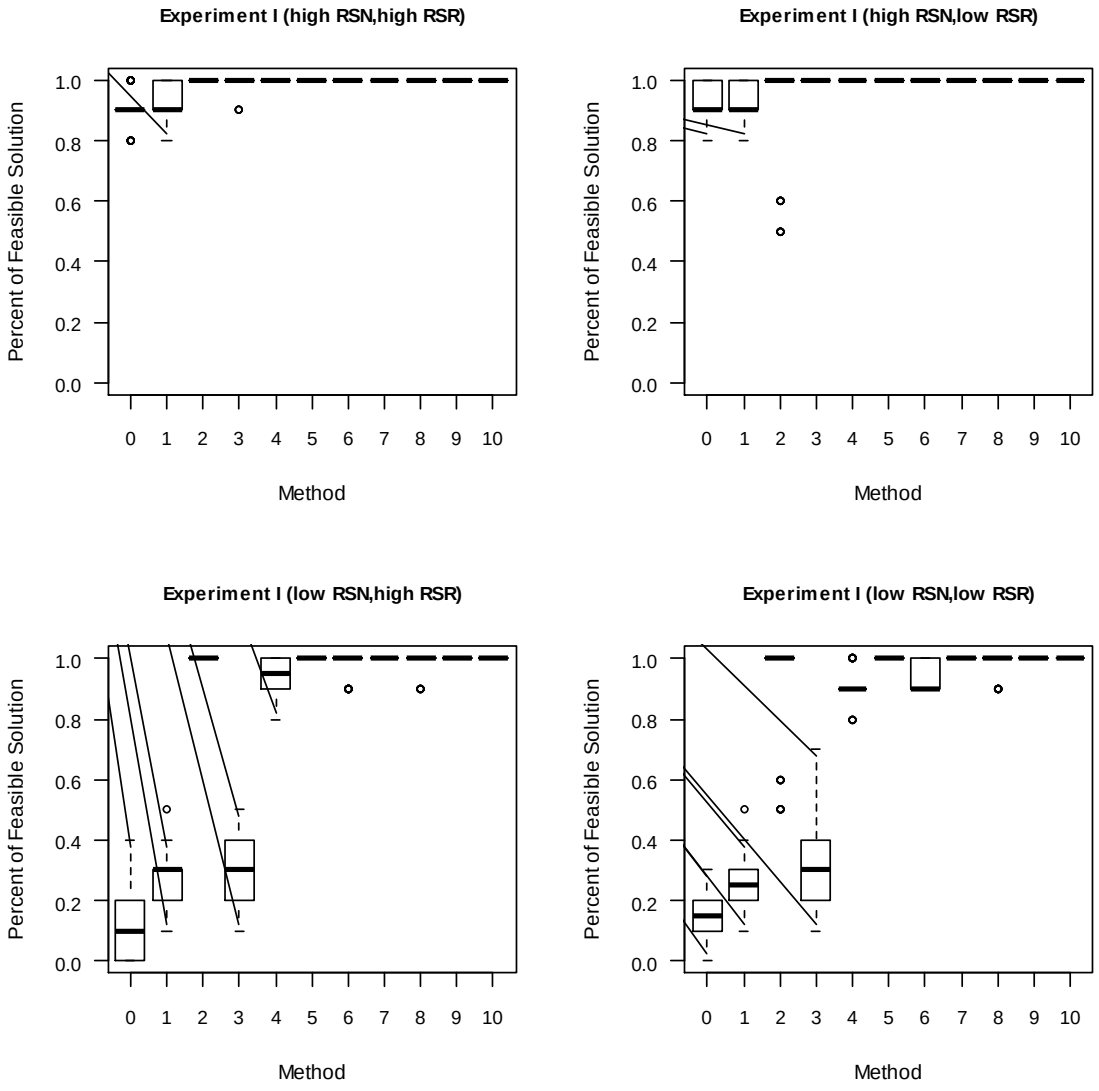

Figure 12: The Percentage of Feasible Solutions Found by Each Heuristic in Experiment I (0=SDM, 1=LTRU_R, 2=LTRU_N, 3=SD-LTRU_N, 4=SA-decom, 5=SA, 6=RKGA-decom, 7=RKGA, 8=DynT-decom, 9=DynT, 10=OPT)

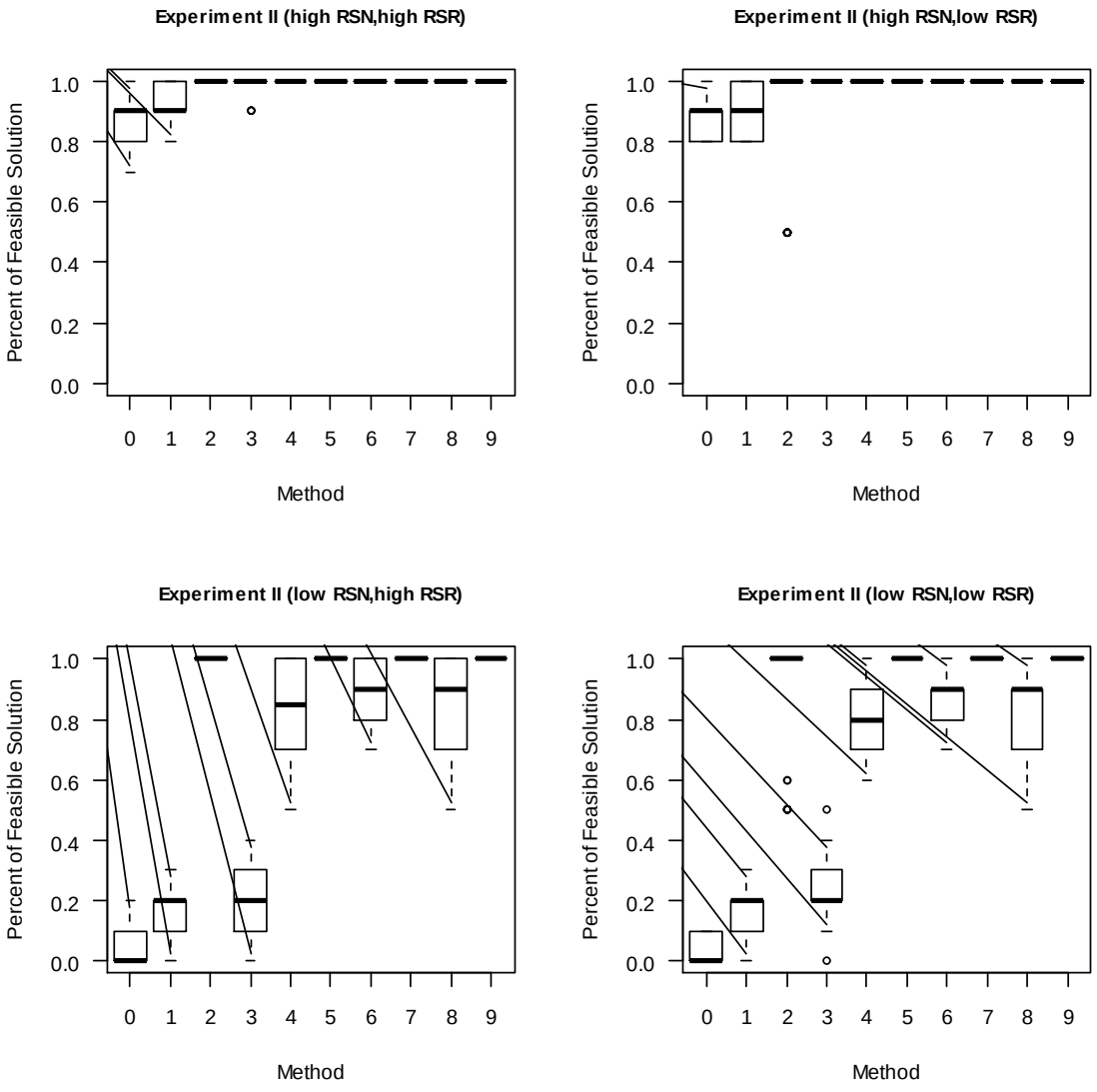

Figure 13: The Percentage of Feasible Solutions Found by Each Heuristic in Experiment II (0=SDM, 1=LTRU_R, 2=LTRU_N, 3=SD-LTRU_N, 4=SA-decom, 5=SA, 6=RKGA-decom, 7=RKGA, 8=DynT-decom, 9=DynT)

Optimality gap results for Experiment I are summarized in Figure 14. The optimal solution (OPT) is used as the baseline in each instance and the optimality gap measures the solution found by each heuristic compared to the baseline.

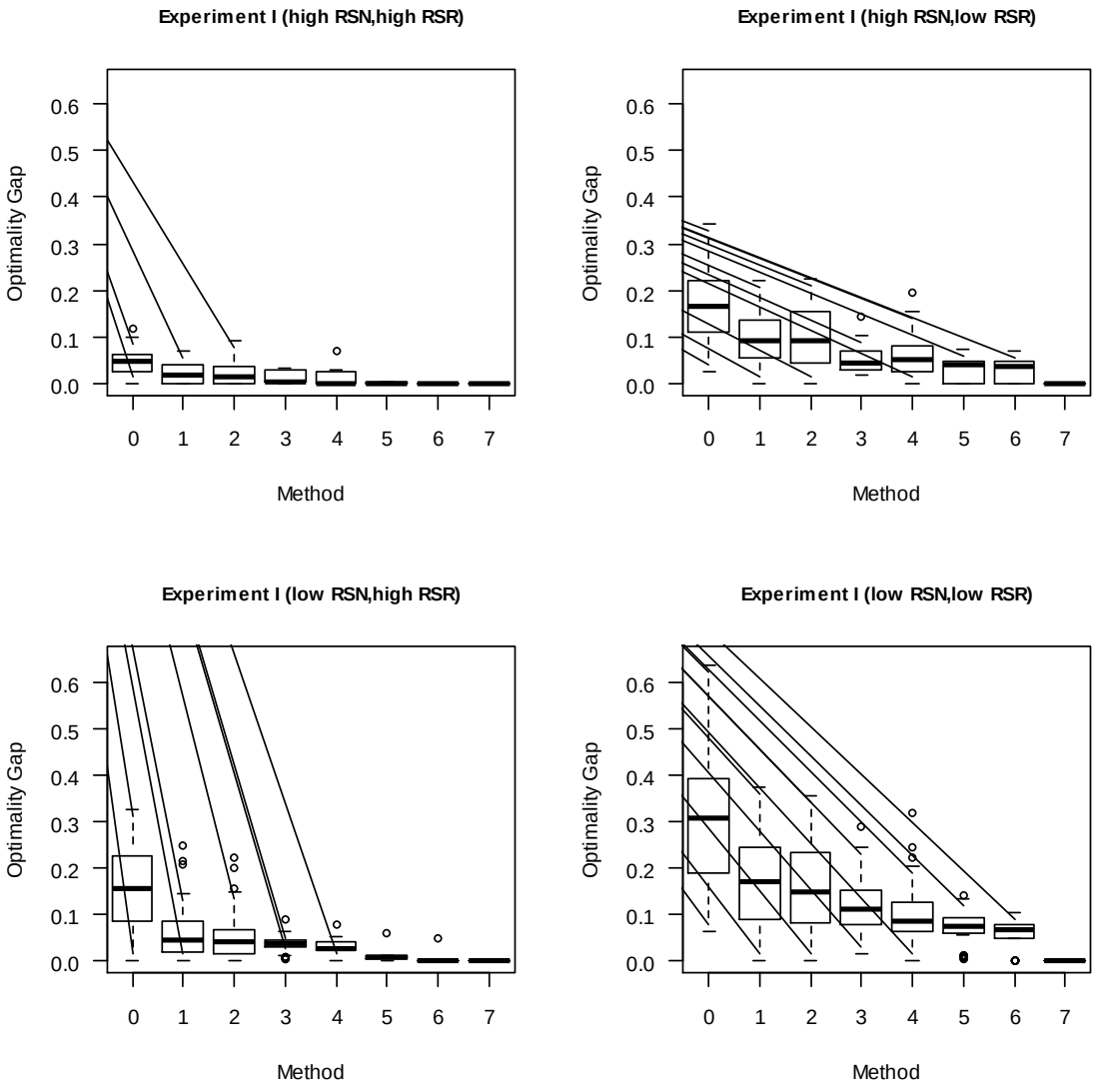

Figure 14: The Comparison of Optimality Gap of Each Heuristic in Experiment I (0=Best Simple, 1=SA-decom, 2=SA, 3=RKGA-decom, 4=RKGA, 5=DynT-decom, 6=DynT, 7=OPT)

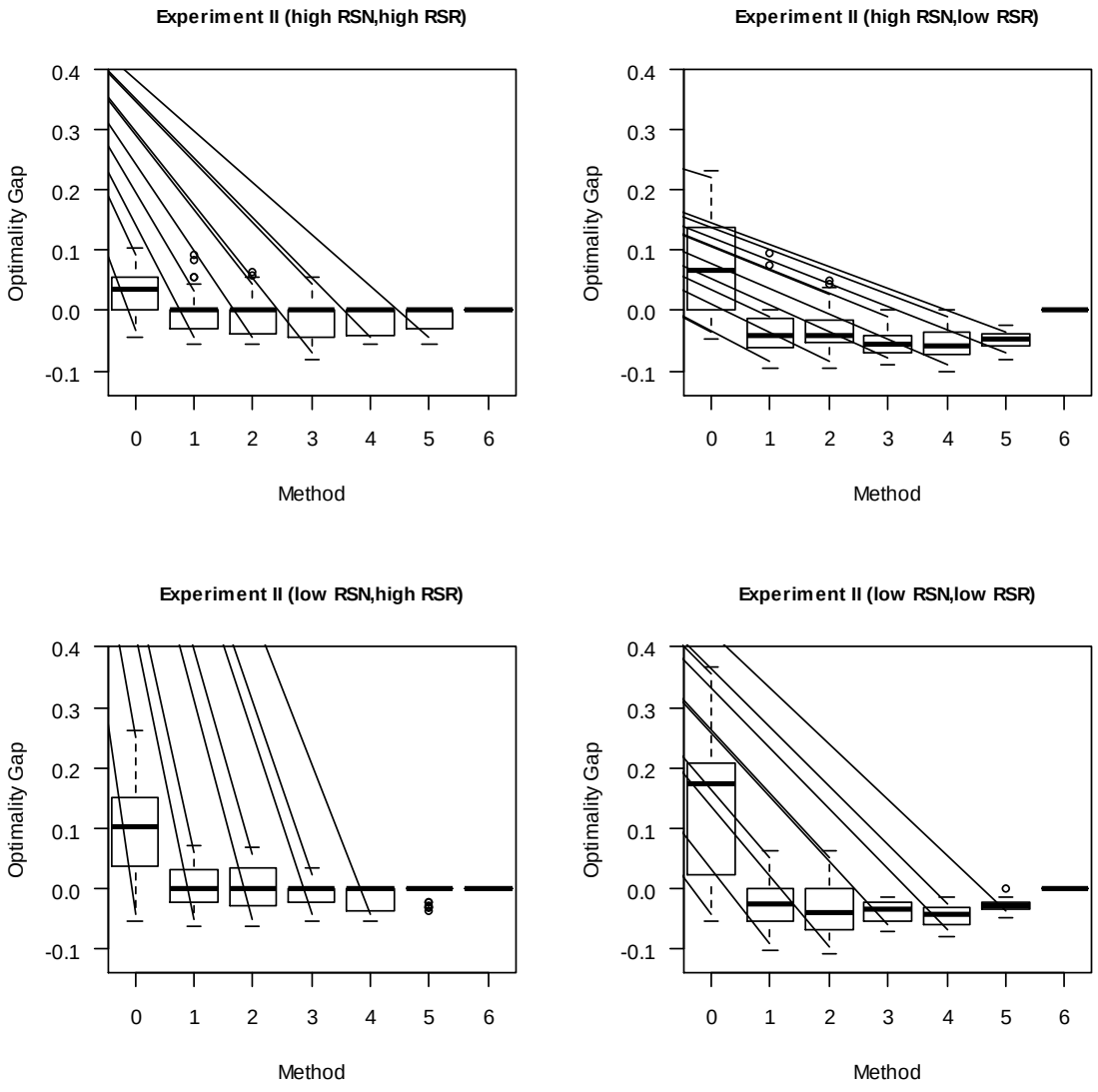

Figure 15: The Comparison of Optimality Gap of Each Heuristic in Experiment II (0=Best Simple, 1=SA-decom, 2=SA, 3=RKGA-decom, 4=RKGA, 5=DynT-decom, 6=DynT)

In Experiment II, since the optimal solution for each instance is not available, the solution found by Dyn T is considered as the new baseline (Figure 15). The reason of choosing Dyn T as the baseline is because Dyn T is the heuristic with the smallest variance in Experiment I. So choosing it in Experiment II can show solution variability

of other heuristics. For both experiments with and without decomposition, modified RKGA outperforms the Best Simple and SA method regardless of the resource levels. When resource levels are low (RSN = low and RSR = low), activities need to compete more for limited resources and instances tend to be more difficult to solve than instances with high resource levels (RSN = high and RSR = high) where most activities can be scheduled at their earliest possible time. Therefore under high resource levels, the optimality gap in Experiment I and the gap from the Dyn T in Experiment II are smaller than under low resource levels for Experiment I and Experiment II. The same result is observed in Buddhakulsomsiri and Kim (2007). Decomposition helps when non-renewable resource levels are high but not for low non-renewable resource levels.

Regression analyses are provided in Table 16 for Experiment I and Table 17 for Experiment II. For the mode selection rule comparison, the "Statistical Significance Levels" column reports the pairwise t-test and levels that are not connected by the same letter are significantly different.

Table 16: How Different Mode Selection Rules are Regarding to Makespan in Experiment I (Least Sq Mean Represents Fitted Makespan by Regression and Rules Represented with Different Levels are Statistically Different)

Model Selection Rule	Least Sq Mean	Levels		
3 (LTRU_N)	64.0	A		
1 (SDM)	58.1		B	
2 (LTRU_R)	55.4			C
4 (SD-LTRU_N)	54.6			C

Table 17: How Different Mode Selection Rules are Regarding to Makespan in Experiment II (Least Sq Mean Represents Fitted Makespan by Regression and Rules Represented with Different Levels are Statistically Different)

Mode Selection Rule	Least Sq Mean	Levels		
3 (LTRU_N)	224.7	A		
1 (SDM)	161.6		B	
4 (SD-LTRU_N)	151.3			C
2 (LTRU_R)	147.9			C

Even though mode selection rule 3 (LTRU_N) generates the highest percentage of feasible solutions (Table 16 and Table 17), it is the worst mode selection rule in both experiments regardless of the activity priority rules. As reported in Boctor (1996) and Buddhakulsomsiri and Kim (2006), the SDM rule has good performance regardless of the job priority rules. Our experiment shows similar results which is not surprising given that the SDM rule picks the shortest duration modes and tends to generate short schedules. However, shortest duration modes generally use the most resources when resource constraints limit activities to be processed in parallel. Thus, shortest duration modes do not necessarily result in the shortest project makespan. Furthermore, the SDM rule generates the lowest percent of feasible instances among all 4 mode selection rules (Table 18 and Table 19). In the two experiments, both the LTRU_R and SD-LTRU_N rules perform well. The SD-LTRU_N rule balances activity durations and non-renewable resource usage to achieve the shortest project makespan. The LTRU_R rule selects the mode with the least renewable resource usage which is calculated by the number of time periods each activity is processed and the resource usage at each time period. The more

preferable mode is selected by the LTRU_R rule when the shorter mode and less renewable resources are required at each time period. Both experiments show that mode selection rules that balance activity duration and resource usage generate the schedules with the shortest makespan. The difference among activity priority rules is less dramatic than the difference among the mode selection rules. Mode selection rules will apply on all activities. But with the network precedence constraint and resource constraints specified, activity priority rules might not impact all activities. For example, if precedence network and resource availability limit activity 1 is followed by activity 2 followed by activity 3, no matter what activity priority rule is selected, the sequence of activities 1, 2 and 3 is the same in the resulting schedule. However, SPT, ERT and EDD perform consistently poorly in both Experiments I and II. A possible explanation could be that these heuristics do not consider the project network which could explain why MLFT, MLST, MMSLK, GRPW and MSLK perform relatively well. MMSLK considers both slackness and activity duration and slightly outperforms MSLK in both experiments.

Table 18: How Different Activity Selection Rules are Regarding to Makespan in Experiment I (Least Sq Mean Represents Fitted Makespan by Regression and Rules Represented with Different Levels are Statistically Different)

Activity Priority Rule	Least Sq Mean	Level				
1 (SPT)	60.7	A				
3 (ERT)	59.9	A	B			
4 (EDD)	59.4	A	B	C		
2 (LPT)	58.8	A	B	C	D	
10 (MMSLK)	57.9		B	C	D	E
8 (MTS)	57.2			C	D	E

7 (MLFT)	56.8	D	E
5 (MSLK)	56.8	D	E
9 (GRPW)	56.8	D	E
6 (MLST)	56.1		E

Table 19: How Different Activity Selection Rules are Regarding to Makespan in Experiment I (Least Sq Mean Represents Fitted Makespan by Regression and Rules are Statistically Different are Represented with Different Levels)

Activity Priority Rule	Least Sq Mean	Level		
1 (SPT)	186.7	A		
4 (EDD)	179.5	A		
3 (ERT)	179.0	A	B	
2 (LPT)	176.6	A	B	C
10 (MMSLK)	175.3	A	B	C
5 (MSLK)	168.1		B	C D
8 (MTS)	167.4			C D
9 (GRPW)	162.1			D
6 (MLST)	160.1			D
7 (MLFT)	159.0			D

The interaction of mode selection rules and activity selection rules are studied through a pair-wise student t-test in Table 20 and Table 21. Basic assumptions for the student t-test are checked and satisfied. Surprisingly, in the two experiments studied in this research, the effects of the cross term is not statistically significant when the model has both the mode selection rule term and activity selection rule term.

Table 20: The Statistical Significance of Mode Selection Rules and Activity Selection Rules to Makespan in Experiment I

Source	DF	Sum of Squares	F Ratio	Prob > F
Mode Selection Rule	3	86589.5	89.9	<.0001
Activity Priority Rule	9	11165.5	3.9	<.0001
Mode Selection Rule *Activity Priority Rule	27	4286.8	0.5	0.9867

Table 21: The Statistical Significance of Mode Selection Rules and Activity Selection Rules to Makespan in Experiment II

Source	DF	Sum of Squares	F Ratio	Prob > F
Mode Selection Rule	3	6201795.6	261.0	<.0001
Activity Priority Rule	9	373143.1	5.2	<.0001
Mode Selection Rule *Activity Priority Rule	27	60565.1	0.3	0.9999

To understand the impact of the decomposition algorithm, Table 22 shows the relationship between decomposition impact (measured by the makespan difference for the same algorithm with and without decomposition) with the *decom_score*. The linear regression coefficients show that with the increase of *decom_score*, decomposition is less useful for the SA, RKGA and Dyn T algorithms. This result is intuitive since the decomposition score is defined as the ratio of the distance for activities in the same subproject with the distance among subprojects. The smaller the *decom_score*, activities in subprojects are closer to each other than activities in other subprojects, so decomposition tends to make more sense.

Table 22: The Relationship between decom_score and the Impact of Using Decomposition with Different Heuristics

Impact of decomposition vs. Decomposability	Fitted Equation	RSquare
(SA – SA_decom) / SA	-0.1011 *decom_score* +0.0072	0.239
(RKGA – RKGA_decom) / RKGA	-0.1302 *decom_score* +0.0109	0.291
(Dyn T – Dyn T_decom) / Dyn T	-0.0864 *decom_score* +0.0037	0.173

In Experiment III, the results for ten instances are shown in Table 23. When problem instance size is as large as 1000 activities, the original problem instance has to

be decomposed into many subprojects to be able to solve with the Dyn T algorithm, so the results are not very attractive. For both sets of instances, RKGA (RKGA-decom) performs better than SA (SA-decom), Dyn T-decom and Best Simple. For instances with low *decom_score* (instances 1-5), the decomposition version of algorithms (RKGA-decom, SA-decom) perform better than without decomposition (RKGA, SA). The opposite results are observed for instances with high *decom_score* (instances 6-10) which is consistent with Experiments I and II. Compared to the Best Simple which is the baseline of current heuristics in practice, the proposed modified RKGA and RKGA-decom can reduce the total project duration by 40~70 days out of total 300~400 days in the total project makespan.

Table 23: Makespan Found by Each Heuristic for all Tested Instances in Experiment III with Relationship with decom_score (best values highlighted)

Instances (days)	1	2	3	4	5	6	7	8	9	10
decom_score	0.16	0.18	0.11	0.1	0.17	0.99	0.92	0.9	1.01	0.85
Best Simple	380	370	394	384	385	364	391	376	386	374
SA-decom	317	309	339	343	336	340	382	369	375	372
SA	331	329	355	358	354	327	360	352	342	357
RKGA-decom	296	294	324	325	322	332	361	363	353	364
RKGA	322	315	344	341	346	314	343	336	343	342
Dyn T-decom	352	341	352	352	358	346	361	352	359	353

6. Conclusion and Future Research

In this paper, the semiconductor capital equipment installation and qualification scheduling problem is modeled as a multi-mode resource-constrained project scheduling

problem with non-preemptive activity splitting. Due to the NP-hardness of the problem and practically-motivated large-sized problem instances, we deploy and compare four different heuristic approaches. Computational experiments show that when an instance's decomposition score is low, combining decomposition with other meta-heuristics is recommended. Decomposition algorithms work better when availability levels for non-renewable resources are high. Overall, the proposed modified RKGA outperforms simulated annealing, simple heuristics and the Dyn T approach, especially for practical-size problem instances. Since static problems are studied in this work, possible future research could consider uncertain activity ready times, due dates or processing times with simulation. Another possible extension is to include other objectives such as time value of money or total amount of resource consumption in addition to project durations.

Supplementary material 1: mathematical formulation from Cheng et al. (2014)

The primary decision variables are as follows:

$y_j^m = 1$ if activity $j \in N$ is being processed in mode $m \in Mod_j$ and 0 otherwise

$x_{jt}^m = 1$ if activity $j \in N$ is being processed in mode $m \in Mod_j$ at time $t = 1, 2, \dots, T$ and 0 otherwise

In addition, variables S_j and C_j represent the start time and completion time of activity j and the start time of the dummy finish activity $S_{|N|+1}$ is essentially the project makespan. Data inputs are resource profiles $[0, U_{kt}]$ for renewable resources and $[0, U_k]$ for non-renewable resources.

The PMRCPSP (***P3***) formulation as given by Buddhakulsomsiri and Kim (2006) can be represented as follows:

$$\min\ S_{|N|+1} \tag{33}$$

subject to

$$\sum_{m \in Mod_j} y_j^m = 1,\ 1, \forall j \in N \tag{34}$$

$$\sum_{t=1}^{T} x_{jt}^m = p_j^m \cdot y_j^m, \quad \forall j \in N, m \in Mod_j \tag{35}$$

$$C_i \leq S_j - 1, \quad \forall (i,j) \in A \tag{36}$$

$$S_j \leq x_{jt}^m \cdot t + M\left(1 - x_{jt}^m\right), t\ \forall j \in N, m \in Mod_j, t = 1, 2, \dots, T \tag{37}$$

$$C_j \geq x_{jt}^m \cdot t, \quad \forall j \in N, m \in Mod_j, t = 1, 2, \dots, T \tag{38}$$

$$S_j \geq rad_j, \quad \forall j \in N \tag{39}$$

$$C_j \leq due_j, \quad \forall j \in N \tag{40}$$

$$\sum_{j\in N}\sum_{m\in Mod_j} r_{jk}^{m}\cdot x_{jt}^{m} \le U_{kt}, t\ \forall k \in R^{r}, t = 1, 2, \dots, T \quad (41)$$

$$\sum_{t=1}^{T}\sum_{j\in N}\sum_{m\in Mod_j} r_{jk}^{m}\cdot x_{jt}^{m} \le U_{k}, td\forall k \in R^{n} \quad (42)$$

$$y_{j}^{m} \in \{0,1\},\ 1\ \forall j \in N, m \in Mod_j \quad (43)$$

$$x_{jt}^{m} \in \{0,1\},\quad \forall j \in N, m \in Mod_j, t = 1, 2, \dots, T \quad (44)$$

$$S_j \ge 0,\ \forall j \in N \quad (45)$$

$$C_j \ge 0,\ \forall j \in N \quad (46)$$

The objective function (1) minimizes the project makespan that can be represented by the starting time of the dummy finish activity $|N| + 1$. Constraint set (2) ensures exactly one mode is selected for each activity. Constraint set (3) ensures that if mode m is selected for activity j, the total processing time must equal the corresponding duration. Constraint sets (4) – (6) are precedence constraints and a big number M can be set as the maximum project planning horizon T. The "-1" in (4) removes strict inequality given integer time units (e.g. an arc (2, 3) and activity 3 starts on time unit 5, $S_3 = 5$, activity 2 has to complete before or on time unit 4 $C_2 \le 5 - 1$). Activity ready times and due dates constraints are in (7) - (8). Constraint sets (9) – (10) specify resource availability for both renewable resources and non-renewable resources, respectively. Constraint sets (11) – (14) are binary (11 and 12) and non-negativity (13 and14) constraints.

To modify the $\boldsymbol{P3}$ formulation for $\boldsymbol{P1}$, constraint set (15) is added to ensure that the duration from the activity start time to the completion time equals the activity duration. In other words, there is no activity splitting for any activity.

$$C_j - S_j = \sum_{m \in Mod_j} \sum_{t=1}^{T} x_{jt}^m - 1, \quad \forall j \in N \tag{47}$$

To modify the $\boldsymbol{P3}$ formulation for $\boldsymbol{P2}$, an indicator function is introduced to specify whether an activity j in mode m is feasible to process at a certain time period:

$$\gamma_{jkt}^m = 1_{[0,U_{kt}]}\left(r_{jk}^m\right) := \begin{cases} 1 & \text{if } r_{jk}^m \in [0, U_{kt}], \ \forall t \\ 0 & \text{otherwise} \end{cases} \tag{48}$$

Additional decision variables o_{jt} and q_{jt} are defined to indicate whether a time period t is between the start time S_j and the completion time C_j of activity j.

$$o_{jt} = \begin{cases} 1 & \text{if } t \le C_j \\ 0 & \text{otherwise} \end{cases}, \quad \forall j \in N \tag{49}$$

$$q_{jt} = \begin{cases} 1 & \text{if } t \ge S_j \\ 0 & \text{otherwise} \end{cases}, \quad \forall j \in N \tag{50}$$

Additional constraint sets (19) – (22) are included to support the new decision variables o_{jt} and q_{jt}. As before, a big number M can be set as the maximum project planning horizon T. Constraint sets (23) – (24) restrict that an activity j cannot be preempted at time t if it is eligible. Constraint sets (25) – (26) are additional variable type constraints.

$$M \cdot o_{jt} \ge C_j - t + 1, \quad \forall j \in N, t = 1, 2, \dots T \tag{51}$$

$$M \cdot \left(1 - o_{jt}\right) \ge t - C_j, \quad \forall j \in N, t = 1, 2, \dots T \tag{52}$$

$$M \cdot q_{jt} \ge t - S_j + 1, \ Eq \forall j \in N, t = 1, 2, \dots T \tag{53}$$

$$M \cdot \left(1 - q_{jt}\right) \ge S_j - t, \quad \forall j \in N, t = 1, 2, \dots T \tag{54}$$

$$x_{jt}^m \ge y_j^m + \gamma_{jkt}^m + o_{jt} + q_{jt} - 3, \ \forall j \in N, m \in Mod_j, k \in R^n, t = 1, 2, \dots T \tag{55}$$

$$4 \cdot x_{jt}^{m} \leq y_{j}^{m} + \gamma_{jkt}^{m} + o_{jt} + q_{jt}, t \; \forall j \in N, m \in Mod_{j}, k \in R^{n}, t = 1, 2, \ldots T$$

(56)

$$o_{jt} \in \{0, 1\}, \; 1 \; \forall j \in N, t = 1, 2, \ldots T \tag{57}$$

$$q_{jt} \in \{0, 1\}, \quad \forall j \in N, t = 1, 2, \ldots T \tag{58}$$

Supplementary material 2: priority rule-based simple heuristics

Table S-1: Mode Selection Rules

Priority Rules	Mathematical Formula	Selected Reference
SDM (shortest duration mode)	$\left\{m \in Mod_j \middle\| p_j^m = min_{\forall l \in Mod_j} p_j^l\right\}$	Boctor (1996), Buddhakulsomsiri and Kim (2007)
LTRU_R (least total renewable resource usage)	$\left\{m \in Mod_j \middle\| \sum_{k \in \mathbb{R}^r}\left(r_{jk}^m \cdot p_j^m\right) = \min_{\forall l \in Mod_j} \sum_{k \in \mathbb{R}^r}\left(r_{jk}^l \cdot p_j^l\right)\right\}$	Boctor (1996), Buddhakulsomsiri and Kim (2007)
LTRU_N (least total non-renewable resource usage)	$\left\{m \in Mod_j \middle\| \sum_{k \in \mathbb{R}^n} r_{jk}^m = \min_{\forall l \in Mod_j} \sum_{k \in \mathbb{R}^n} r_{jk}^l\right\}$	Boctor (1996), Buddhakulsomsiri and Kim (2007)
SD-LTRU_N (shortest duration and least non-renewable resource usage)	$\left\{m \in Mod_j \middle\| \sum_{k \in \mathbb{R}^n}\left(r_{jk}^m * p_j^m\right) = \min_{\forall l \in Mod_j} \sum_{k \in \mathbb{R}^n}\left(r_{jk}^l * p_j^l\right)\right\}$	This paper

Table S-2: Activity Priority Rules

Priority Rules	Mathematical Formula	Selected Reference
SPT (shortest processing time)	$\left\{j \in \mathbb{N} \middle\| p_j^m = \min_{l \in \mathbb{N}} p_l^m\right\}$	Alvarez-Valdes and Tamarit (1989), Lova et al. (2006)
LPT (longest processing time)	$\left\{j \in \mathbb{N} \middle\| p_j^m = \max_{l \in \mathbb{N}} p_l^m\right\}$	Alvarez-Valdes and Tamarit (1989), Lova et al. (2006)
ERT (earliest ready time)	$\left\{j \in \mathbb{N} \middle\| rad_j = \max_{l \in \mathbb{N}} rad_l\right\}$	This paper
EDD (earliest due date)	$\left\{j \in \mathbb{N} \middle\| due_j = \min_{l \in \mathbb{N}} due_l\right\}$	This paper

MSLK (minimum slackness)	$\left\{j \in \mathbb{N} \middle\| LST_j - EST_j = \min_{l \in \mathbb{N}}(LST_l - EST_l)\right\}$	Davis and Patterson (1975) Buddhakulsomsiri and Kim (2007)
MLST (minimum latest start time)	$\left\{j \in \mathbb{N} \middle\| LST_j = \min_{l \in \mathbb{N}} LST_l\right\}$	Alvarez-Valdes and Tamarit (1989), Kolisch (1995)
MLFT (minimum latest finish time)	$\left\{j \in \mathbb{N} \middle\| LFT_j = \min_{l \in \mathbb{N}} LFT_l\right\}$	Davis and Patterson (1975)
MTS (maximum total successors)	$\left\{j \in \mathbb{N} \middle\| \|succ(j)\| = \max_{l \in \mathbb{N}} \|succ(l)\|\right\}$	Alvarez-Valdes and Tamarit (1989)
GRPW (greatest rank positional weight)	$\left\{j \in \mathbb{N} \middle\| p_j^m + \sum_{i \in ALL\_succ(j)} p_j^m = \max_{l \in \mathbb{N}}\left(p_j^m + \sum_{i \in ALL\_succ(j)} p_j^m\right)\right\}$	Helgeson and Birnie (1961), Buddhakulsomsiri and Kim (2007)
MMSLK (modified minimum slack)	$\left\{j \in \mathbb{N} \middle\| \left(LST_j - EST_j\right)/p_j^m = \min_{l \in \mathbb{N}}\left((LST_l - EST_l)/p_l^m\right)\right\}$	This paper

Supplementary material 3: serial SGS and priority rule-based simple algorithm

Serial SGS Algorithm

***Step 1*:** initialize the set of already scheduled activities SJ, initialize $SJ = \emptyset$, the set of un-scheduled activities UJ, initialize $UJ = \mathbb{N} \cup \{0\} \cup \{|\mathbb{N}| + 1\}$, the set of active activities AJ, calculated as follows: If $j \in UJ$ and $\forall l \in pred(j)$, $l \in SJ$, then $j \in AJ$. Initialize the dummy finish activity into the first active activities $AJ = \{|\mathbb{N}| + 1\}$ since the precedence network is reversed in backward scheduling. $t = T$ (backwards)

***Step 2*:** select the activity j from set AJ with highest priority value, schedule the activity from its "earliest" schedulable time unit t:

1) all predecessors are completed: $\forall l \in pred(j)$, $t \leq C_l$
2) the renewable resource levels are sufficient: $\forall t \in [S_j, C_j]$, $U_{kt_{temp}} \geq r_{jk}^m$
3) Determine the start time S_j and completion time C_j based on the following constraints to satisfy non-preemptive activity splitting: $S_j \leq rad_j{}'$, $p_j^m = \sum_{t=S_j}^{C_j}\left(1_{[0,U_{kt}]}\right)$, $\forall t \in [S_j, C_j]$, if $1_{[0,U_{kt}]} = 1$, then $x_{jt}^m = 1$. Check if $C_j < due_j{}'$, then the solution is infeasible, stop. Else, update resource level: $\forall k \in \mathbb{R}^r$, if $x_{jt}^m = 1$, $U'_{kt_{temp}} = U_{kt_{temp}} - r_{jk}^m$

Step 3: update: $SJ' = SJ - \{j\}$, $UJ' = UJ + \{j\}$. If $UJ' \neq \emptyset$, go to step 2, else, return C_0 as the solution. End.

Priority Rule-based Simple Algorithm

Step 1: Assign a processing mode for each activity according to *Mode Selection Rule*.

Step 2: Check mode feasibility regarding non-renewable resources by *Resource Feasibility Check*, if the mode assignment is not feasible, run *Mode Repair Operation*.

Step 3: Check mode feasibility again after *Mode Repair Operation*. If infeasible, return infeasible and stop; otherwise, go to step 3.

Step 4: Assign activity selection key for each activity according to *Activity Priority Selection Rule*.

Step 5: Perform *Schedule Generation Scheme* operation,

Step 6: Apply *Time Feasibility Check*, if feasible, return the completion time of the dummy start activity and stop; if not, return infeasible and stop

Mode Selection Rule

Specify mode assignment vector $MOD = \{m_j | m_j \in Mod_j, j \in \mathbb{N}\}$ for each activity j according to the mode selection rule, break ties by selecting the lower index mode.

Mode Repair Operation

Step 1: Randomly select an activity j

Step 2: Randomly assigned a new mode m_j' ($m_j' \neq m_j$) to form a new mode assignment MOD' to replace the old mode m_j in MOD

Step 3: Check for non-feasibility $NF^{NR\prime}$ for MOD'. If $NF^{NR\prime} \geq 0$, return MOD', exit; else if $NF^{NR\prime} \geq NF^{NR}$, accept m_j' and MOD', go to step 1 for *ITER* number of iterations; else if $NF^{NR\prime} < NF^{NR}$, reject m_j' and MOD', go to step 1 for *ITER* number of iterations

Activity Priority Selection Rule

Specify activity priority vector $RK = \{RK_j | RK_j \in \text{UNIF}(0,1), j \in \mathbb{N}\}$ for each activity j according to the activity priority selection rule, ties broken by selecting the lower index activity.

Resource Feasibility Check

Check for non-feasibility value NF_{MOD}^{NR} regarding to non-renewable resources for the mode assignment vector MOD:

$NF_{MOD}^{NR} = \sum_{k \in \mathbb{R}^n} \left(\max\left(0, \sum_{j \in \mathbb{N}} r_{jk}^m - U_k\right)\right)$

If $NF_{MOD}^{NR} = 0$, feasible; else if $NF_{MOD}^{NR} > 0$, infeasible.

Time Feasibility Check

Check for non-feasibility value NF_{SOL}^{RD} regarding to ready time* due_j' for each activity:

$NF_{SOL}^{RD} = \sum_{j \in \mathbb{N}} \left(\max\left(0, -S_j + due_j'\right)\right)$

If $NF_{SOL}^{RD} = 0$, the current schedule is feasible regarding ready time and due date; else if $NF_{SOL}^{RD} > 0$, the current schedule assignment is infeasible regarding ready time or due date. (*It is due date due_j in forward scheduling)

CHAPTER 4 A SCHEDULING FRAMEWORK FOR THE SEMICONDUCTOR EQUIPMENT INSTALLATION AND QUALIFICATION PROCESS WITH UNCERTAIN MARKET ENVIRONMENT

1. Introduction

As a capital intensive industry, investing and building a semiconductor wafer fabrication (fab) facility requires strategic decision making and careful planning. A state-of-the-art 300mm wafer fab can cost from $3 billion USD (Chien and Zheng (2012), Chasey and Pindukuri (2012)) to $10 billion USD (Ibrahim, Chik and Hashim, 2014) with production capacity from a few thousands WSPW (wafer start per week, a semiconductor terminology to measure production capacity) takes about 2-3 years in various sequential steps. A few hundred to over a thousand pieces of production equipment need to be installed and qualified (Install/Qual process) during the capacity ramp process. The majority of these equipment cost over $10 million dollars. For example, a single piece of optical photolithography stepper tool for reproducing the reticle pattern on wafers costs over $100 million (Lapedus, 2010). Thus, the benefit of accurate capacity planning can be significant. On one hand, the cost for over capacity can result in hundreds of millions of dollars wasted on idle assets or excessive product inventory that leads to low inventory turns and less free cash flow. On the other hand, the cost for under capacity can be worse since sales will not only be lost, but also potentially

a loss of market share and customer loyalty. Thus, a good Install/Qual schedule ramps the right amount of capacity at the right time with the right ramping speed to take advantage of both market price and market demand.

In practice, the entire capacity ramp process is phased into multiple steps. The amount of ramped capacity at each step is determined by the production capacity of all installed and qualified equipment during that step. Multiple types of resources (e.g. trades, supplier resources and company resources) with possibly different working calendars (5 days/week with 8 hours/day, 4 days/week with 10 hours/day or 7 days/week with 24 hours/day) are involved at the same time for each activity. For practical reasons, there are precedence relations among activities when processing them (e.g. some supporting equipment needs to be installed and qualified before installing other equipment). Figure 16 shows a Gantt chart of a sample capacity ramp process. The Y axis represents different pieces of capital equipment while the X axis is time and the color bars represent different Install/Qual activities. It is clear there are multiple ramp steps and activities are processed in parallel with common resources. From a project management point of view, it is critical to manage resources and schedule activities in the Install/Qual process to achieve the maximum expected overall profit which is the difference between product revenue from satisfying customer demand and the time value of money for capital equipment investment and resource consumption costs.

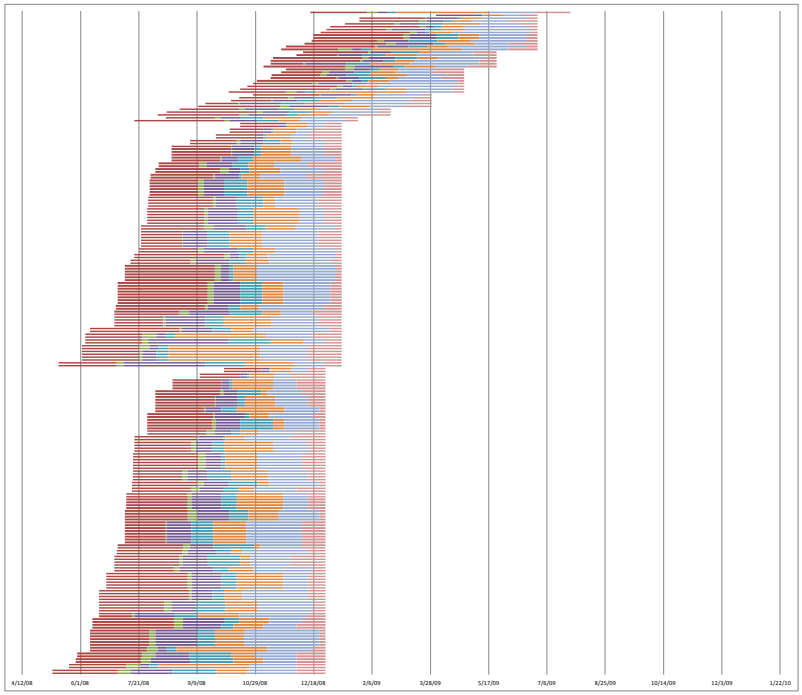

Figure 16: Fab Capacity Ramp Illustration

As shown in Cheng et al. (2014), scheduling the Install/Qual process with the simple objective of minimizing project makespan is challenging enough. The additional information about market price and market demand increases the level of complexity of the problem. For example, it might worthwhile to ramp the Install/Qual process faster (slower) with extra resource consumption because of a higher (lower) forecast of market demand or market price. Even more, the capacity planning of the Install/Qual process happens 2-3 years ahead, so the realized market demand and market price can be dramatically different from original forecasts. In Figure 17, both uncertain market demand and price are illustrated by three possible trending scenarios. The uncertainties of market demand and price increase as the increase in time representing the natural of

forecast. The right capacity planning strategy needs to balance both the demand and price trending slope and variability. One of such capacity ramp scenario is illustrated in the graph. Therefore, the Install/Qual schedule needs to be evaluated under both static conditions and uncertain market information.

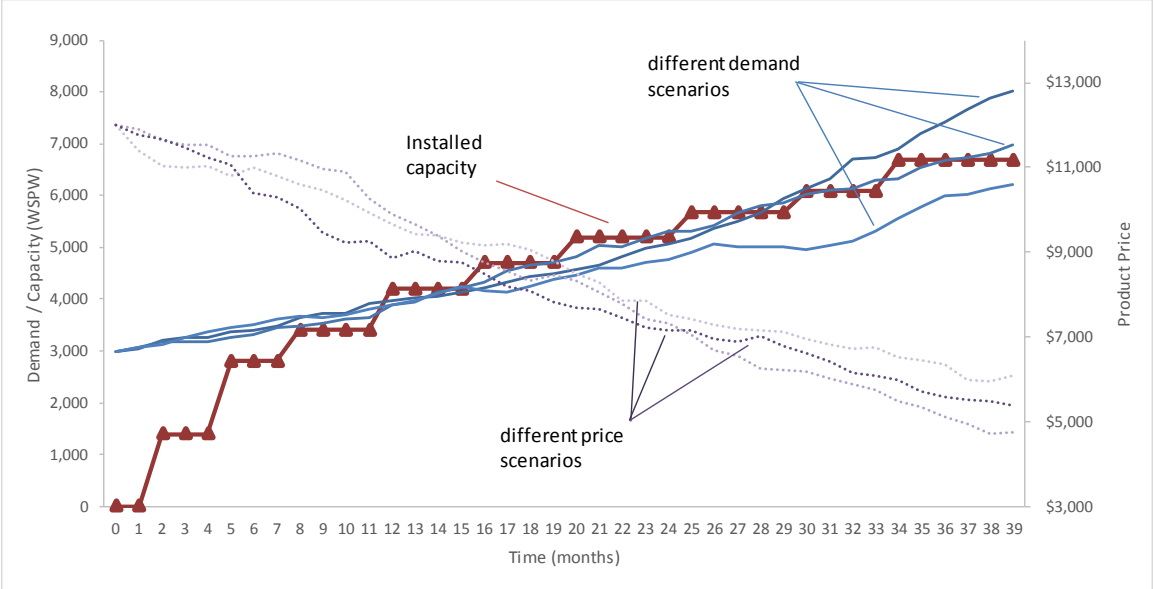

Figure 17: Capacity Ramping under Uncertain Market Price and Market Demand

Chou et al. (2007) (Table 24) categorize capacity planning problems in semiconductor manufacturing into three different levels. Assuming production machines are installed, short-term capacity planning focuses on the operational level; specifically on how to optimally reschedule customer orders to match production capacity with demand. Mid-term capacity planning deals with the tactical level machine portfolio such that decisions are related to when to purchase and install each piece of equipment for

capacity ramping. Long-term capacity planning focuses on longer horizon (e.g. 2-3 years) strategic level business investment related to what product lines to offer in an uncertain market environment. This involves not only the firm's own capacity, but also their competitors' capacity investments. The Install/Qual scheduling problem discussed in Cheng et al. (2014) focuses on when to install and qualify each machine is an example of a mid-term capacity planning problem and this work is a mid-term to long-term problem since we not only focus on the equipment Install/Qual schedule but also capacity investment decisions under an uncertain environment.

Table 24: Capacity Planning Horizons and Objectives

Horizon	**Objective**
short-term	order fulfillment, order rescheduling, alternative routing
mid-term	machine portfolio optimization, machine purchase and decommission
long-term	business planning in technology development, product planning

Since the return on investment in semiconductor capital equipment is highly uncertain, the financial concept of *real options* has become a popular approach to model the risk in capacity investment. The idea of *real options* in finance refers to an alternative or choice that becomes available with a business investment opportunity. In semiconductor capital capacity planning, each unit of capacity once invested provides the option to produce a certain amount of product which is referred as the operating options. On the other hand, the firm also has options to add more capacity which are known as

growth options (Argoneto et al., 2008). Investment decisions are discrete instead of continuous; they could be reversible or irreversible; future decisions depend on decisions made now (Broadie and Detemple, 2004). The Install/Qual process studied in this research is only focused on the capacity ramp up (capacity growth) phase of a fab facility. A similar focus can be found in Benavides et al. (1999) which evaluates different capital investment strategies to sequentially add capacity to satisfy demand growth over time. The expected net present value of future profits is maximized in their research using a cash flow model. Given the volatile environment for demand and the long planning lead time for capital investment, the main challenge studied in Benavides et al. (1999) is to balance the risk of over capacity for idle assets as well as under capacity for losing sales. Outsourcing is assumed as an alternative to adding capacity. The capacity deployment problem is modeled as an optimal stopping problem such that the optimal capacity level can provide the maximum expected net present value which is measured by the per wafer variable production cost, per wafer outsourcing cost and expected return on capacity investment. Assuming irreversible capital investment, Pindyck (1988) shows that the optimal capacity level is the level when the benefit of an incremental unit of capacity equals to its cost. Dangl (1999) shows the future demand uncertainty leads to an increase in optimal capacity and it is optimal to delay decision making to wait for further information even for a small amount of uncertainty. Comprehensive survey on options can be found in Miller and Park (2002) and Broadie and Detemple (2004).

Compared to previous research, this work proposes a scheduling framework to deal with the capacity ramp up process in the semiconductor industry with uncertain market demand and market price. The objective is to determine the optimal capacity ramp strategy supported by an Install/Qual schedule to maximize expected profit which includes the capital investment of equipment, resource utilization cost during the Install/Qual process and revenue generated by fulfilling customer demands. Practical challenges during this process are considered including uncertain market information, sharing common resources, resource vacations and calendars, multiple activity processing modes, precedence constraints among activities, and only allowing activity splitting but not resource preemption. The rest of this paper is organized as follows. Section 2 describes the Install/Qual scheduling problem with static and uncertain market information and how it is modeled as modified a multi-mode resource-constrained project scheduling problem (MRCPSP). Section 3 discusses the proposed scheduling framework and embedded algorithms. A computational study can be found in section 4 and is followed by section 5 with conclusions and possible future studies.

2. Problem Statement

The basic MRCPSP structure is adopted to model the Install/Qual scheduling problem as follows. A project network $G(N, A)$ contains a set of nodes N representing a set of activities $\{j \in N\}$ in the Install/Qual process, e.g. physical installation, supplier

qualification and company qualification and a set of directed arcs A representing the precedence relations among activities. An activity can start as soon as all of its immediate preceding activities are finished. For the purpose of network completeness and modeling convenience, a dummy start node 0 and a dummy finish node $N + 1$ are usually added into the project network. Within the context of this paper, "activities", "tasks" and "jobs" are considered interchangeable. The available units of a renewable resource k ($k \in RES^r$) is restricted by an upper bound U_{kt} and implicit lower bound 0 at each time period t which can also be considered as a "resource profile" function. For non-renewable resource k ($k \in RES^n$), U_k is the upper bound for overall available units throughout the entire planning horizon $[0, T]$. Renewable resource examples include the number of skilled technicians available per day and the number of testing machines available per shift. Examples of non-renewable resources include the total available budget for a project, the total available factory floor space, and the total available amount of raw materials.

Each activity j may have a set of processing modes Mod_j to select from and each mode $m \in Mod_j$ specifies the activity duration p_j^m and the amount of resource required r_{jk}^m for resource k. Based on the difference between preemption and activity splitting discussed at Cheng et al. (2014), the Install/Qual scheduling problem is a MRCPSP with non-preemptive activity splitting such that activities can only split when renewable resources are not available (weekends, holidays) or at a level less than the required

amount. In other words, non-preemptive activity splitting is a special case between MRCPSP without any activity splitting and preemptive MRCPSP where activities are allowed for arbitrary interruptions at any integer time point. Basic modeling of the problem can be found at Cheng et al. (2014) but several unique aspects of the Install/Qual process need to be modeled differently and are discussed as follows.

2.1 Capacity Ramp Up "Steps"

The Install/Qual process is the semiconductor capacity ramp up process. Compared to ramping up capacity in other industries, a fab does not wait for the entire Install/Qual process to be completed (which takes 2-3 years) to start manufacturing. Instead, the capacity ramp is broken into multiple steps ("ramp step") each with a certain increment of capacity expansion so that manufacturing can start as early as the first ramp step is completed. This requires at least one of each machine type. Figure **18** below illustrates capacity ramp step such that the height of a ramp step represents the amount of capacity being ramped and the width of a ramp step measures the time duration between two adjacent capacity ramps. All equipment can be classified into multiple ramp groups ($ra = 1, 2, \dots, RA$) based on their functionality with different levels of capacity it can support (Cp_j). In order to ramp capacity, one or multiple pieces of each type of equipment are necessary. For example in Figure 18, four types of machines are needed to support a capacity ramp. Activity 1 represents one piece of equipment belong to ramp group 1 with production capacity 200 WSPW while activity 2 represents two pieces of

equipment belong to the same ramp group with a total of 400 WSPW. Finishing activities 3 and 4 means ramping two piece of equipment from ramp group 2 each piece of equipment can support 200 WSPW production capacity. If from time t_{start} to time t_{end}, activities 1, 2, 3 and 4 in the Install/Qual project are finished, the incremental capacity during that time period can be calculated as $\min\{400 + 200, 200 + 200\} = 400$ WSPW. Therefore, the capacity (Cp^t) at time t is determined by the minimum capacity can support by all ramp groups.

$$Cp^t = min_{ra}\left\{\sum_{C_j \le t}\sum_{j \in ra} Cp_j\right\}, \forall j \in N, ra = 1,2,\dots,RA \quad (1)$$

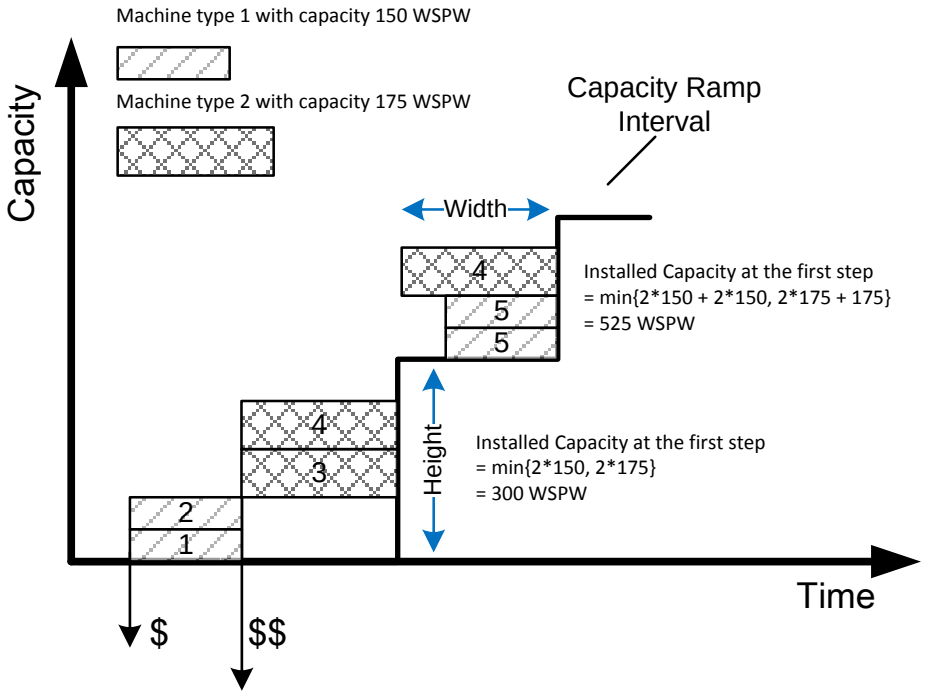

Figure 18: Install/Qual Ramp Step Width and Height

2.2 Product Market Demand and Market Price

In practice, the market demand can be potentially influenced by many factors including product release date, price, quality, competition, the overall economic environment, etc. The Geometric Brownian Motion (GBM) process is most commonly and widely accepted model for the growth in stock price over time (Marathe and Ryan, 2005). To deal with demand uncertainty, research including Whitt (1981), Pindyck (1988), Benavides et al (1999), Tsay (2002), Ryan (2004), Marathe and Ryan (2005), and Chou et al. (2007) as well as many other research effort use the GBM process to model demands in future time periods as a lognormal distribution. Let D_t be a stochastic process that represents customer demand at time t and assume the expected drift rate is μD_t for some constant parameter μ that is independent of D_t. This means in a short interval of time Δt, the expected change in D is $\mu D_t \Delta t$. Let σ be the variance parameter that models the stochastic component of demand volatility, the rate of change of demand can be written as:

$$\frac{dD_t}{D_t} = \mu dt + \sigma dW_t \quad (2)$$

where W_t is a Wiener process or Brownian motion process such that $dW_t = \varepsilon_t \sqrt{dt}$ when ε_t represents independent identically distributed normal random variables with mean zero and standard deviation of one. The formula above indicates that demand variability increases linearly as the horizon of demand forecast increases which is intuitive since the further away the demand forecast is, the higher the variance. Given specific values of μ

and σ, the estimated demand for future time periods can be simulated with Monte Carlo simulation.

After a piece of equipment is installed and qualified, there is a fixed lead-time time (LT) to represent the period from when capacity Cp^t is ramped at time t to wafers being produced and ready for sale at time $t + LT$ for market price Pr^{t+LT}. If that amount of product can all be sold at the current market price (demand is more than capacity), the total expected revenue is $Cp^t \cdot Pr^{t+\mathrm{LT}}$. Under the assumption that demand uncertainty is modeled as a GBM process, the uncertainty of market price and its relationship on demand can be modeled using price elasticity of demand. Research including Carruth et al. (2000) and Chen (2012) model market price at time t with demand function as

$$Pr_t = \theta_t D_t^{-\left(\frac{1}{\varepsilon}\right)} \tag{3}$$

where D_t represents demanded quantity and θ_t represents the uncertainty factor that follows a GBM process. The GBM process can be written as $\frac{d\theta_t}{\theta_t} = \alpha dt + \beta dz_t$ where α is the instantaneous growth rate of θ_t, β is the volatility rate, and dz_t is a standard Brownian motion. In the previous formula, constant value ε represents the price elasticity of demand which is defined as the ratio of relative demand change to relative price change: $\varepsilon = \frac{\Delta D_t / D_t}{\Delta Pr_t / Pr_t}$. Dolan and Simon (1996) summarize empirical estimates of price elasticity for standard industrial products that range between 2 and 100. For Dynamic random-access memory (DRAM) products, Dick (1991), Flamm (1996) and Park (2001) estimated the price elasticity of demand is between -1.5 and -2.0 and very close to -1.8

for several types of products. Chien, Chen and Peng (2010) use -9.32663 as the average price elasticity for semiconductor products in their case study but mentioned that this estimation is on the high end. More discussion on price elasticity of demand can be found in Leachman and Ding (2007). It is worth mentioning that the approach to model uncertain market demand and price is mainly targeted for simulating different scenarios for the Install/Qual process and other reasonable approaches can be easily incorporated in this work in potential future research efforts.

2.3 Activity Cost and Resource Cost

In the Install/Qual process, multiple types of costs and payments exist. For capital equipment, it is a one-time payment (capital investment) paid when a piece of equipment is delivered to the company. Thus, an activity cost Ac_j is assumed to be paid at the start time S_j of every physical installation activity j in the Install/Qual process which represents the acceptance of a piece of equipment. This assumption can be generalized in multiple ways to be able to apply in other project scheduling scenarios. For example, all activities in a project can have project costs instead of just a subset of activities; second, the one-time payments occur at a given time period before or after the start of the activity; third, there could be multiple fixed and pre-negotiated payments instead of just one, or capital equipment can be rented or leaded instead of purchased. In addition to activity cost, resources consumption also incurs costs in the Install/Qual process. To the contrary

of activity cost, resource cost ($t \cdot r_{jk}^{m} \cdot Rc_k$) is based on both the consumption of resources ($t \cdot r_{jk}^{m}$) as well as fixed resource unit cost Rc_k. The payment method of resource cost is time-based, e.g. salary. Thus, the cost of resource consumption is assumed to be paid in each period where there is resource consumption and the daily rate is used since the minimum time unit is days in the Install/Qual process.

2.4 Mathematical Formulation

Decision variables in this problem include $y_j^m = 1$ if activity $j \in N$ is being processed in mode $m \in Mod_j$ and 0 otherwise; $x_{jt}^m = 1$ if activity $j \in N$ is being processed in mode $m \in Mod_j$ at time $t = 1, 2, \dots, T$ and 0 otherwise; resultant variables S_j and C_j represent the start time and completion time of activity j, respectively. The end date (IQ_{end}) of the Install/Qual process determines the relative relation in time among the capacity ramp with market demand and market price and therefore is a decision variable as well. Additional decision variables include $o_{jt} = 1$ if $t \leq C_j$ and 0 otherwise, $q_{jt} = 1$ if $t \geq S_j$ and 0 otherwise to indicate whether a time period t is between the start time S_j and the completion time C_j of activity j. An indicator function $\gamma_{jkt}^m = 1$ if $\gamma_{jkt}^m \in [0, U_{kt}]$ and 0 otherwise specifies whether an activity j in mode m is feasible to process at a certain time period. The total profit (TP) equals the difference between total product market revenue (RE) and total activity cost (AC) and total resource usage cost (RC) which are defined as below.

$$RE = \begin{cases} 0 & t < LT \\ \sum_{t=1}^{T}(Cp^{t-LT}) \cdot Pr^t \cdot (1+e)^{-t} & t \geq LT, \text{if } Cp^t \leq Dd^t \\ (\sum_{t=1}^{T} Dd^t \cdot Pr^t + \sum_{t=1}^{T}(Cp^{t-LT} - Dd^t) \cdot Pr^t \cdot dis)(1+e)^{-t} & t \geq T, \text{if } Cp^t \geq Dd^t \end{cases} \quad (4)$$

$$AC = \sum_{j \in N} Ac_j \cdot (1+e)^{-S_j} \quad (5)$$

$$RC = \sum_{j \in N} \sum_{t=1}^{T} \sum_{k \in RES^n} \sum_{m \in Mod_j} x_{jt}^m \cdot t \cdot r_{jk}^m \cdot Rc_k \cdot (1+e)^{-t} \quad (6)$$

The definition for market revenue assumes excessive inventory can be sold at a discounted price. To appropriately include market revenue into the MIP formulation, a binary variable u^t is introduced such that $u^t = 1$ if $Cp^t \geq Dd^t$, and 0 otherwise. Thus the total product market revenue equation can be rewritten as:

$$RE = \sum_{t=1}^{T}(Cp^{t-LT}) \cdot Pr^t \cdot (1+e)^{-t} \cdot u^t + (\sum_{t=1}^{T} Dd^t \cdot Pr^t + \sum_{t=1}^{T}(Cp^{t-LT} - Dd^t) \cdot Pr^t \cdot dis)(1+e)^{-t} \cdot (1-u^t), \ \forall t = 1, 2, \dots T \quad (7)$$

$$M \cdot u^t \geq Cp^t - Dd^t + 1, \ \forall t = 1, 2, \dots T \quad (8)$$

$$M(1-u^t) \geq Dd^t - Cp^t, \ \forall t = 1, 2, \dots T \quad (9)$$

With the help of decision variable o_{jt}, the MIP representation of the ramped capacity is defined as below. At each given time period t, the production capacity equals to the minimum of the ramped capacity for each ramp group ra which is the total ramped capacity for all machines belongs to that ramp group.

$$Cp^t = \min_{ra}\left(\sum_{j \in ra} Cp_j \cdot \left(1 - o_{jt}\right)\right), \ \forall j \in N, ra = 1,2, \dots, RA, t = 1, 2, \dots, T \quad (10)$$

$$Cp^t \leq \left(\sum_{j \in ra} Cp_j \cdot \left(1 - o_{jt}\right)\right), \ \forall j \in N, ra = 1,2, \dots, RA, t = 1, 2, \dots, T \quad (11)$$

Therefore, the mixed-integer programming formulation is provided below.

$$\max TP \tag{12}$$

Subject to:

$$\sum_{m\in Mod_j} y_j^m = 1,\ \forall j \in N \tag{13}$$

$$\sum_{t=1}^{T} x_{jt}^m = p_j^m \cdot y_j^m,\ \forall j \in N, m \in Mod_j \tag{14}$$

$$C_i \le S_j - 1,\ \forall arc(i,j) \in A \tag{15}$$

$$S_j \le x_{jt}^m \cdot t + M\left(1 - x_{jt}^m\right),\ \ \forall j \in N, m \in Mod_j, t = 1, 2, \dots, T \tag{16}$$

$$C_j \ge x_{jt}^m \cdot t,\ \ \forall j \in N, m \in Mod_j, t = 1, 2, \dots, T \tag{17}$$

$$S_j \ge rad_j,\ \forall j \in N \tag{18}$$

$$C_j \le due_j,\ \forall j \in N \tag{19}$$

$$\sum_{j\in N} \sum_{m\in Mod_j} r_{jk}^m \cdot x_{jt}^m \le U_{kt},\ \ \forall k \in RES^r, t = 1, 2, \dots, T \tag{20}$$

$$\sum_{j\in N} \sum_{m\in Mod_j} r_{jk}^m \cdot y_j^m \le U_k,\ \ \forall k \in RES^n \tag{21}$$

$$M \cdot o_{jt} \ge C_j - t + 1,\ \forall j \in N, t = 1, 2, \dots T \tag{22}$$

$$M \cdot \left(1 - o_{jt}\right) \ge t - C_j,\ \forall j \in N, t = 1, 2, \dots T \tag{23}$$

$$M \cdot q_{jt} \ge t - S_j + 1,\ \forall j \in N, t = 1, 2, \dots T \tag{24}$$

$$M \cdot \left(1 - q_{jt}\right) \ge S_j - t,\ \forall j \in N, t = 1, 2, \dots T \tag{25}$$

$$o_{jt} + q_{jt} \ge 1,\ \forall j \in N, t = 1, 2, \dots, T \tag{26}$$

$$x_{jt}^m \ge y_j^m + \gamma_{jkt}^m + o_{jt} + q_{jt} - 3,\ \forall j \in N, m \in Mod_j, \forall k \in RES^n, t = 1, 2, \dots T \tag{27}$$

$$4 \cdot x_{jt}^m \le y_j^m + \gamma_{jkt}^m + o_{jt} + q_{jt},\ \forall j \in \mathrm{N}, m \in Mod_j, \forall k \in RES^n, t = 1, 2, \dots T \tag{28}$$

$$TP = RE - AC - RC \tag{29}$$

$$C_j \le IQ_{end},\ \forall j \in N \tag{30}$$

$$RC = \sum_{j\in N}\sum_{t=1}^{T}\sum_{k\in RES^n}\sum_{m\in Mod_j} x_{jt}^{m}\cdot t\cdot r_{jk}^{m}\cdot Rc_k\cdot(1+e)^{-t} \quad (31)$$

$$AC = \sum_{j\in N} Ac_j\cdot(1+e)^{-S_j} \quad (32)$$

$$RE = \sum_{t=1}^{T}(Cp^{t-LT})\cdot Pr^t\cdot(1+e)^{-t}\cdot u^t + (\sum_{t=1}^{T} Dd^t\cdot Pr^t + \sum_{t=1}^{T}(Cp^{t-LT} - Dd^t)\cdot Pr^t\cdot dis)(1+e)^{-t}\cdot(1-u^t),\ \forall t = 1,2,\dots T \quad (33)$$

$$M\cdot u^t \geq Cp^t - Dd^t + 1,\ \forall t = 1,2,\dots T \quad (34)$$

$$M(1-u^t) \geq Dd^t - Cp^t,\ \forall t = 1,2,\dots T \quad (35)$$

$$Cp^t \leq \left(\sum_{j\in ra} Cp_j\cdot\left(1-o_{jt}\right)\right),\ \forall j\in N,\ ra = 1,2,\dots,RA,\ t = 1,2,\dots,T \quad (36)$$

$$y_j^m \in \{0,1\},\ \forall j\in N, m\in Mod_j \quad (37)$$

$$x_{jt}^m \in \{0,1\},\ \forall j\in N, m\in Mod_j, t = 1,2,\dots,T \quad (38)$$

$$o_{jt}\in\{0,1\},\ \forall j\in N, t = 1,2,\dots T \quad (39)$$

$$q_{jt}\in\{0,1\},\ \forall j\in N, t = 1,2,\dots T \quad (40)$$

$$u^t\in\{0,1\},\ \forall t = 1,2,\dots T \quad (41)$$

The objective function (12) maximizes the total profit in the Install/Qual process. Constraint sets (13) – (28) are basic MRCPSP formulation for the Install/Qual scheduling problem discussed in Cheng et al. (2014). Constraint set (13) ensures only one mode can be selected for each activity. Constraint set (14) ensures that if mode m is selected for activity j, the total processing time must equal the corresponding duration. Constraint sets (15) – (17) are precedence constraints. Constraint sets (18) – (19) ensure ready times and due dates are not violated. Constraint sets (20) – (21) ensure resource availability for both renewable resources and non-renewable resources. Constraint sets (22) – (25) are

included to support the new decision variables o_{jt} and q_{jt}. Constraint set (26) ensures the activity completion time is no earlier than the start time for activity j. Constraint sets (27) – (28) ensure an activity j cannot be preempted at time t if resources are available. Constraint sets (29) – (36) are additional constraints for market price and demand. Constraint (29) sets the value of the objective function; constraint set (30) represents the end date of the Install/Qual process; constraint set (31) calculates the resource cost, constraint set (32) calculates the activity cost (capital equipment investment); constraint sets (33) – (35) define the product market revenue; constraint set (36) is the ramped capacity calculation. Constraint sets (37) – (41) are the non-negativity and binary constraints. A big number M in the MIP formulation is set to be the maximum project planning horizon T.

3. Methodology

In this research, optimization and simulation are integrated together in a combined scheduling framework. A case study of an early version of this approach is discussed in Cheng et al. (2012). The optimization module is used to search for good Install/Qual schedules and capacity ramp strategies while the simulation module evaluates the solution quality under uncertain market price and market demand. Simulated Annealing (SA) with priority rule-based simple heuristics is selected as the optimization algorithm. SA exploits an analogy between the annealing process and the

search for the optimum in a more general system. Each step of the SA attempts to replace the current solution by a random "neighborhood" solution. The new solution may be accepted with a probability that depends both on the difference between the corresponding target values and the current "temperature". The SA is selected over other meta-heuristics (e.g. GA) because the "temperature" parameter is a straightforward threshold parameter to decide when to start simulating candidate solutions. Cheng et al. (2014) also demonstrates that SA can be a good meta-heuristic candidate for the Install/Qual scheduling problem. Monte Carlo simulation is selected to simulate solutions with different market price and market demand scenarios. For a given Install/Qual schedule, *max_sim* number of different market demand and market price scenarios are simulated and evaluated. The expected profit is calculated as the average performance of a particular Install/Qual schedule under uncertain market information.

3.1 Solution Encoding Scheme and Schedule Generation Scheme

An activity loading list is defined to represent an Install/Qual schedule. A random key-based solution encoding scheme $SOL = \{MOD, RK\}$ includes a mode assignment vector $MOD = \{m_j | m_j \in Mod_j,\ j \in N\}$ and a random key assignment vector $RK = \{RK_j | RK_j \in (0,1),\ j \in N\}$. The mode assignment key m_j specifies which processing mode is selected for activity j, while random key RK_j represents the relative priority of an activity compared to other activities when they are all available to be scheduled next.

A detailed discussion on this solution encoding scheme can be found in Cheng et al. (2014). In this research, a serial and a backward schedule generation scheme (SGS) are selected in which activities are selected one-by-one and loaded in backward manner from their due date. SGSs are mechanisms that translate an encoded solution to a schedule. By reversing the precedence network and activity ready and due dates, a backward schedule generation approach follows the same procedure of commonly studied forward schedule generation approaches.

3.2 Initial Solution Generation

Priority rule-based simple heuristics combine a mode selection rule and an activity priority rule. A mode selection rule determines which mode to choose from when multiple processing modes are available. Common mode selection rules such as shortest duration mode (SDM) and least total (renewable) resource usage (LTRU_R) are adopted in this work. An activity priority rule defines the relative priority sequence to select activities from a list of activities waiting to be scheduled. The activity priority rules selected for this work are: minimum slack (MSLK), minimum latest finish time (MLST), modified minimum slack (MMSLK), greatest rank positional weight (GRPW), shortest processing time (SPT) and longest processing time (LPT). Since the computational effort for all combinations of mode selection rules and activity priority rules is minimal, the initial solution generator uses all 12 combinations (2 mode rules, 6 activity rules) and

selects the best schedule as the initial solution for the simulated annealing algorithm.

3.3 Simulated Annealing

The neighborhood solution in simulated annealing is defined as a solution that is obtained by modifying the current solution key multiple ($NBsearch$) times. At each time, an activity is randomly selected and its mode assignment key m_j is randomly modified $m_j^{'}, m_j^{'} \in Mod_j$, and two non-adjacent activities replace their activity priority key: $RK_i^{'} = RK_j, RK_j^{'} = RK_i,\ i,j \in N$. Activities *i* and *j* are randomly selected but need to be non-adjacent for the reason that adjacent activities are highly likely to have a precedence relation and the activity priority key does not impact the scheduling sequence of activities that have a precedence relation. Pseudo code of the algorithm can be found below.

Simulated Annealing Pseudo Code

Step 1: Initialization
    Generate initial solution $i_{initial}$
    Update the current solution as the initial solution $i_{current} = i_{initial}$
    Set initial temperature $Tem_{initial}$, freezing temperature Tem_{freeze}
    Set temperature to start simulation Tem_{sim} (*threshold gate2*)
    Set current temperature as the initial temperature $Tem_{current} = Tem_{initial}$
    Set cooling ratio $cool\_ratio$
    Calculate the fitness value $fit_{current}$ for $i_{current}$
    Initial the best fitness value $fit_{best} = fit_{current}$
    Go to step 2
Step 2:
    Move from the current solution $i_{current}$ to a neighborhood solution $i_{neighbor}$
    Decide whether to initiate the simulation module, if $Tem_{current} \leq Tem_{sim}$
        Go to step 4
    Else

Go to step 3

Step 3:

Schedule the Install/Qual process and return total profit as $fit_{neighbor}$

If the neighborhood solution is better than the current solution, $fit_{neighbor} \geq fit_{current}$

accept the neighborhood solution as current: $i_{current} = i_{neighbor}$

update the current fitness value $fit_{current} = fit_{neighbor}$

update the best fitness value $fit_{best} = fit_{neighbor}$

else

check the current temperature $Tem_{current}$, and acceptance function

if $random \geq \exp\left(\frac{fit_{neighhor} - fit_{best}}{Tem_{current}}\right)$

accept the neighborhood solution $i_{current} = i_{neighbor}$

else

reject the neighborhood solution

keep the current solution

Go to step 5

Step 4:

Apply ***Monte Carlo Simulation*** Module

Go to step 5

Step 5:

update the current temperature $Tem_{current} = Tem_{current} * cool\_ratio$

Go to Step 6

Step 6:

check the termination rule

If $Tem_{current} < Tem_{freeze}$

stop the cooling process

return the $fit_{current}$ ($\overline{fit}_{current}$) and fit_{best} ($\overline{fit}_{best}$)

else

Go to step 2

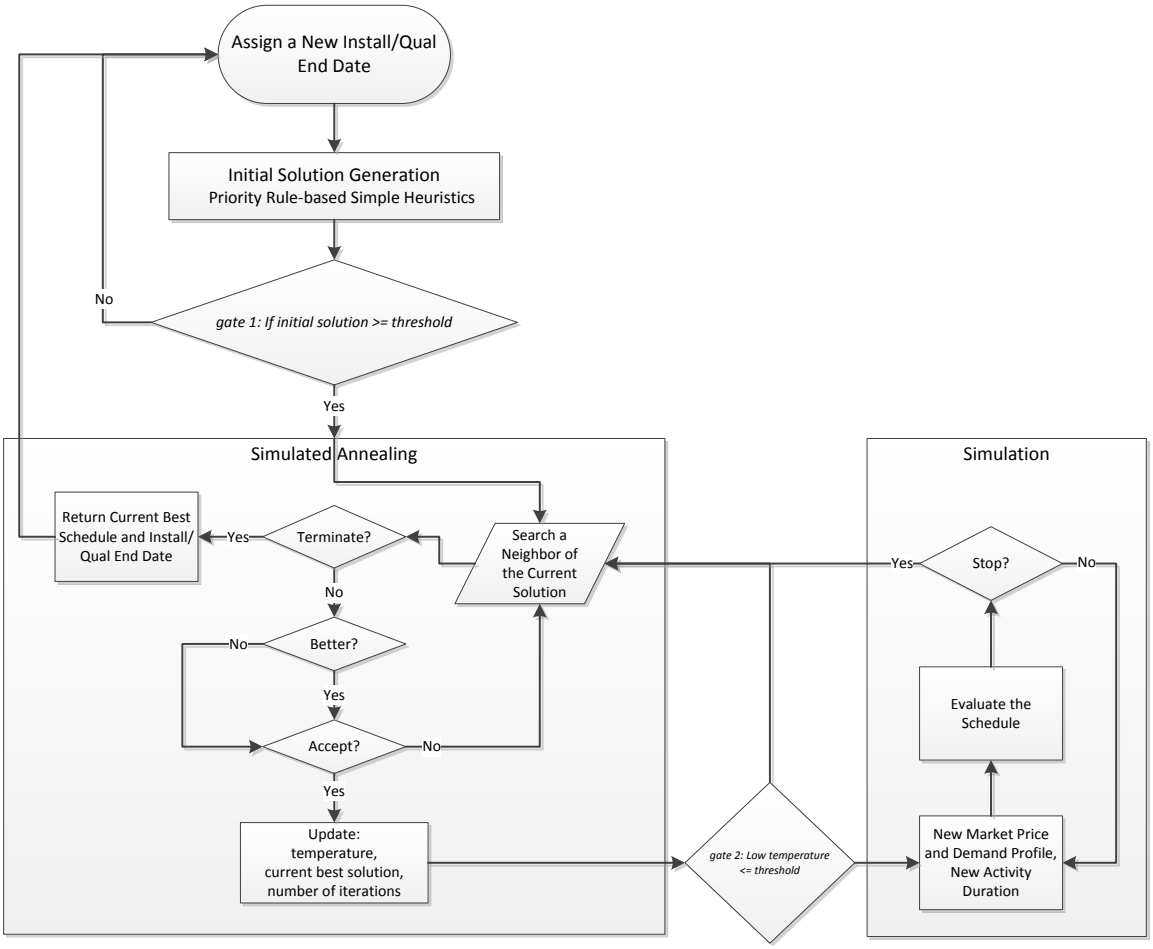

Figure 19: Schedule Algorithm Logic Flow

The overall Install/Qual scheduling algorithm flow can be found in Figure 19. There are two threshold values embedded in the algorithm. The first threshold *gate1* indicates whether the initial solution is good enough for applying simulated annealing to improve; the second threshold *gate2* applies to the "temperature" parameter of the SA algorithm to determine when to simulate a solution. After one SA terminates, the algorithm returns the current schedule and then moves to a new Install/Qual end date. The number of different Install/Qual end dates depends on the length of the total decision

window. Based on the practical experience of the Install/Qual process, a roughly 6 month (day 868 to day 1049) decision window where the end date of the Install/Qual process falls is reasonable. This means that based on the available information for market prediction, the wafer fab needs to be ramped to full capacity in that 6 month period. Or in other words, the "optimal" date to fully ramp the fab will not fall out of that 6 month time window. More computational effort is required if the time window is longer. The second assumption of time intervals for two adjacent Install/Qual end date is 30 days. Operationally it makes sense to focus on granularity of one month for a project in 2-3 years in advance. Thus, for each tested instance, there are 6 candidate solutions for the Install/Qual end date that will be explored and evaluated. With a shorter time interval between adjacent Install/Qual end dates, more computational effort is required since there are more candidate solutions that need to be optimized and evaluated.

4. Computational Experiment

Two computational experiments are described in this section. In the first experiment, the set of capital equipment that needs to go through the Install/Qual process is assumed to be fixed. That is, the final ramped capacity of the wafer fab is not a decision but is an input. Instead, only the Install/Qual schedule and when to reach the maximum capacity need to be decided with the objective to maximize expected profit. In the second experiment, we assume a baseline number of necessary equipment is given

and there is a subset of equipment as an "option" that we can add/remove to expand/reduce maximum production capacity if needed. However, due to the long lead time of ordering these equipment, the decision of adding extra capacity needs to be made at the beginning of the planning stage. In both experiments, we compare the approach of scheduling based on expected market information (mean demand and mean price) with the proposed approach that relies on both optimization and simulation.

Tested instances are designed to be at the same scale of the practical Install/Qual process but does not use actual data for confidentiality reasons. Therefore, the computational results can only be interpreted directionally. First, 500 pieces of major production equipment are assumed and each piece of equipment has 3 Install/Qual activities (physical installation, supplier qualification, company qualification) so that each problem instance has 1500 activities. Second, each piece of capital equipment specifies the equipment supplier, cost and arrival time and due date window. Third, there are a total of 5 suppliers and each piece of equipment is randomly assigned to one of the 5 suppliers. Finally, the cost of these 500 pieces of production equipment are randomly generated according to a distribution so that 50% of capital equipment cost range $0.1M ~ $5M; 45% of them cost range between $10M ~ $40M and the remaining 5% cost range between $60M ~ $100M. Initial demand at time 0 is assumed to be 3,000 wafers per week with the starting price $12,000 per wafer. Baseline demand drift is assumed to be 25% per year and demand volatility is 20% per year. In the experiment, three levels (high,

baseline, low) of demand drift and volatility factors are considered. The uncertainty of price has the same volatility (20%) but negative drift (-25%) and price elasticity of demand $\varepsilon = 2$. Detailed parameter settings can be found in Table 25.

Table 25: Computation Experiment Overview

	Experiment I	Experiment II
The set of Install/Qual activities	Fixed	Decision variable
Final capacity (WSPW)	6,000	5000, 6000, 7000
Install/Qual schedule	Decision variable	
Market demand	Uncertain	
Market price	Uncertain	
$\|N\|$	1500	
$\|Mod_j\|$	1~3	
$\|RES^r\|$	7 (trades, 5 suppliers, company resource)	
$\|RES^n\|$	2 (budget for trades, budget for suppliers)	
Threshold *gate1* & *gate2*	tight, baseline, loose	
Demand drift	30% (high), 25% (baseline), 20% (low)	
Demand volatility	25% (high), 20% (baseline), 15% (low)	
Price drift	- 25% (baseline)	
Price volatility	20% (baseline)	
# Tested Instances	20	

Regarding the final fab capacity, Experiment I assumes the final capacity is 6,000 WSPW while Experiment II considers three possible capacity investment scenarios: 5,000, 6,000, 7,000 WSPW. As for capacity ramp, there are as many as 50 different types of equipment and each equipment between 4 and 20 instances of each. The number of equipment groups is close to the actual Install/Qual process. At least one piece of equipment is of each type needed to produce a wafer but additional pieces might not be needed extra for a ramp step. The "Ramp height" for each group of tools can be 200, 400,

600 or 800 WSPW: e.g. equipment group 1 has two pieces of equipment each with 200 WSPW, equipment group 2 has one piece of equipment with 800 WSPW, and the remaining equipment groups all have one piece of equipment with capacity of 600 WSPW. This ramp volume would be $\min\{200 \cdot 2, 800, 600, \ldots, 600\} = 400$ WSPW.

For each piece of equipment, only the physical installation process in the Install/Qual process has alternative processing modes, other activities only have one processing mode. There are total 7 renewable resource types: one type of trades resource that work 4 days/week and 10 hours/day, 5 types of equipment supplier resources that work 5 days/week and 8 hours/day, one type of company resource that work 7 days/week and 24 hours/day with multiple shifts. The manufacturing lead time is assumed to be $LT = 60$ days and the relative time horizon we study is about 40 months. If the produced wafers match the market demand, all produced wafers can be sold at the current market price. If produced capacity is higher than the market demand, the extra product will incur inventory holding costs. We assume the amount of extra product can be sold with $dis = 50\%$ of the current market price. Unmet market demand is assumed to be lost. For time value of money consideration, a 6% annual interest rate is assumed on capital equipment cost, resource usage cost and product selling revenue. The SA algorithm and simulation are programmed in C++ (http://www.microsoft.com/visualstudio). The experiments were conducted on a laptop with Intel ® Core ™ i5-2520M CPU @ 2.50GHz, 4.00GB installed memory, the Windows 7 Enterprise 64-bit Operating System.

4.1 Experiment I – Fixed Final Capacity

In the first experiment, the final fab capacity level is fixed but when to reach to the maximum capacity is a decision variable. First of all, the expected profit impact of different threshold levels for simulation is shown in the results in Figure 20.

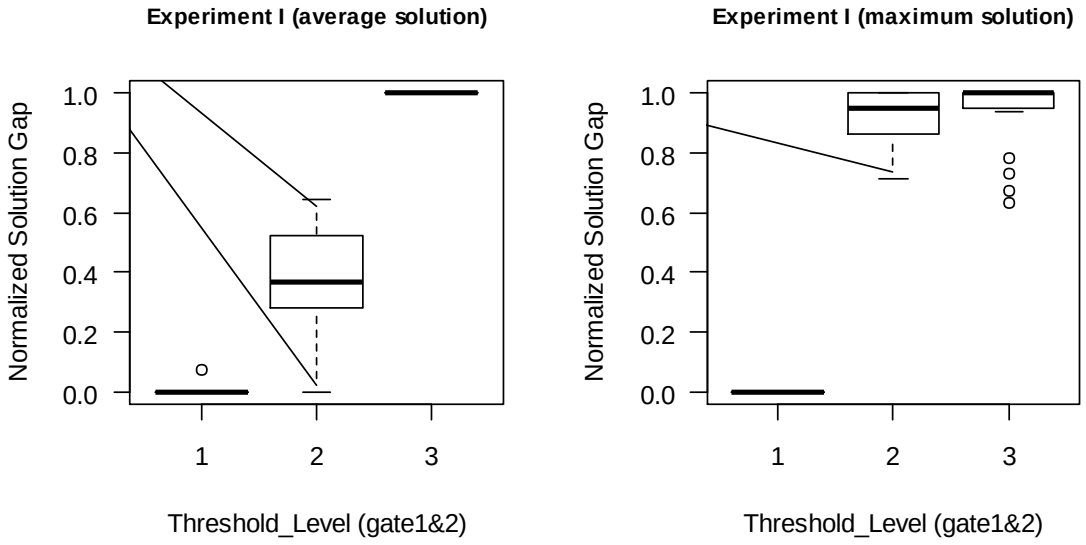

Figure 20: The Impact of Different Threshold Levels on Normalized Solution Gap for Both Average Solution and Maximum Solution (1=Tight, 2=Baseline, 3=Loose)

Threshold *gate1* decides how early an Install/Qual end date can pass the initial optimization barrier. *Gate2* determines when to stop optimizing the deterministic solution and start integrating simulation. Both threshold values trade off computational time with potential solution quality. From "Tight" to "Baseline" of *gate1* and *gate2*, more candidate solutions pass the barrier of being optimized and simulated and that increases the

possibility of finding a better solution as well as the required computational effort. However, when *gate1* and *gate2* go from "Baseline" to "Loose", more of the "less promising" solutions are allowed to be optimized and simulated but they are still not as good as the best solution. Thus from Figure 20, as the threshold value changes from "Tight" to "Loose", the average solution and computational time always improve, but the best solution is only improved from "Tight" to "Baseline", but remained almost the same from "Baseline" to "Loose". If in practice only the best Install/Qual ramp process is the concern, the "Baseline" threshold level can achieve the target with minimal computational effort.

The second portion of this result is to show the benefit of using simulation under different levels of uncertainty. "non-sim" results are Install/Qual solutions optimized with only static information (with demand drift and price drift = 0) while "sim" solutions are Install/Qual schedules that with both simulation and optimization. Results in Figure 21 show that with the help of simulation, better capacity ramp strategies can be found than without simulation ("non-sim"). Also, the impact of simulation increases as the demand volatility factor increases. This is intuitive since smaller demand volatility factor means less uncertainty of future demand. The extreme case of zero demand volatility is essentially assuming demand with steady drift trend and in that case "non-sim" is the same as "sim".

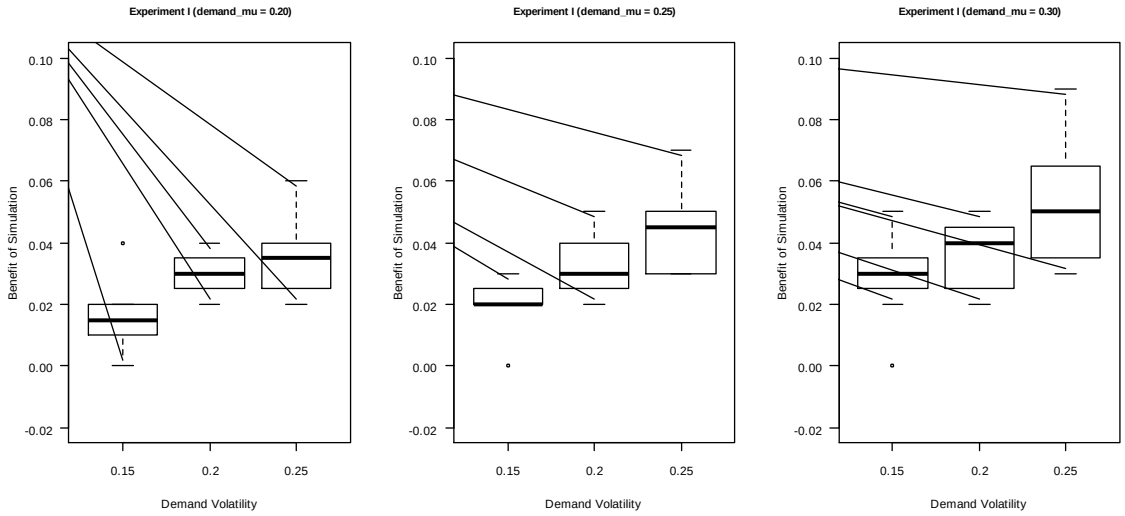

Figure 21: The Benefit of Combining Simulation with Optimization Compared to Without Simulation on Total Profit at Different Demand Trend and Volatility Levels

4.2 Experiment II – Uncertain Final Capacity

In the first experiment, the final capacity of the fab (6,000 WSPW) is assumed to be fixed so that capacity investment is not a decision variable. However in Experiment II, there are three levels of final capacity estimation: pessimistic (5,000 WSPW), realistic (6,000 WSPW) and optimistic (7,000 WSPW) that need to be decided at the beginning of the Install/Qual process. Results are shown in Figure 22. Assuming 6,000 WSPW is the baseline capacity investment strategy that roughly equals to expected projected demand with $\mu = 0.25$, the pessimistic (5,000 WSPW) scenario and the optimistic (7,000 WSPW) scenario roughly match with $\mu = 0.20$ and $\mu = 0.25$, respectively. Compared to the baseline case, a set of equipment supporting one ramp step with 1,000 WSPW can be added or removed to adjust the final capacity investment. Results are shown in Figure 22

with several observations. First, with given capacity investment, the expected profit increases as the demand drift μ increases. This is straightforward since higher demand indicates higher profit regardless of the amount of capacity investment when the opportunity cost of losing sales is not considered. Second, when demand volatility is low ($\sigma = 0.15$), highest expected profit scenario is achieved when invested capacity matches with the project demand drift trend μ which is intuitive. Third, when demand volatility σ increases to $\sigma = 0.20$ or $\sigma = 0.25$, over investing in capacity is preferable to under investing in capacity. This is true since the cost of idle assets is less than the cost of losing sales. Therefore, higher capacity is preferred to cover the high level of demand variability.

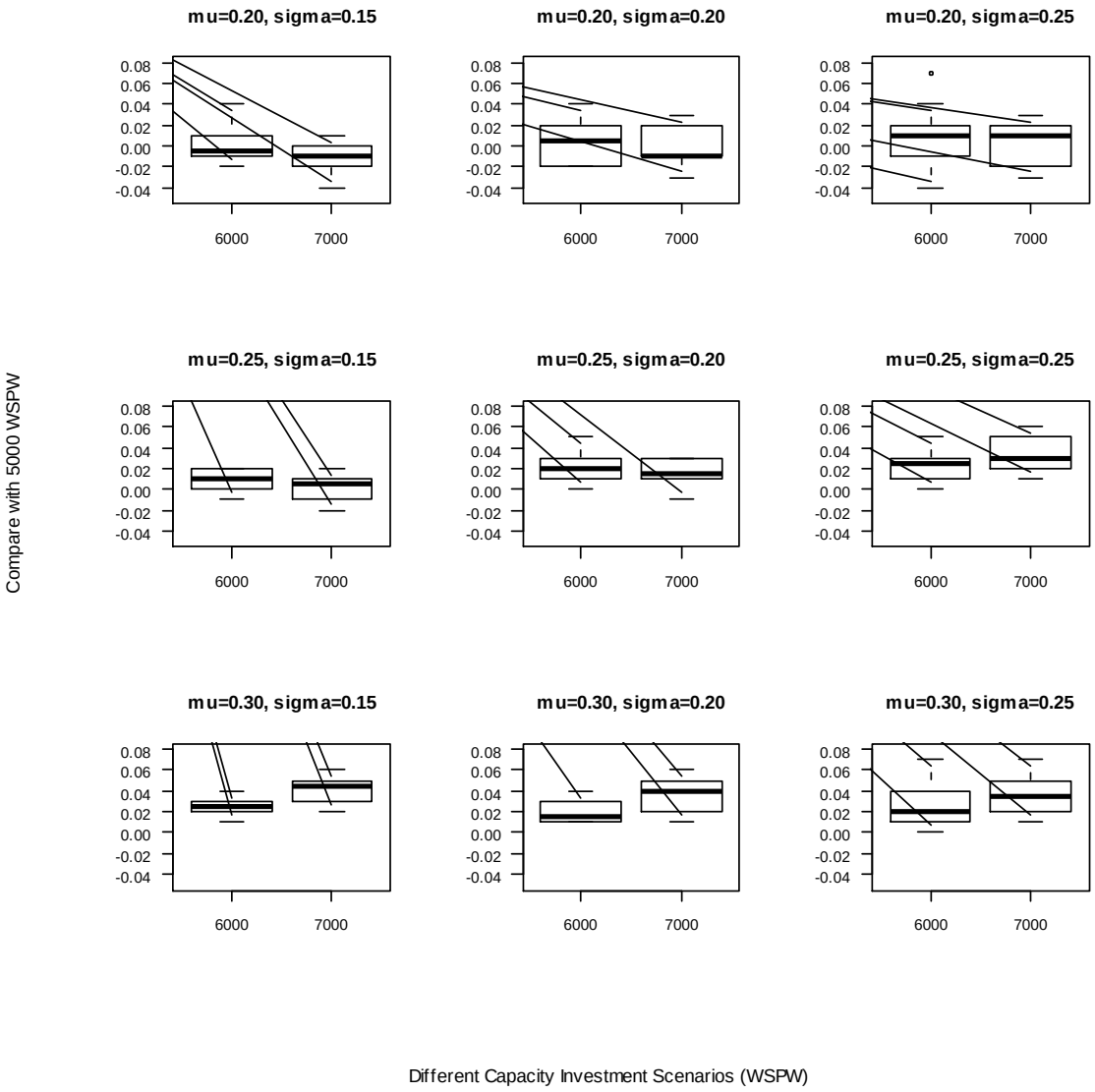

Figure 22: The Benefit of Different Capacity Investment Scenarios Comparing to 5000 WSPW at Different Demand Trend and Volatility Levels

5. Conclusion and Future Research

In this research, a scheduling framework is proposed to approach the semiconductor capital equipment Installation and Qualification process under both static and uncertain market information. The proposed framework integrates an optimization

module using Simulated Annealing with a simulation module using Monte Carlo simulation to search for better solutions. It is shown that with careful threshold level settings, good quality results can be found with reasonable computational effort. With the help of simulation, SA can improve the expected value of a solution instead of the deterministic solution and it outperforms the scheduling approach with only optimization. The benefit of integrating simulation increases as the demand volatility level increases. Further, if capacity investment becomes a decision variable, matching capacity with expected demand is recommended when demand volatility level is low while over capacity is recommended when demand volatility level is high. There are at least several possible future research directions beyond this work. First, the uncertainty studied in this research work focuses on market demand and price and can be extended to consider uncertain activity duration and equipment arrival times. Second, different negotiation strategies with equipment suppliers can be studied since when demand is highly uncertain, it might be beneficial to delay investment decisions until a certain time period to wait for better market information. Peng et al. (2012) is one example of such work.

CHAPTER 5 CONCLUDING REMARKS

In this dissertation, mathematical-based analytical methodologies are proposed to approach the semiconductor capital equipment installation and qualification scheduling problem. Mathematical programming, a branch-and-bound algorithm, priority rule-based simple heuristics and meta-heuristics are proposed and discussed to analyze different versions of the Install/Qual scheduling problem with multiple practical considerations.

This dissertation contains three main phases of research efforts. In phase 1, the Install/Qual scheduling problem is formulated as a multi-model resource-constrained project scheduling problem with minimizing project makespan as the objective. Multiple practical extensions are considered such as multiple processing modes, time-varying resource constraints and resource vacations. Special attention is paid to the difference between activity splitting, preemption and non-preemptive activity splitting. A precedence tree-based branch-and-bound algorithm is proposed to solve small size academic problem instances to optimality. Computational experiments show activity splitting can bring significant project makespan reduction and non-preemptive activity splitting instead of preemption is main reason for that. The tighter the resource limits are and the higher range of resource limits vary, the bigger the makespan reduction is.

Due to the NP-hardness nature of the problem, exact methods can only solve very small size problem instances. The second phase of the dissertation studies both simple

heuristics and meta-heuristics to be able to solve larger size problem instances within a reasonable amount of computational time. Priority rule-based simple heuristics and a modified random key-based genetic algorithm are used along with project decomposition to solve both small and large size problem instances. Heuristic-based scheduling approaches can find reasonable solutions for much less computational time than the exact method. The modified random key-based genetic algorithm outperforms simple heuristics but the solution gap narrows with the increase of resource availability levels. Decomposition makes sense when the proposed decomposition score is low.

In phases 3, a more comprehensive decision support framework for the Install/Qual scheduling problem is proposed. The framework supports decision making in the capacity ramp strategy to maximize profit with consideration of uncertain market information. Priority rule-based simple heuristics are combined with a simulated annealing algorithm to form an optimization module. It then integrates with a Monte Carlo simulation module to search for better capacity ramping process and Install/Qual schedule under uncertain product market price and market demand. Computational results demonstrate that the integration of optimization and a simulation approach outperforms just static approach.

REFERENCES

Alcaraz, J., and Maroto, C. 2001. A robust genetic algorithm for resource allocation in project scheduling. Annals of Operations Research, 102, 83-109.

Alcaraz, J., Maroto, C., and Ruiz, R. 2003. Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms. Journal of the Operational Research Society, 54, 614-626.

Alvarez-Valdes, R., and Tamarit, J. M. 1989. Heuristic algorithms for resource-constrained project scheduling: A review and an empirical analysis. Advances in Project Scheduling, 113-134.

Argoneto, P., Perrone, G., Renna, P., Nigro, G. L., Bruccoleri, M., and La Diega, S. N. 2008. Production planning in production networks: models for medium and short-term planning. Springer Science and Business Media.

Balas, E. 1971. Project scheduling with resource constraints. In Anonymous Applications of Mathematical Programming E.M.L. Beale ed.. The English Universities Press: London.

Ballestin F, Valls V. and Quintanilla S. 2009 Scheduling projects with limited number of preemptions. Computers & Operations Research 36, 2913-2925.

Ballestin, F., Valls, V. and Quintanilla, S. 2008. Pre-emption in resource-constrained project scheduling. European Journal of Operational Research, 189, 1136-1152.

Benavides, D. L., Duley, J. R., and Johnson, B. E. 1999. As good as it gets: optimal fab design and deployment. Semiconductor Manufacturing, IEEE Transactions on, 123, 281-287.

Blanning, R. W. and Rao, A. G. 1965. Communications to the Editor-A Note on "Decomposition of Project Networks". Management Science, 121, 145-148.

Boctor, F. F. 1990. Some efficient multi-heuristic procedures for resource-constrained project scheduling. European Journal of Operational Research, 49, 3-13.

Boctor, F. F. 1996. A new and efficient heuristic for scheduling projects with resource restrictions and multiple execution modes. European Journal of Operational Research, 90, 349-361.

Bomsdorf, F. and Derigs, U. 2008. A model, heuristic procedure and decision support system for solving the movie shoot scheduling problem. OR Spectrum, 304, 751-772.

Böttcher, J., Drexl, A., Kolisch, R., and Salewski, F. 1999. Project scheduling under partially renewable resource constraints. Management Science, 45, 543-559.

Bouleimen, K., and Lecocq, H. 2003. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. European Journal of Operational Research, 149 2: 268-281.

Broadie, M., and Detemple, J. B. 2004. ANNIVERSARY ARTICLE: Option Pricing: Valuation Models and Applications. Management Science, 509, 1145-1177.

Brucker, P., and Knust, S. 2006. Complex scheduling. Springer Verlag, Berlin 2006.

Brucker, P., Drexl, A., Möehring, R., Neumann, K., and Pesch, E. 1999. A Resource-constrained project scheduling: Notation, classification, models, and methods. European Journal of Operational Research, 112, 3-41.

Brucker, P., Drexl, A., Möhring, R., Neumann, K. and Pesch, E, 1999. Resource-constrained project scheduling: Notation, classification, models, and methods, European Journal of Operational Research, 112, 1, 3-41.

Buddhakulsomsiri, J., and Kim, D. S. 2006. Properties of multi-mode resource-constrained project scheduling problems with resource vacations and activity splitting. European Journal of Operational Research, 175, 279-295.

Buddhakulsomsiri, J., and Kim, D. S. 2007. Priority rule-based heuristic for multi-mode resource-constrained project scheduling problems with resource vacations and activity splitting. European Journal of Operational Research, 178, 374-390.

Carruth, A., Dickerson, A., and Henley, A. 2000. What do we know about investment under uncertainty? Journal of Economic Surveys, 142, 119-154.

Chasey, A. and Pindukuri, S. 2012. Information Exchange Requirements for Capital Equipment and Facility Infrastructure for Semiconductor Facilities. Construction Research Congress 2012. 437-446.

Chen, B. 1993. A better heuristic for preemptive parallel machine scheduling with batch setup times. SIAM Journal on Computing, 226, 1303-1318.

Chen, P. Y. 2012. The investment strategies for a dynamic supply chain under stochastic demands. International Journal of Production Economics, 1391, 80-89.

Cheng, J., Fowler, J. and Kempf, K. 2012. Simulation-based multi-mode resource-constrained project scheduling of semiconductor equipment installation and qualification.

Proceedings of the 2012 Winter Simulation Conference WSC, Berlin, Germany.

Cheng, J., Fowler, J., Kempf, K., and Mason, S. 2015. Heuristic-based scheduling algorithms for the semiconductor capital equipment installation/qualification process. Working Paper.

Cheng, J., Fowler, J., Kempf, K., and Mason, S. 2014. Multi-mode resource-constrained project scheduling problems with non-preemptive activity splitting. Computers & Operations Research, 53, 275-287.

Chien, C. F. and Zheng, J. N. 2012. Mini–max regret strategy for robust capacity expansion decisions in semiconductor manufacturing. Journal of Intelligent Manufacturing, 236, 2151-2159.

Chien, C., Chen, Y. and Peng, J. 2010. Manufacturing intelligence for semiconductor demand forecast based on technology diffusion and product life cycle, International Journal of Production Economics, 128, 2, 496-509.

Chiu, H. N. and Tsai, D. M. 1993. A comparison of single-project and multi-project approaches in resource-constrained multi-project scheduling problems. Journal of the Chinese Institute of Industrial Engineers, 103, 171-179.

Chou, Y. C., Cheng, C. T., Yang, F. C., and Liang, Y. Y. 2007. Evaluating alternative capacity strategies in semiconductor manufacturing under uncertain demand and price scenarios. International Journal of Production Economics, 1052, 591-606.

Confessore, G., Giordani, S. and Rismondo, S. 2007. A market-based multi-agent system model for decentralized multi-project scheduling. Annals of Operations Research, 1501, 115-135.

Damay, J., Quilliot, A., and Sanlaville, E. 2007. Linear programming based algorithms for preemptive and non-preemptive RCPSP. European Journal of Operational Research, 182, 1012-1022.

Dangl, T. 1999. Investment and capacity choice under uncertain demand. European Journal of Operational Research, 1173, 415-428.

Davis, E. W. 1966. Resource allocation in project network models - A Survey. The Journal of Industrial Engineering, 17, 177-188.

Davis, E. W. 1973. Project scheduling under resource constraints—historical review and categorization of procedures. IIE Transactions, 5, 297-313.

Davis, E. W., and Patterson, J. H. 1975. A comparison of heuristic and optimum solutions in resource-constrained project scheduling. Management Science, 21, 944-955.

De Reyck, B., and Herroelen, W. 1999. The multi-mode resource-constrained project scheduling problem with generalized precedence relations. European Journal of Operational Research, 119 2: 538-556.

Debels, D., and Vanhoucke, M. 2005. A bi-population based genetic algorithm for the resource-constrained project scheduling problem. Computational Science and Its Applications–ICCSA 2005, 378-387.

Debels, D., and Vanhoucke, M. 2006. Preemptive resource-constrained project scheduling with setup times. Working Papers of Faculty of Economics and Business Administration, Ghent University, Belgium.

Debels, D., and Vanhoucke, M. 2007. A decomposition-based genetic algorithm for the resource-constrained project-scheduling problem. Operations Research, 55, 457-469.

Debels, D., De Reyck, B., Leus, R., and Vanhoucke, M. 2006. A hybrid scatter search/electromagnetism meta-heuristic for project scheduling. European Journal of Operational Research, 169, 638-653.

Deckro, R. F., Winkofsky, E. P., E Hebert, J. and Gagnon, R. 1991. A decomposition approach to multi-project scheduling. European Journal of Operational Research, 511, 110-118.

Demeulemeester, E. L., and Herroelen, W. S. 1996. An efficient optimal solution procedure for the preemptive resource-constrained project scheduling problem. European Journal of Operational Research, 90, 334-348.

Dick, A.R. 1991. Learning by doing and dumpling in the semiconductor industry. Journal of Law and Economics, 34.

Dolan, R. J. and Simon, H 1996. Power Pricing: How Managing Price Transforms the Bottom Line. Free Press, New York.

Drexl, A., and Grüenewald, J. 1993. Nonpreemptive multi-mode resource-constrained project scheduling. IIE Transactions, 25, 74-81.

Dumond, J. and Mabert, V. A. 1988. Evaluating project scheduling and due date assignment procedures: an experimental analysis. Management Science, 341, 101-118.

Flamm, K 1996. Mismanaged Trade? Strategic Policy and the Semiconductor Industry.

Brookings Institution Press, Washington, DC.

Franck, B. and Neumann, K. 1997. Resource-constrained project scheduling with time windows: Structural questions and priority rule methods. Technical Report WIOR-492, Universität Karlsruhe, Germany.

Franck, B., Neumann, K., and Schwindt, C. 2001. Project scheduling with calendars. OR Spectrum, 23, 325-334.

Fung, S. P. 2008. Lower bounds on online deadline scheduling with preemption penalties. Information Processing Letters, 1084, 214-218.

Gen, M., and Cheng, R. 2000. Genetic algorithms and engineering optimization. John Wiley & Sons: New York.

Gen, M., Cheng, R., and Lin, L. 2008. Network models and optimization: Multiobjective genetic algorithm approach. Springer Verlag.

Gonçalves, J. F., Resende, M. G. C., and Mendes, J. J. M. 2010. A biased random-key genetic algorithm with forward-backward improvement for the resource constrained project scheduling problem. Journal of Heuristics, 17, 467-486.

Gonsalves, T., Ito, A., Kawabata, R., and Itoh, K. 2008. Swarm intelligence in the optimization of software development project schedule. In Computer Software and Applications, 2008. COMPSAC'08. 32nd Annual IEEE International pp. 587-592. IEEE.

Graham, R. L., Lawler, E. L., Lenstra, J. K., and Rinnooy Kan, A. H. G. 1979. Optimization and approximation in deterministic sequencing and scheduling: a survey. Discrete Optimization, 5, 287-326.

Hallefjord, Å., and Wallace, S. W. 1998. Work patterns in project scheduling. Annals of Operations Research, 82, 1-8.

Hapke, M., Jaszkiewicz, A. and Słowiński, R. 1998 Interactive analysis of multiple-criteria project scheduling problems, European Journal of Operational Research, 107, 2, 1, 315-324.

Hartmann, S. 1998. A competitive genetic algorithm for resource-constrained project scheduling. Naval Research Logistics, 45 7: 733-50.

Hartmann, S. 1999. Project scheduling under limited resources: models, methods, and applications. Springer Verlag, Berlin 1999.

Hartmann, S. 2001. Project scheduling with multiple modes: A genetic algorithm. Annals of Operations Research, 102, 111-135.

Hartmann, S. 2002. A self-adapting genetic algorithm for project scheduling under resource constraints. Naval Research Logistics, 49 5: 433-448.

Hartmann, S., and Briskorn, D. 2010. A survey of variants and extensions of the resource-constrained project scheduling problem. European Journal of Operational Research, 207, 1-14.

Hartmann, S., and Drexl, A. 1998. Project scheduling with multiple modes: A comparison of exact algorithms. Networks, 32, 283-297.

Hartmann, S., and Kolisch, R. 2000. Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. European Journal of Operational Research, 127, 394-407.

Helgeson, W. B., and Birnie, D. P. 1961. Assembly line balancing using the ranked positional weight technique. Journal of Industrial Engineering, 12, 394-398.

Herroelen, W., De Reyck, B., and Demeulemeester, E. 1998. Resource-constrained project scheduling: a survey of recent developments. Computers & Operations Research, 25, 279-302.

Herroelen, W., Demeulemeester, E., and De Reyck, B. 2001. A note on the paper "Resource-constrained project scheduling: Notation, classification, models and methods". European Journal of Operational Research, 128, 679-688.

Holland, J. H. 1975. Adaptation in natural and artificial systems. University of Michigan Press: Ann Arbor.

Hoogeveen, H., Potts, C. N., and Woeginger, G. J. 2000. On-line scheduling on a single machine: Maximizing the number of early jobs. Operations Research Letters, 275, 193-197.

Ibrahim, K., Chik, M. A. and Hashim, U. 2014. Horrendous capacity cost of semiconductor wafer manufacturing. In Semiconductor Electronics ICSE, 2014 IEEE International Conference. 329-331.

Icmeli, O., Erenguc, S. S., and Zappe, C. J. 1993. Project scheduling problems: a survey. International Journal of Operations & Production Management, 13, 80-91.

Jarboui, B., Damak, N., Siarry, P., and Rebai, A. 2008. A combinatorial particle swarm

optimization for solving multi-mode resource-constrained project scheduling problems. Applied Mathematics and Computation, 195 1: 299-308.

Johnson, T. J. R. 1967. An algorithm for the resource constrained project scheduling problem. Ph.D. Dissertation, Massachusetts Institute of Technology, Massachusetts.

Józefowska, J., Mika, M., Różycki, R., Waligóra, G., and Węglarz, J. 2001. Simulated Annealing for Multi-Mode Resource-Constrained Project Scheduling. Annals of Operations Research, 102 1: 137-155.

Kaplan, L. A. 1988. Resource-constrained project scheduling with preemption of jobs. Ph.D. Dissertation, The University of Michigan, Michigan.

Kelley, J. E. J. 1963. The Critical Path Method: Resource Planning and Scheduling. Prentice Hall, New Jersey 1963.

Kelley, J. E. J. 1963. The critical path method: resource planning and scheduling. Prentice Hall, New Jersey.

Klein, R. 2000. Project scheduling with time-varying resource constraints. International Journal of Production Research, 38, 3937-3952.

Kochetov, Y., and Stolyar, A. 2003. Evolutionary local search with variable neighborhood for the resource constrained project scheduling problem. Proceedings of the Computer Science and Information Technologies Conference, Ufa, Russia, 2003.

Kolisch, R. 1996. Efficient priority rules for the resource-constrained project scheduling problem. Journal of Operations Management, 14, 179-192.

Kolisch, R. 1996a. Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. European Journal of Operational Research, 90, 320-333.

Kolisch, R. 1996b. Efficient priority rules for the resource-constrained project scheduling problem. Journal of Operations Management, 14, 179-192.

Kolisch, R., and Drexl, A. 1997. Local search for nonpreemptive multi-mode resource-constrained project scheduling. IIE Transactions, 29, 987-999.

Kolisch, R., and Hartmann, S. 1999. Heuristic algorithms for solving the resource-constrained project scheduling problem. In Project Scheduling: Recent Models, Algorithms, and Applications, 14, 147-178. Boston, MA: Kluwer Academic Publishers.

Kolisch, R., and Padman, R. 2001. An integrated survey of deterministic project scheduling. Omega, 29, 249-272.

Kolisch, R., and Sprecher, A. 1997. PSPLIB-A project scheduling problem library* 1:: OR Software-ORSEP Operations Research Software Exchange Program. European Journal of Operational Research, 96, 205-216.

Kolisch, R., Sprecher, A., and Drexl, A. 1995. Characterization and generation of a general class of resource-constrained project scheduling problems. Management Science, 41, 1693-1703.

Krüger, D. and Scholl, A. 2009. A heuristic solution framework for the resource constrained multi- project scheduling problem with sequence-dependent transfer times. European Journal of Operational Research, 1972, 492-508.

Laue, H. J. 1968. Efficient methods for the allocation of resources in project networks. Mathematical Methods of Operations Research, 12, 133-143.

Leachman, R., Ding, S. 2007. Integration of speed economics into decision-making for manufacturing management, International Journal of Production Economics, 107, 1, 39-55.

Leon, V. J. and Balakrishnan, R. 1995. Strength and adaptability of problem-space based neighborhoods for resource-constrained scheduling. OR Spectrum, 17 2: 173-182.

Li, K., and Willis, R. 1992. An iterative scheduling technique for resource-constrained project scheduling. European Journal of Operational Research, 56, 370-379.

Liu, Z., and Cheng, T. 2002. Scheduling with job release dates, delivery times and preemption penalties. Information Processing Letters, 822, 107-111.

Liu, Z., and Cheng, T. E. 2004. Minimizing total completion time subject to job release dates and preemption penalties. Journal of Scheduling, 74, 313-327.

Lova, A. and Tormos, P. 2001. Analysis of scheduling schemes and heuristic rules performance in resource-constrained multiproject scheduling. Annals of Operations Research, 1021-4, 263-286.

Lova, A., Tormos, P., and Barber, F. 2006. Multi-mode resource constrained project scheduling: Scheduling schemes, priority rules and mode selection rules. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial, 10, 69-86.

Lova, A., Tormos, P., Cervantes, M., and Barber, F. 2009. An efficient hybrid genetic

algorithm for scheduling projects with resource constraints and multiple execution modes. International Journal of Production Economics, 117, 302-316.

Magazine, M. J., and Hall, N. G. 1997. Generalized preemption models for single-machine dynamic scheduling problems. IIE Transactions, 295, 359-372.

Malcolm, D. G., Roseboom, J. H., Clark, C. E., and Fazar, W. 1959. Application of a technique for research and development program evaluation. Operations Research, 7, 646-669.

Marathe, R. R., and Ryan, S. M. 2005. On the validity of the geometric Brownian motion assumption. The Engineering Economist, 502, 159-192.

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

Merkle, D., Middendorf, M., and Schmeck, H. 2002. Ant colony optimization for resource-constrained project scheduling. Evolutionary Computation, IEEE Transactions, 6 4: 333-346.

Miller, L. T., and Park, C. S. 2002. Decision making under uncertainty—Real options to the rescue? The Engineering Economist, 472, 105-150.

Mönch, Lars, John W. Fowler, and Scott J. Mason. Production Planning and Control for Semiconductor Wafer Fabrication Facilities: Modeling, Analysis, and Systems. Vol. 52. Springer, 2012.

Monma, C. L., and Potts, C. N. 1993. Analysis of heuristics for preemptive parallel machine scheduling with batch setup times. Operations Research, 415, 981-993.

Montgomery, D. C. 2008. Design and analysis of experiments. 7th ed.. John Wiley & Sons Inc. 2008

Montoya-Torres, J. R., Gutierrez-Franco, E., and Pirachicán-Mayorga, C. 2010. Project scheduling with limited resources using a genetic algorithm. International Journal of Project Management, 28 6: 619-628.

Mori, M., and Tseng, C. C. 1997. A genetic algorithm for multi-mode resource constrained project scheduling problem. European Journal of Operational Research, 100, 134-141.

Neumann, K., Schwindt, C., and Zimmermann, J. 2003. Project scheduling with time

windows and scarce resources: temporal and resource-constrained project scheduling with regular and nonregular objective functions. 2nd ed.. Springer Verlag, New York 2003.

Nonobe, K., and Ibaraki, T. 2001. An improved tabu search method for the weighted constraint satisfaction problem. Information Systems and Operational Research, 39 2: 131-151.

Norman, B. A., and Bean, J. C. 1995. Random keys genetic algorithm for scheduling: unabridged version. Ann Arbor, 1001, 48109.

Nudtasomboon, N., and Randhawa, S. U. 1997. Resource-constrained project scheduling with renewable and non-renewable resources and time-resource tradeoffs. Computers & Industrial Engineering, 32, 227-242.

Okada, I., Zhang, X. F., Yang, H. Y., and Fujimura, S. 2010. A random key-based genetic algorithm approach for resource-constrained project scheduling problem with multiple modes. Proceedings of the 2010 International Multi-Conference of Engineers and Computer Scientists, 1.

Özdamar, L. 1999. A genetic algorithm approach to a general category project scheduling problem. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 29, 44-59.

Özdamar, L., and Ulusoy, G. 1995. A survey on the resource-constrained project scheduling problem. IIE Transactions, 27, 574-586.

Özdamar, L., and Ulusoy, G. 1996. A note on an iterative forward/backward scheduling technique with reference to a procedure by Li and Willis. European Journal of Operational Research, 89, 400-407.

Palpant, M., Artigues, C. and Michelon, P. 2004. LSSPER: solving the resource-constrained project scheduling problem with large neighbourhood search. Annals of Operations Research, 1311-4, 237-257.

Parikh, S. C. and Jewell, W. S. 1965. Decomposition of project networks. Management Science, 113, 444-459.

Park, S., 2001. Learning Curve Optimization and the 1986 Semiconductor Trade Agreement. Working Paper, State University of New York at Stony Brook.

Patterson, J. H. 1984. A comparison of exact approaches for solving the multiple constrained resource project scheduling problem. Management science, 30, 854-867.

Patterson, J. H., Slowinski, R., Talbot, F. B., and Weglarz, J. 1989. An algorithm for a general class of precedence and resource constrained scheduling problems. Advances in project scheduling, 3–28.

Payne, J. H. 1995. Management of multiple simultaneous projects: a state-of-the-art review. International journal of project management, 133, 163-168.

Peteghem, V. V., and Vanhoucke, M. 2010. A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem. European Journal of Operational Research, 201, 409-418.

Petrović, R. 1968. Optimization of resource allocation in project planning. Operations Research, 16, 559-568.

Pindyck, R. S. 1988. Irreversible investment, capacity choice, and the value of the firm. The American Economic Review, 78, 5, 969-985.

Pinedo, M. 2008. Scheduling: Theory, Algorithms, and Systems. 3rd ed. Springer Verlag.

Prashant Reddy, J., Kumanan, S., and Krishnaiah Chetty, O. V. 2001. Application of Petri nets and a genetic algorithm to multi-mode multi-resource constrained project scheduling. The International Journal of Advanced Manufacturing Technology, 17, 305-314.

Pritsker, A. A. B., Waiters, L. J. and Wolfe, P. M. 1969. Multiproject scheduling with limited resources: A zero-one programming approach. Management science, 161, 93-108.

Ranjbar, M., Reyck, B. De., and Kianfar, F. 2009. A hybrid scatter search for the discrete time/resource trade-Off problem in project scheduling. European Journal of Operational Research, 193 1: 35-48.

Ryan, S. M. 2004. Capacity expansion for random exponential demand growth with lead times. Management Science, 506, 740-748.

Schuurman, P., and Woeginger, G. J. 1999. Preemptive scheduling with job-dependent setup times. Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, 759-767.

Serafini, P. and Speranza, M. G. 1991. Resource Assignment in a DSS for Project scheduling. Lecture Notes in Economics and Mathematical Systems, 254.

Sivanandam, SN., and Deepa, SN. 2007. Introduction to genetic algorithms. Springer Publishing Company, Incorporated.

Slowinski, R. 1980. Two approaches to problems of resource allocation among project activities--a comparative study. Journal of the Operational Research Society, 31 8: 711-723.

Slowinski, R. and Soniewicki, B. 1994. DSS for multiobjective project scheduling. European Journal of Operational Research, 79 2: 220-229.

Spears, W., De Jong, K., Bäck, T., Fogel, D. and De Garis, H. 1993. An overview of evolutionary computation. Springer, Berlin Heidelberg.

Speranza, M. G., and Vercellis, C. 1993. Hierarchical models for multi-project planning and scheduling. European Journal of Operational Research, 64, 312-325.

Sprecher, A. 1994. Resource-constrained project scheduling: exact methods for the multi-mode case. Springer Verlag, Berlin 1994.

Sprecher, A. 2002. Network decomposition techniques for resource-constrained project scheduling. Journal of the Operational Research Society, 405-414.

Sprecher, A., and Drexl, A. 1998. Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm. European Journal of Operational Research, 107, 431-450.

Sprecher, A., Hartmann, S., and Drexl, A. 1997. An exact algorithm for project scheduling with multiple modes. OR Spectrum, 19, 195-203.

Sprecher, A., Kolisch, R., and Drexl, A. 1995. Semi-active, active, and non-delay schedules for the resource-constrained project scheduling problem. European Journal of Operational Research, 80, 94-102.

Talbot, F. B. 1982. Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case. Management Science, 28, 1197-1210.

Tkindt, V., and Billaut, J. C. 2006. Multicriteria scheduling: theory, models and algorithms. Springer Verlag, Berlin 2006.

Tormos, P., and Lova, A. 2001. A competitive heuristic solution technique for resource-constrained project scheduling. Annals of Operations Research, 102, 65-81.

Tormos, P., and Lova, A. 2003. An efficient multi-pass heuristic for project scheduling with constrained resources. International Journal of Production Research, 41, 1071-1086.

Tsay, Ruey S. 2002. Analysis of Financial Time Series, New York, John Wiley and Sons,

Inc.

Valls, V., Ballestín, F., and Quintanilla, S. 2004. A population-based approach to the resource-constrained project scheduling problem. Annals of Operations Research, 131 1: 305-324.

Valls, V., Ballestin, F., and Quintanilla, S. 2005. Justification and RCPSP: A technique that pays. European Journal of Operational Research, 165, 375-386.

Valls, V., Ballestin, F., and Quintanilla, S. 2008. A hybrid genetic algorithm for the resource-constrained project scheduling problem. European Journal of Operational Research, 185 2: 495-508.

Valls, V., Laguna, M., Lino, P., Pérez, A., and Quinatanilla, S. 1999. Project scheduling with stochastic activity interruptions. International Series in Operations Research and Management Science, 333-354.

Valls, V., Quintanilla, S., and Ballesti'n, F. 2003. Resource-constrained project scheduling: a critical activity reordering heuristic. European Journal of Operational Research, 149, 282-301.

Vanhoucke, M., and Debels, D. 2008. The impact of various activity assumptions on the lead time and resource utilization of resource-constrained projects. Computers & Industrial Engineering, 54, 140-154.

Voß, S and Witt, A 2007. Hybrid flow shop scheduling as a multi-mode multi-project scheduling problem with batching requirements: A real-world application, International Journal of Production Economics, 105, 2, 445-458.

Wang, H., Lin, D., and Li, M. Q. 2005. A competitive genetic algorithm for resource-constrained project scheduling problem. In Machine Learning and Cybernetics. Proceedings of 2005 International Conference on 5. 2945-2949.

Wang, J. 2005 Constraint-based schedule repair for product development projects with time-limited constraints, International Journal of Production Economics, 95, 3, 399-414.

Węglarz, J., Józefowska, J., Mika, M., and Waligóra, G. 2011. Project scheduling with finite or infinite number of activity processing modes–A survey. European Journal of Operational Research, 208, 177-205.

Whitt, W. 1981. The stationary distribution of a stochastic clearing process. Operations Research, 292, 294-308.

Wiest, J. D. 1964. Some properties of schedules for large projects with limited resources. Operations Research, 12, 395-418.

Wiest, J. D. 1967. A heuristic model for scheduling large projects with limited resources. Management Science, 13, 359-377.

Zamani, R. 2004. An efficient time-windowing procedure for scheduling projects under multiple resource constraints. OR Spectrum, 263, 423-440.

Zamani, R. 2011. A hybrid decomposition procedure for scheduling projects under multiple resource constraints. Operational Research, 11, 93-111.

Zdrzałka, S. 1994. Preemptive scheduling with release dates, delivery times and sequence independent setup times. European Journal of Operational Research, 761, 60-71.

Zhang, H., Tam, C. M., and Li, H. 2006. Multimode project scheduling based on particle swarm optimization. Computer-Aided Civil and Infrastructure Engineering, 21 2: 93-103.

Zheng, F., Fung, S. P., Chan, W., Chin, F. Y., Poon, C. K., and Wong, P. W. 2006. Improved on-line broadcast scheduling with deadlines. Computing and combinatorics,320-329.

Zheng, F., Xu, Y., and Zhang, E. 2007. On-line production order scheduling with preemption penalties. Journal of Combinatorial Optimization, 132, 189-204.

Zhu, G., Bard, J. F., and Yu, G. 2006. A branch-and-cut procedure for the multimode resource-constrained project-scheduling problem. INFORMS Journal on Computing, 18, 377.